CLOUD COMPUTING 2012

The Third International Conference on Cloud Computing, GRIDs, and Virtualization

ISBN: 978-1-61208-216-5

July 22-27, 2012

Nice, France

CLOUD COMPUTING 2012 Editors

Wolf Zimmermann, Martin-Luther University Halle-Wittenberg, Germany

Yong Woo Lee, University of Seoul, Korea

Yuri Demchenko, University of Amsterdam, Netherlands

CLOUD COMPUTING 2012

Foreword

Cloud computing is a normal evolution of distributed computing combined with Service-oriented architecture, leveraging most of the GRID features and Virtualization merits. The technology foundations for cloud computing led to a new approach of reusing what was achieved in GRID computing with support from virtualization.

The Third International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012), held between July 22 and 27, 2012, in Nice, France, intended to prospect the applications supported by the new paradigm and validate the techniques and the mechanisms. A complementary target was to identify the open issues and the challenges to be fixed, especially on security, privacy, and inter- and intra-clouds protocols.

We welcomed technical papers presenting research and practical results, position papers addressing the pros and cons of specific proposals, such as those being discussed in the standard fora or in industry consortia, survey papers addressing the key problems and solutions on any of the above topics short papers on work in progress, and panel proposals.

We take here the opportunity to warmly thank all the members of the CLOUD COMPUTING 2012 Technical Program Committee, as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to CLOUD COMPUTING 2012. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the CLOUD COMPUTING 2012 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that CLOUD COMPUTING 2012 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the area of cloud computing.

We are convinced that the participants found the event useful and communications very open. We hope Côte d'Azur provided a pleasant environment during the conference and everyone saved some time for exploring the Mediterranean Coast.

CLOUD COMPUTING 2012 Chairs:

CLOUD COMPUTING Advisory Chairs
Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Wolf Zimmermann, Martin-Luther University Halle-Wittenberg, Germany
Yong Woo Lee, University of Seoul, Korea

CLOUD COMPUTING 2012 Industry/Research Chairs
Wolfgang Gentzsch, Senior HPC Consultant, Germany
Tony Shan, Keane Inc., USA
Donglin Xia, Microsoft Corporation, USA

CLOUD COMPUTING 2012 Research Institutes Chairs
Jorge Ejarque, Barcelona Supercomputing Center, Spain
Leslie Liu, IBM T.J Watson Research, USA

COULD COMPUTING 2012 Special Area Chairs
Virtualization
Toan Nguyen, INRIA, France

GRID
Jorge Ejarque, Barcelona Supercomputing Center, Spain
Javier Diaz, Indiana University, USA

Autonomic computing
Ivan Rodero, Rutgers the State University of New Jersey/NSF Center for Autonomic Computing, USA
Hong Zhu, Oxford Brookes University, UK

Service-oriented
Qi Yu, Rochester Institute of Technology, USA

Security
Chih-Cheng Hung, Southern Polytechnic State University - Marietta, USA

Platforms
Arden Agopyan, IBM Central & Eastern Europe, Russia, Middle East & Africa (CEE & MEA), Turkey
Dariusz Król, Academic Computer Center CYFRONET - Cracow, Poland

CLOUD COMPUTING 2012

Committee

CLOUD COMPUTING Advisory Chairs

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Wolf Zimmermann, Martin-Luther University Halle-Wittenberg, Germany
Yong Woo Lee, University of Seoul, Korea

CLOUD COMPUTING 2012 Industry/Research Chairs

Wolfgang Gentzsch, Senior HPC Consultant, Germany
Tony Shan, Keane Inc., USA
Donglin Xia, Microsoft Corporation, USA

CLOUD COMPUTING 2012 Research Institutes Chairs

Jorge Ejarque, Barcelona Supercomputing Center, Spain
Leslie Liu, IBM T.J Watson Research, USA

COULD COMPUTING 2012 Special Area Chairs

Virtualization
Toan Nguyen, INRIA, France

GRID
Jorge Ejarque, Barcelona Supercomputing Center, Spain
Javier Diaz, Indiana University, USA

Autonomic computing
Ivan Rodero, Rutgers the State University of New Jersey/NSF Center for Autonomic Computing, USA
Hong Zhu, Oxford Brookes University, UK

Service-oriented
Qi Yu, Rochester Institute of Technology, USA

Security
Chih-Cheng Hung, Southern Polytechnic State University - Marietta, USA

Platforms
Arden Agopyan, IBM Central & Eastern Europe, Russia, Middle East & Africa (CEE & MEA), Turkey
Dariusz Król, Academic Computer Center CYFRONET - Cracow, Poland

CLOUD COMPUTING 2012 Technical Program Committee

Jemal Abawajy, Deakin University - Victoria, Australia
Imad Abbadi, University of Oxford, UK
Arden Agopyan, IBM Central & Eastern Europe, Russia & CIS (CEE), Turkey
Ali Beklen, IBM Turkey - Software Group, Turkey
Simona Bernardi, Centro Universitario de la Defensa / Academia General Militar - Zaragoza, Spain
Nik Bessis, University of Derby, UK
William Buchanan, Edinburgh Napier University, UK
Massimo Canonico, University of Piemonte Orientale, Italy
Juan-Vicente Capella-Hernández, Universitat Politècnica de València, Spain
Simon Caton, Karlsruhe Institute of Technology, Germany
Hsi-Ya Chang, National Center for High-Performance Computing (NCHC), Taiwan
Antonin Chazalet, France Télécom - Orange, France
Shiping Chen, CSIRO ICT Centre, Australia
Ye Chen, Microsoft Corp., USA
Yixin Chen, Washington University in St. Louis, USA
Zhixiong Chen, Mercy College - NY, USA
William Cheng-Chung Chu(朱正忠), Tunghai University, Taiwan
Antonio Corradi, Università di Bologna, Italy
Marcelo Corrales, University of Hanover, Germany
Yuri Demchenko, University of Amsterdam, The Netherlands
Nirmit Desai, IBM Research - Bangalore, India
Edna Dias Canedo, Universidade de Brasília - UnB Gama, Brazil
Javier Diaz, Pervasive Technology Institute/Indiana University, USA
Qiang Duan, Pennsylvania State University Abington College, USA
Jorge Ejarque Artigas , Barcelona Supercomputing Center, Spain
Atilla Elçi, Suleyman Demirel University - Isparta, Turkey
Khalil El-Khatib, University of Ontario Institute of Technology - Oshawa, Canada
Umar Farooq, Amazon.com - Seattle, USA
Sören Frey, University of Kiel, Germany
Wolfgang Gentzsch, Senior HPC Consultant, Germany
Nils Grushka, NEC Laboratories Europe - Heidelberg, Germany
Weili Han, Fudan University, China
Haiwu He, INRIA, France
Neil Chue Hong, University of Edinburgh, UK
Kenneth Hopkinson, Air Force Institute of Technology - Dayton, USA
Chih-Cheng Hung, Southern Polytechnic State University - Marietta, USA
Anca Daniela Ionita, University "Politehnica" of Bucharest, Romania
César A. F. De Rose, Catholic University of Rio Grande Sul (PUCRS), Brazil
Luca Foschini, Università degli Studi di Bologna, Italy
Song Fu, University of North Texas - Denton, USA
Spyridon Gogouvitis, National Technical University of Athens, Greece
Yi-Ke Guo, Imperial College London, UK
Richard Hill, University of Derby, UK
Benoit Hudzia, SAP Research, France
Ming Jiang, University of Leeds, UK
Xuxian Jiang, North Carolina State University, USA

Sabri Pllana, University of Vienna, Austria
Jari Porras, Lappeenranta University of Technology, Finland
Thomas E. Potok, Oak Ridge National Laboratory, USA
Alfonso Niño Ramos, Universidad de Castilla-La Mancha-Ciudad Real, Spain
Christoph Reich, Hochschule Furtwangen University, Germany
Sebastian Rieger, Steinbuch Centre for Computing (SCC)/Karlsruher Institut für Technologie (KIT), Germany
Philip Robinson, SAP Research - Belfast, UK
Benny Rochwerger, IBM Haifa Research Lab., Israel
Ivanm Rodero, NSF Center for Autonomic Computing, Rutgers the State University of New Jersey - Piscataway, USA
Majd F. Sakr, Carnegie Mellon University in Qatar, Qatar
Iñigo San Aniceto Orbegozo, Universidad Complutense de Madrid, Spain
Volker Sander, FH Aachen University of Applied Sciences, Germany
Gregor Schiele, University of Mannheim, Germany
Igor Sfiligoi, University of California San Diego-La Jolla, USA
Alan Sill, Texas Tech University, USA
Raül Sirvent, Barcelona Supercomputing Center, Spain
Luca Spalazzi, Università Politecnica delle Marche - Ancona, Italy
George Spanoudakis, City University London, UK
Jie Tao, Steinbuch Centre for Computing/Karlsruhe Institute of Technology (KIT), Germany
Sofie Van Hoecke, Ghent University, Belgium
Luis Vaquero, HP Labs., Spain
Michael Gr. Vassilakopoulos, University of Central Greece - Lamia, Greece
Jose Luis Vazquez-Poletti, Universidad Complutense de Madrid, Spain
Salvatore Venticinque, Second University of Naples - Aversa, Italy
Mario Jose Villamizar Cano, Universidad de loa Andes - Bogotá, Colombia
Eugen Volk, High Performance Computing Center Stuttgart (HLRS) - Stuttgart, Germany
Andy Ju An Wang, Southern Polytechnic State University - Marietta, USA
Cho-Li Wang, University of Hong Kong, China
Zhi Wang, North Carolina State University, USA
Philipp Wieder, Gesellschaft fuer wissenschaftliche Datenverarbeitung mbH - Goettingen (GWDG), Germany
Yong Woo Lee, University of Seoul. Korea
Christos Xenakis, University of Piraeus, Greece
Hiroshi Yamada, Keio University, Japan
Chao-Tung Yang, Tunghai University, Taiwan R.O.C.
Hongji Yang, De Montfort University (DMU) - Leicester, UK
Yanjiang Yang, Institute for Infocomm Research, Singapore
Jinhui Yao, CSIRO ICT Centre, Australia
Qi Yu, Rochester Institute of Technology, USA
Jong P. Yoon, Mercy College - Dobbs Ferry, USA
Jie Yu, National University of Defense Technology (NUDT), China
Massimo Villari, University of Messina, Italy
Baokang Zhao, National University of Defence Technology, China
Zibin (Ben) Zheng, Shenzhen Research Institute, The Chinese University of Hong Kong, Hong Kong
Hong Zhu, Oxford Brookes University, UK
Wolf Zimmermann, University of Halle, Germany

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Proposed Joint Multiple Resource Allocation Method for Cloud Computing Services with Heterogeneous QoS

Yuuki Awano

Dept. of Computer and Information Science
Seikei University
Musashino, Tokyo, Japan
us092008@cc.seikei.ac.jp

Shin-ichi Kuribayashi

Dept. of Computer and Information Science
Seikei University
Musashino, Tokyo, Japan
kuribayashi@st.seikei.ac.jp

Abstract - ***This paper proposes to enhance the proposed joint multiple resource allocation method so that it can handle multiple heterogeneous resource-attributes. The basic idea is to identify the key resource-attribute first which has the most impact on resource allocation and to select the resources which provide the lowest Quality of Service for the key resource-attribute as it satisfies required Quality of Service. It is demonstrated by simulation evaluations that the enhanced method can reduce the total amount of resources up to 30%, compared with the conventional methods. The enhanced method could be also effective to the resource allocation in a hybrid-cloud in which either a private-cloud or a public-cloud is selected depending on the required security level.***

Keywords - *cloud computing; heterogeneous QoS; joint multiple resource allocation; hybrid cloud.*

1. Introduction

Cloud computing services are allow the user to rent, only at the time when needed, only a desired amount of computing resources (ex. processing ability, storage capacity) out of a huge mass of distributed computing resources without worrying about the locations or internal structures of these resources [1]-[5]. The popularity of cloud computing owes to the increase in the network speed, and to the fact that virtualization and grid computing technologies have become commercially available. It is anticipated that enterprises will accelerate their migration from building and owning their own systems to renting cloud computing services, because cloud computing services are easy to use and can reduce both business costs and environmental loads.

As cloud computing services rapidly expand their customer base, it has become important to provide them economically. To do so, it is essential to optimize resource allocation under the assumption that the required amount of resource can be taken from a common resource pool and rented out to the user on an hourly basis. In addition, to be able to provide processing ability and storage capacity, it is necessary to allocate simultaneously a network bandwidth to access them and the necessary power capacity. Therefore, it is necessary to allocate multiple types of resources (such as processing ability, bandwidth, and storage capacity) simultaneously in a coordinated manner, instead of allocate -ing each type of resource independently [6]-[8].

Moreover, it is necessary to consider not only the required resource size but also resource-attributes in actual resource allocation. Resource-attributes of bandwidth, for example, are network delay time, packet loss probability, etc. If it is required to respond quickly, bandwidth with a short network delay time should be selected from a group of bandwidths. Computation time is one of resource-attributes of processing ability. References [6] and [7] consider a model in which there are multiple data centers with processing ability and bandwidth to access them, and proposed the joint multiple resource allocation method (referred to as "**Method 3**").

The basic idea of Method 3 is to select a bandwidth with the longest network delay time from a group of bandwidths that satisfy the condition on service time. It is for maximizing the possibility to accept requests later, which need a short network delay time. It was demonstrated by simulation evaluations that Method 3 can handle more requests than the case where network delay time is not taken into account, and thus can reduce the required amount of resources by up to 20% [6],[7].

Method 3 takes into account only a single resource-attribute of network bandwidth (namely, network delay time). However, it is usually necessary to consider multiple heterogeneous resource-attributes in a real cloud computing environment. It is proposed to enhance the proposed method, Method 3, to handle multiple heterogeneous resource-attributes. The enhanced-Method 3 could be also effective to the efficient resource allocation in hybrid clouds [9]. In a hybrid cloud, transactions that require a critical security are executed using private clouds only and other transactions that require a normal security may be executed using more economical public clouds. For the preliminary evaluation, this paper assumes two types of resources (processing ability and bandwidth), loss-system based services and the static resource allocation.

The rest of this paper is organized as follows. Section 2 explains related works. Section 3 provides the resource allocation model for cloud computing environments. Section 4 proposes to enhance the proposed joint multiple resource allocation method, Method 3, to be able to handle multiple heterogeneous resource-attributes. Section 5 describes simulation evaluations which confirm the effectiveness of the enhanced-Method 3 (referred to as "**Method 3E**"). Finally, Section 6 gives the conclusions.

2. Related work

Resource allocation for clouds has been studied very extensively in References [10]-[19]. References [14],[15] have proposed automatic or autonomous resource management in cloud computing. Reference [10] has proposed the heuristic algorithm for optimal allocation of cloud resources. Reference [16] has presented the system architecture to allocate resources assuming heterogeneous hardware and resource demands. References [11] and [12]

 ISBN: 978-1-61208-216-5

have proposed market-oriented allocation of resources including auction method. Reference [13] has proposed to use game-theory to solve the problem of resource allocation. Energy aware resource allocation methods for clouds have been proposed [18]-[20].

However, most of conventional studies on resource allocation in a cloud computing environments are treating each resource-type individually. To the best our knowledge, the cloud resource allocation has not been fully studied which assumes that multiple resources are allocated simultaneously to each service request and there are multiple heterogeneous resource-attributes for each resource-type.

3. Resource allocation model for cloud comput -ing environments

3.1 Resource allocation model

The resource allocation model for a cloud computing environment is such that multiple resources with heterogeneous resource-attributes taken from a common resource pool are allocated simultaneously to each request for a certain period. For the preliminary evaluation, this Section considers two resource-types: processing ability and bandwidth. It is assumed that the physical facilities for providing cloud computing services are distributed over multiple data centers, in order to make it easy to increase the number of the facilities when demand increases, to allow load balancing, and to enhance reliability.

The cloud resource allocation model that incorporates these assumptions is illustrated in Figure 1. Each center has servers which provide processing ability and network devices which provide the bandwidth to access the servers. The maximum size of processing ability and bandwidth at center j (j=1,2,..,k) is assumed to be C_{maxj} and N_{maxj} respectively. The different resource-attributes of processing ability and network bandwidth could be provided by each center.

When a service request is generated, one optimal center is selected from among k centers, and the processing ability and bandwidth in that center are allocated simultaneously to the request for a certain period. If no center has sufficient resources for a new request, the request is rejected. These

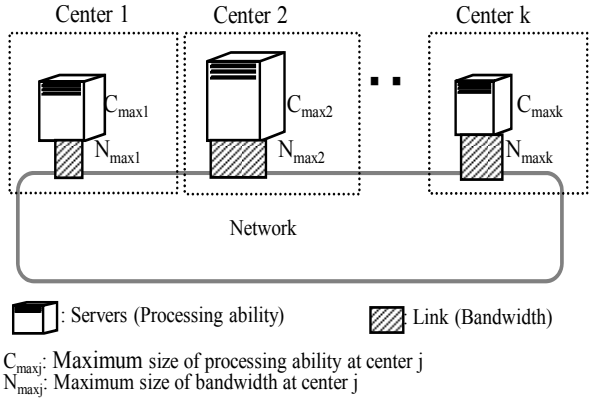


Figure 1. Resource allocation model for cloud computing environments

are the same as those in References [6]-[8].

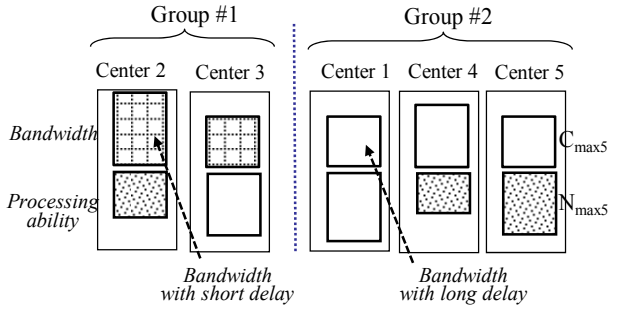
3.2 Guidelines of joint multiple resource allocation assuming multiple heterogeneous resource-attributes

In general, a cloud computing environment includes multiple resource-types and multiple resource-attributes for each resource-type. For example, resource-attributes of bandwidth are network delay time, packet loss probability, required electric power capacity, etc. If a request requires quick-response, it is needed to select one with a short network delay from a group of bandwidths. On the contrary, if a request requires a less power consumption, it is needed to select a bandwidth whose power consumption is small. Resource-attributes of processing ability are computation time, memory size, required electric power capacity, etc. In a hybrid cloud, resource-attributes may additionally include the levels of security (critical or normal) and reliability.

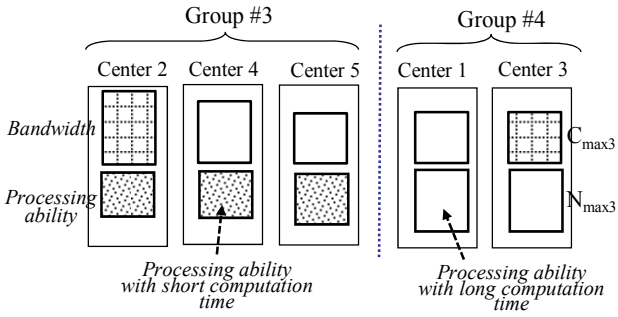
The center selection algorithm with Method 3 proposed in References [6] and [7] is explained with Figure 2. Figure 2 is just an example. There are five centers in different locations, and that each center has two resource-types: bandwidth and processing ability. In Figure 2(1), centers are divided to multiple groups according a resource-attribute (network delay time) of bandwidth. That is, centers in Group #1 can provide bandwidth with short delay and centers in Group #2 provide bandwidth with long delay. If a request's requirement on response is not so stringent, Method 3 first tries to select a center from Group #2, and only when there is no center with appropriate resources available in this group, it selects a center from Group #1. This approach makes it possible to meet more future requests later, which need a short delay. We next consider center groups taking a resource-attribute (computation time) of processing ability into consideration, as shown in Figure 2(2). If a request has no stringent requirement on computation time, Method 3 first attempts to select a center from Group #4, and only when there is no center with appropriate resources available in this group, it selects a center from Group #3.

In this way, the priority with which a center group is selected differs between Figure 2(1) and Figure 2(2). If a request with no strong requirement is allocated to a center 4 or center 5 taking only one resource-type into consideration, for example, then fewer resources are likely to be available later when requests with a stringent requirement on processing ability are generated. Therefore, it is necessary to take both multiple resource-types and multiple resource-attributes into consideration simultaneously in selecting a center. Moreover, it would be necessary to consider a new center group if requests with a stringent requirement on both bandwidth and processing ability are generated. Even if center groups are created taking all the resource-types and resource-attributes into consideration, the combinations of different requirements can be too numerous to be manageable, and it would not be easy to develop a guideline as to the sequence of priority in which center groups are to be selected.

 ISBN: 978-1-61208-216-5



(1) Grouping with resource-attribute of bandwidth



(2) Grouping with resource-attribute of processing ability

Figure 2. Example of resource allocation assuming heterogeneous resource-attributes

Therefore, the simplified algorithm adopted by the authors in References [6] and [7] would be also applicable here.

The above guidelines could also be effective to the resource allocation in a hybrid-cloud. In hybrid-cloud, either a private or a public cloud will be selected depending on the required levels of security or reliability, as shown in Figure 3. Requests that require a normal security should be allocated to the public cloud first, and then to the private cloud so that the resources in the private cloud can be kept available for future requests that require a critical security. It turns out that security level or reliability level need to be considered as one of resource-attributes.

4. Enhanced joint multiple resource allocation supporting multiple resource-attributes

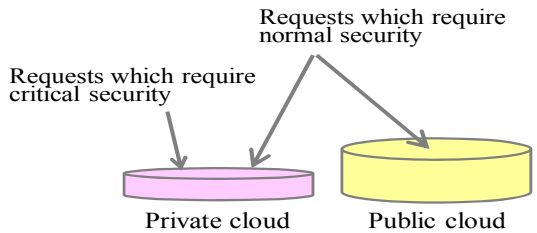


Figure 3. Services with both private and public cloud

4.1 Principle

As discussed in Section 3.2, it is difficult to take multiple resource-types and multiple resource-attributes for each resource-type into consideration simultaneously. It is proposed to apply the same principle adopted in References [6] and [7]. That is, it is proposed to allocate resources focusing on the most important resource-attribute (hereafter referred to as the "**key resource-attribute**"). The key resource-attribute is decided by the system (not by the user), and can be different for each request.

The resource allocation algorithm of enhanced Method 3 (Method 3E) is explained in the next Section 4.2, which adopts the concept of key resource-attribute above.

4.2 Resource allocation algorithm of Method 3E

4.2.1 Identification of key resource-attribute

An attribute with the lowest **relative amount of resource** is selected as key resource-attribute from among multiple resource-attributes for all resource-types. The relative amount of resource, M_g, for resource-attribute g is calculated by

$$M_g = d_{2g}/d_{1g} \quad (1)$$

where d_{1g} is the sum of resources which offer resource-attribute g and all the resources which offer higher quality of service (QoS) than resource-attribute g. d_{2g} is the expected amount of resources with resource-attribute g required by all requests.

For example, if there are bandwidths with network delay time of 50ms and those with network delay time of 200ms, d_{1g} for network delay time of 200ms includes not only the amount of bandwidths with network delay time of 200ms but also the amount of bandwidths with network delay time of 50ms.

It is also proposed that resource-attribute g is not selected as key resource-attribute when the ratio of the number of requests requiring resource-attribute g to the total number of requests is lower than a certain value (e.g., 10%).

4.2.2 Identification of a center group

Here we focus on the resource-type associated with the key resource-attribute, and classify center groups into three categories: Center Group X, which contains resources that provide lower QoS than that provided by the key resource-attribute, Center Group Y, which contains resources that provide QoS equal to that provided by the key resource-attribute, and Center Group Z, which contains resources that provide higher QoS than that provided by the key resource-attribute. In some cases, Center Group X or Center Group Z may not exist.

4.2.3 Selection of a center

- A center that can provide multiple resources required by the request is selected. If there is no center that can satisfy the requirements, the request is rejected.
- If there are several selectable centers in the center group, one is selected either at random or sequentially.
- A center is selected as follows depending on the QoS required by the request.

i) If the request requires lower QoS than that associated with the key resource-attribute, it is tried to select a center in Center Group X. If there is no selectable center in the group,

a selectable center in Center Group Y or in Center Group Z is selected in this order.

ii) If the request requires the QoS associated with the key resource-attribute, a center is selected in Center Group Y. If there is no selectable center there, a center in Center Group Z is selected.

iii) If the request requires higher QoS than that associated with the key resource-attribute, it is tried to select a center in Center Group Z.

- The multiple resources with required resource-attribute in the selected center are allocated to the request simultaneously.
- When the service time to the request has expired, all the resources allocated in Section 4.2.4 are released.

5. Simulation evaluation

5.1 Evaluation model

1) Method 3E proposed in Section 4.2 is evaluated using a (self-made) simulator written in the C language.

2) For the preliminary evaluation, we consider only two resource-types: processing ability and bandwidth. 'Computation time' is used as a resource-attribute of processing ability and 'network delay time' as that of bandwidth here.

3) Figure 1 with k=3 is assumed as the resource allocation model. That is, there are three centers, Centers 1, 2 and 3, which provide resources with different resource-attributes as follows:

<Attribute: Computation time>
- long for Centers 1 and 3
- short (referred to as 'high_1') for Center 2

< Attribute: Network delay time>
- long for Centers 1 and 2
- short (referred to as 'high_2') for Center 3

Any attribute other than high_1 or high_2 is referred to as 'normal' in this Section.

4) Three types of requests are considered here:

<Type_1> Requests that can be satisfied with attribute 'normal' for both computation time and network delay time. Selectable resources exist in any center. The probability at which type_1 request occurs is designated as q_1.

<Type_2> Requests that can be satisfied only with attribute 'high_1' for computation time, but can be satisfied with attribute 'normal' for network delay time. Selectable resources exist only in Center 2. The probability at which type_2 request occurs is designated as q_2.

<Type_3> Requests that can be satisfied only with attribute 'high_2' for network delay time, but can be satisfied with attribute 'normal' for computation time. Selectable resources exist only in Center 3. The probability at which type_3 request occurs is designated as q_3 ($q_1+q_2+q_3=1$).

5) When a new request is generated, one appropriate center is selected according to the resource allocation algorithm (Method 3E) in Section 4.2 and then both processing ability and bandwidth from that center is allocated to the request simultaneously. For the purpose of comparison, the proposed method, Method 3, and Round Robin method (referred to as "RR Method") in which a center is selected in sequence, are also evaluated in the simulation. Method 3, which does not have the concept of key resource-attribute, considers only network delay time here.

6) The size of required processing ability and bandwidth by each request is assumed to follow a Gaussian distribution (dispersion is 5). Let C and N be the averages of the distributions of processing ability and bandwidth respectively.

7) The intervals between requests follow an exponential distribution with the average, r. The length of resource holding time, H, is constant. All allocated resources are released simultaneously after the resource holding time expires.

8) The pattern in which requests occur is a repetition of $\{C=a_1, N=b_1; C=a_2, N=b_2; \ldots; C=a_w, N=b_w\}$, where w is the number of requests that occur within one cycle of repetition, a_u (u=1~w) is the size of C of the u-th request, and b_u (u=1~w) is the size of N of the u-th request.

5. 2 Simulation results and evaluation

The simulation results are shown in Figures 4, 5 and 6. The horizontal axis shows the probability q_1 at which type_1 request occurs. The value of q_2 and q_3 is set to $(1- q_1)/2$ respectively. The vertical axis of Figures 4 and 5 shows the average request loss probability. The vertical axis of Figure 6 shows the ratio of required amount of resources by Method 3E and those by RR method, on the condition of keeping the same average request loss probability. Figure 4(1) shows evaluation results for the case where the request generation pattern is uniform. Figure 4(2) shows the case where it is uneven (i.e., rise and fall in anti-phase). Figure 5 is intended to evaluate the impact of the unevenness of the total amount of resources between centers. While the total amount of resources in each center is the same in Figure 4, the total resource amount of Center 3 is twice that of Center 1 or Center 2 in Figure 5. Figure 5(1) and 5(2) show the total average request loss probability and the request loss probability for each request-type respectively. The parenthesis following Method 3 or Method 3E in Figure 5 indicates the request-type.

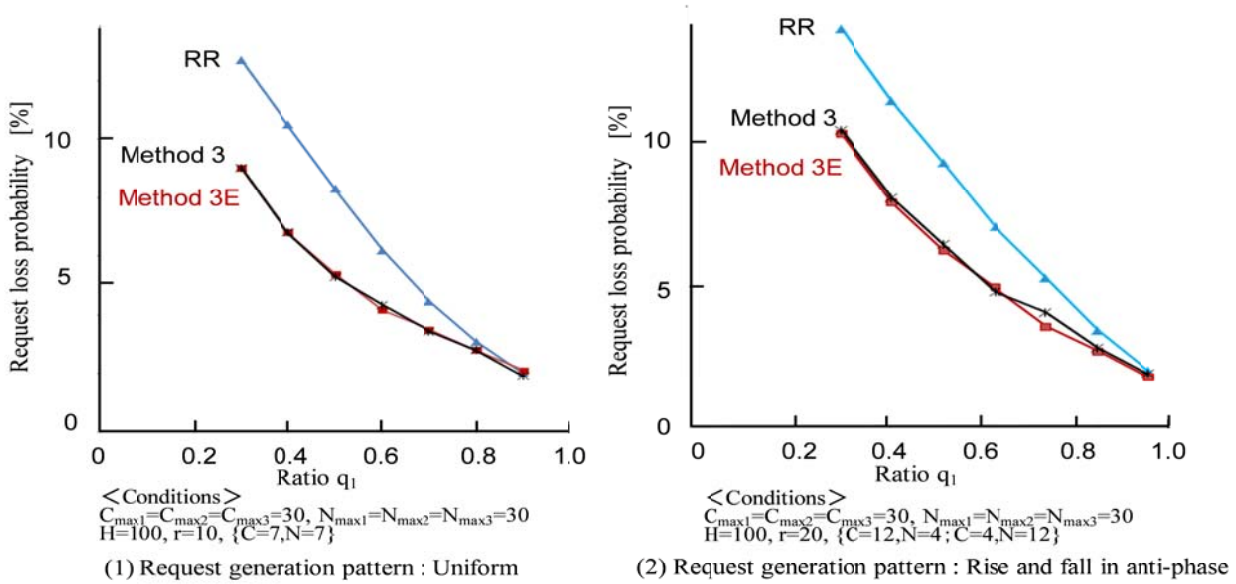
The following points are clear from these Figures:

i) Except for the area near q_1=1.0 (i.e., the area where almost all requests are type_1), the request loss probabilities of Method 3E and Method 3 are smaller than that of the RR method by up to 30%. This tendency is effective regardless of the request generation pattern.

<Reason> Even when requests are type_1, RR method tends to select Center 2 or Center 3 more often compared with Method 3E or Method 3. The reason why there is not much difference in results between Methods 3E and 3 is that type_1 requests use almost all resources in Centers 1, 2 and 3 when q_1 comes close to 1.0.

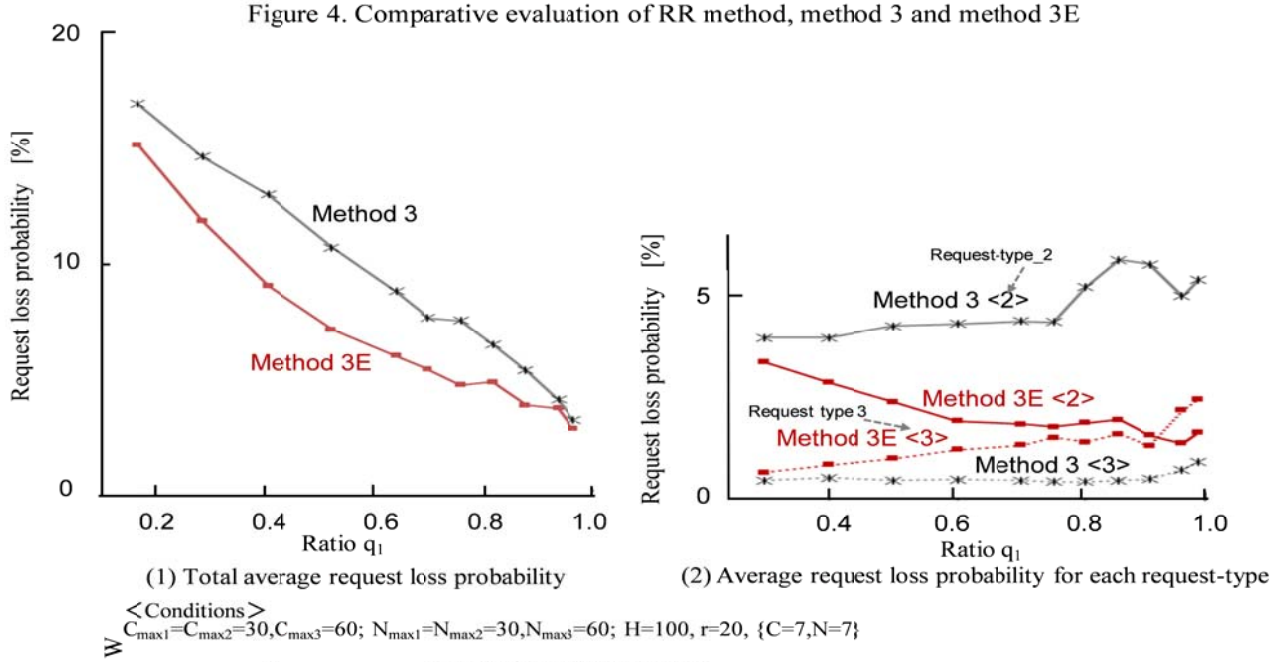
ii) Except for the area near q_1=1.0, the request loss probability of Method 3E is smaller than that of Method 3 when the total resource amount used by each request-type is

 ISBN: 978-1-61208-216-5



(1) Request generation pattern : Uniform (2) Request generation pattern : Rise and fall in anti-phase

Figure 4. Comparative evaluation of RR method, method 3 and method 3E



(1) Total average request loss probability (2) Average request loss probability for each request-type

Figure 5. Comparative evaluation of method 3E and method 3

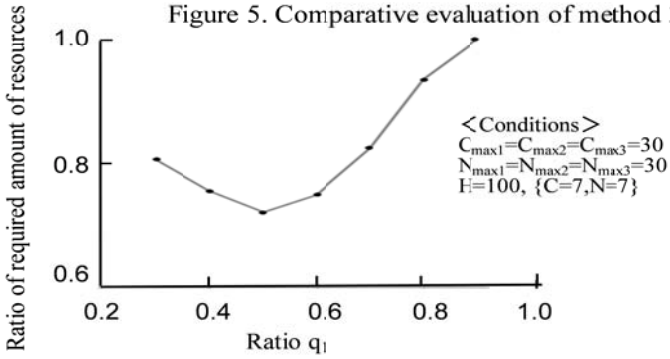


Figure 6. Ratio of required amount of resources

different. The differences in the request loss probability between different request-types can also be made smaller by Method 3E.

< Reason> Method 3, which does not have the concept of key resource-attribute and takes attribute high_2 into consideration, goes on to select Center 2 for type_1 requests if the appropriate resources are not available in Center 1. Therefore, the amount of resources available in Center 2 decreases rather than that in Center 3. As a result, the request loss probability of type_2 requests increases, which require

resources with attribute high_1 (key resource-attribute here).

In Method 3E, the key resource attribute is set to attribute high_1, and when type_1 requests cannot use Center 1, they attempt to select Center 3, which has more resources. As a result, more resources are kept available in Center 2 than in the case of using Method 3, and it is possible to reduce the request loss probability of type_2 requests. As the value of q_1 becomes small, the number of type_2 requests to handle increases and the request loss probability of type_2 will increase also by Method 3E.

iii) The total amount of resources required for keeping the same request loss probability could be smaller with Method 3E than with RR method by up to 30%.

6. Conclusion and Future Work

This paper has enhanced the proposed joint multiple resource allocation method (Method 3) so that it can handle multiple heterogeneous resource-attributes. The basic idea of the enhanced Method 3 (**Method 3E**) is to identify the key resource-attribute first which has the most impact on resource allocation and to select the resources which provide the lowest QoS for the key resource-attribute as it satisfies required QoS, so that future requests with more stringent requirement can still find available resources.

It has been demonstrated by simulation evaluations that Method 3E can reduce the total amount of resources up to 30%, compared with the conventional methods. Method 3E could be also effective to the resource allocation in a hybrid-cloud in which either a private-cloud or a public-cloud is used depending on the required level of security.

For the preliminary evaluation, we have limited the numbers of request types, centers, resource-types, and resource-attributes to small numbers in our simulation evaluation. We will make an evaluation with larger numbers of these to confirm the effectiveness of the proposed method and to identify the conditions in which the proposed method is effective. Moreover, the value of resource-attribute related to bandwidth may change with the location where a request occurs. For example, the procedure to regulate the access from a distant location temporarily when the amount of available resources are less than the threshold value is required to be studied.

Acknowledgement

This work was supported in part by the Japan Society for the Promotion of Science through a Grant-in-Aid for Scientific Research (C) (21500041).

References

[1] G.Reese: “Cloud Application Architecture”, O’Reilly& Associates, Inc., Apr. 2009.
[2] J.W.Rittinghouse and J.F.Ransone: “Cloud Computing: Impremention, Management, and Security”, CRC Press LLC, Aug. 2009.
[3] P.Mell and T.Grance, “Effectively and securely Using the Cloud Computing Paradigm”, NIST, Information Technology Lab., July 2009.
[4] P.Mell and T.Grance : “The NIST Definition of Cloud Computing” Version 15, 2009.
[5]Z.Hang, L.Cheng, and R.Boutaba, “Cloud compuing: state-of-the-art and research challenges”, J Internet Serv Apl, Jan. 2010.
[6] S.Kuribayashi, “Optimal Joint Multiple Resource Allocation Method for Cloud Computing Environments”, International Journal of Research and Reviews in Computer Science (IJRRCS), Vol. 2, No.1, Feb. 2011.
[7] S.Tsumura and S.Kuribayashi: “Simultaneous allocation of multiple resources for computer communications networks”, In Proceeding of 12th Asia-Pacific Conference on Communications (APCC2006), 2F-4, Aug. 2006.
[8] K.Mochizuki and S.Kuribayashi, “Evaluation of optimal resource allocation method for cloud computing environments with limited electric power capacity”, Proceeding of the 14-th International Conference on Network-Based Information Systems (NBiS-2011), Sep. 2011.
[9] H.Zhang, G.Jiang, K.Yoshihira, H.Chen, and A.Saxena, “Intelligent workload factoring for a hybrid cloud computing model”, Proceedings of the 2009 IEEE Congress on Services (Services’09), July 2009.
[10] B. Soumya, M. Indrajit, and P. Mahanti, “Cloud computing initiative using modified ant colony framework,” in In the World Academy of Science, Engineering and Technology 56, 2009.
[11]R.Buyya, C.S. Yeo, and S.Venugopal, “Market-Oriented Cloud Computing:Vision, Hype, and Reality for Delivering IT Services as Computing Utilities”, Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC-08), Sep. 2008
[12] W.Y. Lin, G.Y. Lin, and H.Y.Wei, “Dynamic Auction Mechanism for Cloud Resource Allocation”, 10th IEEEACM International Conference on Cluster Cloud and Grid Computing (2010)
[13] G.Wei, A.V. Vasilakos, Y.Zheng, and N.Xiong, “A game-theoretic method of fair resource allocation for cloud computing services”, The journal of supercomputing, Vol.54, No.2.
[14] Yazir, Y.O., Matthews, C., Farahbod, R., Neville, S., Guitouni, A., Ganti, S., and Coady, Y., “Dynamic Resource Allocation in Computing Clouds through Distributed Multiple Criteria Decision Analysis”, 2010 IEEE 3rd Internatiuonal Conference on Cloud Computing (CLOUD 2010), July 2010.
[15] B.Malet and P.Pietzuch, “Resource Allocation across Multiple Cloud Data Centres”, 8th International workshop on Middleware for Grids, Clouds and e-Science. (MGC'10), Nov. 2010.
[16] G.Leey, B.G.Chunz, and R.H.Katz, “Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud”, HotCloud '11 June. 2011.
[17] B. Rajkumar, B. Anton, and A. Jemal, “Energy efficient management of data center resources for computing: Vision, architectural elements and open challenges,” in International Conference on Parallel and Distributed Processing Techniques and Applications, Jul. 2010.
[18] M. Mazzucco, D. Dyachuk, and R. Deters, “Maximizing Cloud Providers’ Revenues via Energy Aware Allocation Policies,” in 2010 IEEE 3rd International Conference on Cloud Computing. IEEE, 2010.
[19] K.Mochizuki and S.Kuribayashi, “Evaluation of optimal resource allocation method for cloud computing environments with limited electric power capacity”, Proceeding of the 14-th International Conference on Network-Based Information Systems (NBiS-2011), Sep. 2011.

 ISBN: 978-1-61208-216-5

Guidelines for Increasing the Adoption of Cloud Computing within SMEs

Marius MARIAN
Department of Computers and Information Technology
University of Craiova
Craiova, Romania
marius.marian@cs.ucv.ro

Ileana HAMBURG
Institut Arbeit und Technik
Westfälische Hochschule
Gelsenkirchen, Germany
hamburg@iat.eu

***Abstract* — This document is part of a larger effort meant to define a set of guidelines useful for the fast adoption of cloud computing and social media technologies within small and medium enterprises (SMEs) in European Union. The topic under scrutiny is how SMEs should approach and what they should do when embracing these new technologies, and also what to know about their potential impact on the SMEs businesses.**

Keywords: SME, guidelines, cloud computing.

I. INTRODUCTION

Cloud computing is not a new technology, but rather a natural evolution of efficient using and combining several modern technologies. Computing power, data storage and internetworking resources have all been put into a novel context and consequently, transformed into services (either separately or taken together). The paradigm in cloud computing is based on an old commercial approach – on-demand pay per use – in which you better rent a service for a specific period of time instead of buying the support infrastructure (utilities included), building a solution and administering it all by yourself. The cloud service providers (CSPs) promise reliable and configurable resources, made available promptly to consumers with a minimum effort and involvement on their behalf.

Small and medium enterprises (SMEs) are – as everyone else is – interested in reducing costs, and remaining competitive. Also, green computing is getting momentum and SMEs are targeting the issue too. Cloud computing is able to offer solutions to these aspects obviously for a price; the pay per use approach encourages a responsible behavior and maximum efficiency for what concerns consumption of resources and energy. In order to decide whether to pay that price and go all the way through, SMEs need guidance and knowledge of what are the best practices when approaching cloud computing technologies. This paper is about a work in progress on this topic emphasizing the importance of cloud computing for SMEs and the necessity to provide SMEs decision makers with guidelines and best practices.

II. MOTIVATION

SMEs are socially and economically important, since they represent 99% of all enterprises in the EU, employ more than 90 million people, and contribute to entrepreneurship and innovation [1], [2]. In Germany alone, there are about 3.2 million SMEs, most of them regionally anchored. Significant international, social and economic changes like globalization, market competition, technological innovation, the European Union enlargement, and particularly, the last financial crises affect the situation of SMEs; they need innovative and sustainable approaches to survive and be competitive. But most of European SMEs have shortage of financial resources and of skilled staff, no sustainable ICT (Information and Communication Technologies) strategies, have difficulties with the management of missing knowledge, and a low transfer of knowledge to improve the effectiveness of their work tasks, have not enough knowledge of policies of communication and cooperation in research and production. SME staff is often frustrated of constantly missing out on critical internal information due to complicated existing collaboration tools requiring users a lot of work to search out information necessary to their daily work tasks and other needs [3], [4].

Last developments in cloud computing and a most structured approach to social media in the work place can change this situation. The managers can select employees to form individual teams for given business activities, the teams can work together with a greater efficiency, and employees can seek expert advice across departments, share, and download updated documents. The real-time collaboration supported by the new approach of cloud computing and social media enables individuals and teams to reduce the time previously wasted searching through inboxes or in file servers for important documents or content. Two studies carried out in Germany (within the European-funded projects ReadiSME – http://www.readisme.com, and NetKnowing 2.0 – http://www.netknowing.com) show that about 70% of SMEs use standard software what is an advantage because most services offered by CSPs are standard.

But in connection with Software as a Service (SaaS – see also section III) [5], the results of the studies show that 30% of ICT sector SMEs use SaaS, 75% of SMEs from other sectors did not have plans for using SaaS till the end of 2011.

Some causes that are often mentioned are that in many SMEs, particularly small ones, there is only one decision maker, there are security problems of outsourcing (85%), there is a lack of trust in what concerns the CSP (75%), there are concerns related to the integration of SaaS with the existing ICT in the company (30%), there is no support for large bandwidth Internet connectivity in the company

 ISBN: 978-1-61208-216-5

(81.2%), there are no precise rules in the company about the issue of social media and social networks.

European Network and Information Security Agency (ENISA) is also conducting a security risk assessment of cloud computing technologies aimed at giving advice to SME's on the most important risks in adopting cloud computing technologies, as well as ways to address those risks. The timeframe of the survey was prolonged from 2010 to 2012 and will investigate in deep the actual needs, requirements and expectations of the SMEs for cloud computing services. Up to now the ENISA survey [1] (published and updated regularly) focused on topics such as the driving forces towards adopting the cloud, the size and the geo-location of the company, the cloud models, types, and services of potential interest, the possible use of multiple CSPs, the recovery options in case of disasters and incidents, and obviously the main concerns facing such a paradigm shift.

III. CLOUD COMPUTING SERVICES FOR SMES

In literature, there are clearly delimited three main classes of cloud computing services. Additionally, there exist other newly-defined classes that appeared as variants or reinterpretations of the main classes. Therefore, we have Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS). All these services are available remotely via some communication channel and will require a payment for use (even though there are some free services available, especially in what concerns e-mail and social media web-based applications).

The first service made available by most CSPs was IaaS. In practice, it is a complete virtual machine running a specific operating system. For SMEs the suspicion regarding the multi-tenancy/sharing of resources is alleviated since the level of control and the possibility to define perimeters of resources among tenants is easier to achieve with this class of service. As said, the fundamental unit in IaaS is the virtual machine that is in most cases a server. A CSP may provide depending on the specific business of a customer four subclasses of IaaS: *vendor-managed private cloud*, *dedicated hosting*, *hybrid hosting*, and *cloud hosting*.

In a vendor-managed private cloud, the client rents a number of physical servers placed in the same area of the data center so that they are as separate as possible with other hardware and internetworking. This IaaS configuration is the most expensive but also it is considered the most secure. The flexibility and scalability of the solution are poor therefore an SME should be able to estimate and know in advance the needed infrastructure. All changes up the scale are slow and require interaction and timely scheduling with the CSP. This scenario works best for large enterprises building their own data centers and not for SMEs.

Dedicated hosting is for clients requiring one or more physical servers anywhere within a data center, available on-demand. In this service configuration, even though the hardware and internetworking is mixed with other servers from the data center, a particular SME will not share its rented servers with any other tenant in the CSP cloud. This is less expensive than the previous configuration and is both scalable and flexible as long as the CSP provisioning of resources is well handled for the peak periods.

An intermediate service configuration between the two above is hybrid hosting. With it, a client pays for a mix of costly physical servers (they may be occasionally required to be located in the same perimeter within the CSP's data center) and some inexpensive virtual server instances. The sensitive data and the applications of the SME run on the physical servers, while the rest of the data is stored in the virtual servers. The solution is flexible and dynamically scalable when it comes to renting more virtual server instances during peak hours. Physical servers may be rented but only if the customer accepted them to be anywhere within the CSP's data center.

The last IaaS configuration is what everyone expected from the cloud, technical and environmental efficiency. Lots of virtual server instances available on-demand with a high degree of scalability and flexibility in use, and at a very low price. The reverse of the medal is that a customer shares all the hardware and internetworking resources with the other tenants. A security and privacy perimeter can be achieved only at virtual server instance. It is the best commercial offer for SMEs and start-ups. SMEs may want to consider it since they do not have the capital and (perhaps, they are not willing to invest in) the know-how for the hardware-software infrastructure and management. Start-up companies find themselves in a happy scenario with the cloud since this could be a perfect business incubator at a very low initial investment cost.

PaaS is the second class of services in which SMEs may acquire only the specific platform they need. It is an extension of the IaaS to accommodate the middleware and to improve the performance in using it. It may be for example a web development platform containing the web/application server, the integrated development environment, the associated database and all additional utilities for development and testing. The tenants are sharing a large part of the middleware, and the CSPs can no longer distinguish some clear perimeters among them. Problems typically appear when the middleware is not as robust as expected or the shared databases are not well configured. The downside of this happening is that one customer may influence negatively the quality of the PaaS observed by the others.

Many European-based SMEs and start-up companies in the field of IT development and research may be interested in renting such highly customized platforms at acceptable low costs. Highly interested could be for example start-up firms working to deliver mobile applications for the extremely crowded market of smart mobile devices. They could then use this cloud service and produce and eventually sell their own on-line SaaS applications. Careful attention should be paid by the SMEs to service level agreements (SLA), to protection mechanisms enabled by the CSP for its tenants, and to business continuity (BCP) and disaster recovery plans (DRP).

SaaS is the third main class of services, and with it CSPs offer SMEs the possibility of acquiring on-demand usage-time for different types of software services. This includes a wide range of applications: office tools, graphic utilities, data

 ISBN: 978-1-61208-216-5

storage facilities, etc. SaaS is dynamically scalable, device independent (giving no access to the hardware for the tenant) and most of the applications are collaborative, allowing thus multiple users to share documents and work on them concurrently. Adding social media services through SaaS can only enhance this collaboration. The most common problems with this type of service are generated by the authentication mechanism, the management of authentication credentials by end users, the access control, the lack of securely tunneled communications, or intrinsic faults with the web applications used.

A. Advantages

They can be summarized in remote accessibility, flexibility, scalability, security, and environmentally friendly. Flexibility and scalability means that SMEs will only pay and use the resources they need and for the time they need them. CSP promise that provisioning and de-provisioning of resources will be transparent and easy to handle. Then, accessibility means that the business of the SME is no longer restricted to a particular location. Actually, for certain areas of business this is even more beneficial for the employees could work remotely and thus telecommuting and contributing to green computing. Most internal company servers use only approximately 30% of their capacity while in a large cloud data center the percentage of utilization goes up to circa 80% [8]. On a European scale this means that energy and consequently, carbon footprint reductions can be made. Furthermore, recent research in the field of microprocessors (e.g. Intel Atom, AMD Geode, VIA C7, etc.) proves that cloud computing users may use lower-power computers that are performing sufficiently to run cloud applications, thus cutting down the electricity bills and individual carbon emission footprints. Security as strange as it may sound has now a better chance to be well implemented (right from the start) than in any previous computing approach. In fact, Security as a Service is gaining momentum since it may represent a worldwide implementation of security standards, frameworks and regulations that will eventually minimize the existing security implementation differences or absence.

B. Concerns and challenges

First concern for SMEs is raised by the multi-tenancy property of the cloud. As we have seen, there are ways to counter this concern by acquiring only particular configurations of IaaS. Then, there is the performance and quality of the services offered by the CSP. On this issue, SMEs adopting the cloud into their business should carefully elaborate on the SLAs signed with the CSP. Another security concern is that users with administrative privileges on the side of the CSP might take an unauthorized look at their data. Procedures, frameworks, agreements and audits may facilitate a reasonable level of trust between SMEs and CSP. Associated with this last point are also de-provisioning of data and the way in which data are handled when SMEs are leaving the cloud plus the data geo-location. Data geo-location might create legal and compliance problems for SMEs when the CSP is not clear about where they have their facilities. A *Buy European* approach would settle in a positive manner this issue in tandem with a stimulus offered to CSPs to have their installations on EU territory only [9].

IV. CLOUD ADOPTION RECOMMENDATIONS FOR SMES

In what follows we will try to delineate some of the major areas of interest for SMEs when approaching the cloud. This set of cloud adoption guidelines that will be devised for European SMEs are based on the Security Guidance of CSA (Cloud Security Alliance) [10], ENISA cloud analyses, and Jericho Forum commandments [11], and will further elaborate on other areas of interest for SMEs. The cited documents are broad and thorough analyses of the subject. We believe that SMEs in particular would be better supported and encouraged to take advantage of the cloud if there were some specific documents containing the best practices and the guidelines for adopting the cloud into their business. Furthermore, standards, frameworks, benchmarks and regulations at EU level would help refining these guidelines and perhaps, they will also benefit from the ideas contained in these guidelines. CSA contributions in what concerns cloud security and privacy (Trusted Cloud initiative, Cloud Control Matrix, and Certification of Cloud Security Knowledge) are giving hints that this is the direction to be followed.

European SMEs are small in number of people employed (up to a maximum of 250 persons). It goes without saying that they would primarily invest their capital in improving their business process (production, services, etc.) and they do not always have the know-how to manage in-house the IT support infrastructure. In fact, ENISA found out that European SMEs are interested first of all in avoiding capital expenditure in hardware, software, IT support, information security (68.1% of the respondents). On the second place were the scalability and flexibility of required IT resources (63.9%), and on the third position were business continuity and disaster recovery capabilities (52.8%). ENISA survey proved that the highest percentage of SMEs willing to move into

The SMEs decision makers must understand well and fast what differences exist among different cloud computing solutions available on the market, what their costs are, what the security and privacy impacts are, and how their availability and acquisition may add value to their particular business. Decision factors must also decide what really matters for the SME business from a data security and privacy stand point and if there are any guarantees from the CSP to ensure data security and privacy (if possible, cryptography should be ubiquitous in the cloud or negotiated when performance reasons demand so, such that all data at rest or in transit be encrypted). This preliminary analysis must be performed just before initiating any other step. A related questionnaire for the decision makers would greatly simplify putting the things in context and providing some quick analytical results.

Certifications and benchmarking of the various CSPs would also be helpful. CSP transparency and openness for external auditing of their internal processes is also a sign of trust and a great control mechanism for SMEs. Auditing

 ISBN: 978-1-61208-216-5

preserves the level of trust of customers, and SMEs should investigate negotiate and pay attention to the terms agreed in contracts and SLAs concerning audit, monitoring, event log reviews, physical inspection of the CSP facilities, etc.

Special attention will be paid within the guidelines also to the legal and compliance implications of moving into the cloud for the SMEs. Proposals must be drafted towards a set of common practices to be followed when signing contracts and accepting SLAs. Awareness and dissemination instruments (social media) will be used to publish and bring into discussion the findings and real-case SME cloud migration scenarios. Cloud migration must be investigated not only at the first adoption of the paradigm, but also for cases when an SME decides to switch and move from one CSP to another. This investigation has legal, financial (on short and long terms) and technical implications related to deleting the data from the former cloud. Third trusted party audits and confidentiality agreements must be enforced.

ENISA found out that SMEs are mainly concerned with confidentiality of the corporate data, privacy, integrity and availability of services and data. It is important that SMEs rest assured by the CSP that their data will be private, available, and untouched. SMEs will have to answer themselves if they are ready to plan and enforce business continuity in cooperation with the CSP. Incident response and disaster recovery are related topics for which both CSPs and SMEs will have to agree upon and put in practice. A collection of best practices related to these topics will be shared among SMEs.

V. CONCLUSIONS AND FUTURE WORK

It is expected that the European SMEs will lead the global economic trend of adopting cloud computing paradigm within their daily businesses. EU Commission should further investigate through ENISA the need for a clear legislation in the field of cloud computing as a public utility of the following decade. Thus, a European Network of Clouds can be built upon, and also, participating third party CSPs could certify their services on various levels of compliance with that EU cloud legislation. This would encourage SMEs since trust is the base for economic development and creating new opportunities.

We strongly believe that the cloud adoption by SMEs could be further accelerated by establishing at least a set of guidelines including some recommendations and a book of good practices at European level. In our research, we have not found anything similar so far.

Our next effort is thus aimed to developing a short practical guide for using cloud computing and social media within European SMEs and to discuss these guidelines by conducting focused interviews with all the partners from the NetKnowing 2.0 project. A second step will be to disseminate these guidelines at European level also by using the social media-based platform developed within the same project and to organize moderated forums for discussing (and further improving) the guidelines and other issues concerning this topic. Last step is trying to apply the results and findings in SMEs from project partner countries and to identify specific areas of organizational improvements both within European-based CSPs and SMEs consuming the cloud-based services.

ACKNOWLEDGMENT

This work was supported by the strategic grant POSDRU/89/1.5/S/61968, Project ID61968 (2009), co-financed by the European Social Fund within the Sectorial Operational Program Human Resources Development 2007 – 2013.

The studies have been carried out within the innovation-transfer projects ReadiSME and NetKnowing 2.0 supported by the EU LLP Leonardo da Vinci sub-programme.

REFERENCES

[1] I. Hamburg, "Supporting cross-border knowledge transfer through virtual teams, communities and ICT tools", in Robert J. Howlett (ed.) "Innovation through knowledge transfer", Springer, 2011, Berlin, Germany, pp. 23 – 29.

[2] G. Attwell, D. Dirckinck-Holmfeld, P. Fabian, A. Kárpáti, P. Littig, "E-Learning", in "e-Learning in Europe – Results and Recommendations", Thematic Monitoring under the European Union Leonardo da Vinci Programme, 2003, Bonn, Germany.

[3] T. Hall, I. Hamburg, "Readiness for knowledge management, methods und environments for innovation", in Emma O'Brien, Seamus Clifford, Mark Southern, (eds.): "Knowledge management for process, organizational and marketing innovation: tools and methods", Hershey Information Science Reference, 2011, pp. 1 – 15.

[4] I. Hamburg, "eLearning 2.0 and social, practice-oriented communities to improve knowledge in companies", in Ortiz Bellot, G., Sasaki, H., Ehmann, M. & Dini, C. (eds.), Proceedings of The Fifth International Conference on Internet and Web Applications and Services (ICIW 2010), May 9 – 15, 2010, Barcelona, Spain, pp. 411 – 416.

[5] T. Haselmann, G. Vossen, "Software-as-a-Service in Small and Medium Enterprises: An Empirical Attitude Assessment", Proceedings of the 12th International Conference on Web Information Systems Engineering (WISE 2011), Springer, October 12 – 14, 2011, Sydney, Australia.

[6] ENISA, "An SME Perspective on Cloud Computing – A Survey," November 2009, available on-line at http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-sme-survey.

[7] R. Harms, M. Yamartino, "The economics of the cloud for the EU public sector", Microsoft white paper, November 2010, available at http://www.microsoft.eu/Portals /0/Document/EU_Public_Sector_Cloud_Economics_A4.pdf .

[8] J. Stanley, K.G. Brill, J. Koomey, „Four Metrics Define Data Center Greennes", Uptime Institute white paper, 2007, available at http://uptimeinstitute.org/wp pdf/(TUI3009F) FourMetricsDefineDataCenter.pdf .

[9] F. Etro, „The Economic Impact of Cloud Computing on Business Creation, Employment and Output in Europe", International Think-Tank on Innovation and Competition (INTERTIC), Review of Business and Economics, 2009, available at http://www.intertic.org/Policy Papers/CC.pdf

[10] Cloud Security Alliance, „Security Guidance for Critical Areas of Focus in Cloud Computing", versiunea 3.0, 2011, available at https://cloudsecurityalliance.org/guidance/ csaguide.v3.0.pdf

[11] Jericho Forum, „Jericho Forum Commandments", available at http://www.opengroup.org/jericho/commandments_v1.2.pdf

IDSaaS: Intrusion Detection System as a Service in Public Clouds

Turki Alharkan
School of Computing
Queen's University
Kingston, ON Canada
alharkan@cs.queensu.ca

Patrick Martin
School of Computing
Queen's University
Kingston, ON Canada
martin@cs.queensu.ca

***Abstract* - In a public cloud computing environment, consumers cannot always just depend on the cloud provider's security infrastructure. They may need to monitor and protect their virtual existence by implementing their own intrusion detection capabilities along with other security technologies within the cloud fabric. Intrusion Detection as a Service (IDSaaS) targets security of the infrastructure level for a public cloud (IaaS) by providing intrusion detection technology that is highly elastic, portable and fully controlled by the cloud consumer. A prototype of IDSaaS is described.**

Keywords-Security; Cloud Computing; Intrusion Detection System

I. INTRODUCTION

As the number of cyber attacks against social networks and large internet enterprises continues to rise, organizations are questioning the safety of moving their computational assets toward the cloud [1]. Traditional network security measurements face new challenges in the cloud such as virtual machine intrusion attacks and malicious user activities. New security measures are therefore needed to increase users' level of trust in clouds. Currently, cloud providers enforce data encryption for the storage containers, virtual firewalls and access control lists [2]. However, cloud consumers need to develop secure and customizable solutions to comply with their application requirements. For example, an attack classified as SQL injection with the ability to control the host operating system targeting the business application may wish to impose a combination of application and system level policies [3]. The current security mechanisms from the cloud providers are not intended to enforce this level of constraints so additional measurements are required.

In this paper, we propose the Intrusion Detection System as a Service (IDSaaS) framework, which is a network and signature based IDS for the cloud model. In particular, IDSaaS is an on-demand, portable, controllable by the cloud consumer and available through the pay-per-use cost model. IDSaaS mainly targeting the IaaS level of the cloud. However, other levels of the cloud can be monitored such as the SaaS level. Therefore, the IDSaaS primary task is to monitor and log suspicious network activities between virtual machines within a pre-defined virtual network in public clouds. A proof-of-concept prototype for the Amazon EC2 cloud [4] is presented.

The major contribution for this work is a scalable and customizable cloud-based service that provides cloud consumers with IDS capabilities regardless of the cloud model. IDSaaS administrators have the abilities to monitor and react to attacks on multiple VMs residing within a consumer's Virtual Private Cloud (VPC) [5], and to identify specific attacking scenarios based on their application needs. Moreover, the system can adapt its performance to the traffic load by activating the on-demand elasticity feature. For example, the number of the available IDS Core components can change based on the amount of traffic targeting the protected business application. Furthermore, IDSaaS components can be scaled to protect virtual machines residing in different cloud regions. These features are designed with the consideration of the cloud environment.

The rest of this paper is organized as follows: Section II describes related work. Section III introduces the concept of IDSaaS and outlines its main features. Section IV reviews the IDSaaS main components and tools. A proof-of-concept prototype implementation of IDSaaS in Amazon's EC2 public cloud is presented in Section V. Section VI presents a sample attack scenario and evaluates the operation of the prototype IDSaaS. Finally, Section VII summarizes the paper and discusses future work.

II. RELATED WORK

The introduction of IDS in the cloud is the focus of several research projects. Each of these projects, however, targets different service models of the cloud or pursues a different goal. IDSaaS is intended to fill the gap in this research area.

The Intrusion Detection based on Cloud Computing (IDCC) architecture [6] was developed to achieve a global monitoring view of the network resources and to help in discovering coordinated attacks on local sites. This architecture consists of two major parts, the local sites and the global site. The purpose of the global site is to collects the alerts generated by the local sites. When a threat is detected by the global site, the particular local site security administrator is informed so a proper action can be taken such as blacklisting the source of the attack. This architecture is more suitable for private clouds that are designed with the needed infrastructure to allow global and local site nodes to be communicated privately. As a result, cloud users at the local sites are more dependent on the cloud provider's global IDS administration. Furthermore, the process of

 ISBN: 978-1-61208-216-5

administrating the global and local sites raises some serous security challenges.

The work by Mazzariello et al. [7] discusses various deployments of existing IDS to an open source cloud environment. The suggested model is to deploy multiple IDSs next to every cloud physical controller, which monitors a smaller portion of network traffic for a set of virtual machines. The general setup for this approach requires deep alteration of the physical implementation of the cloud assets, which results in a strong dependency between the IDS components and the cloud provider's infrastructure. Consequently, the IDS administration process available to the cloud consumers is limited and lacks customization.

The authors of Intrusion Detection In the cloud (IDC) [8] introduce the concept of a partial IDS management for the cloud users. The proposed architecture consists of several sensors and a central management unit. This distributed-IDS architecture is implemented in all of the three cloud computing layers (Application layer, Platform layer and System layer), which includes a combination of host-based IDS (HIDS) and network-based IDS (NIDS) sensors. HIDS is incorporated with every VM initialized by the user. On the other hand, NIDS sensors are placed in each cloud layer to monitor the management module of that layer. In the central IDS management unit, alerts can be correlated and analyzed from different sensors in different layers. Furthermore, cloud users can configure which rules to use from the existing rule-set based on their application needs. One of the main issues with the IDC approach is the strong dependency between the cloud users and the cloud provider substructure. The cloud provider has to implement the main components of the intrusion detection environment like the central management unit, the integrated HIDS for each VM host, signature databases, and the communication channels between VMs and the IDS management unit. Cloud users are totally dependent on the provider's IDS infrastructure but they still partially control the IDS management unit with limited functionality. Moreover, there are serious privacy concerns arising from integrating IDS components on every customer virtual machine that is installed by the cloud provider.

Much of the proposed academic research on IDSs in the cloud has focused on providing intrusion detection mechanisms for specific security problems. The Autonomic Violation Prevention System (AVPS) [9] concentrates on self-protection against security policy violations generated by privileged users. This goal is achieved by defining the system's access policies and continuously monitoring the internal traffic for any violations of these policies. The authors of the AVPS framework suggest their system can be deployed to virtual network environments like the cloud. However, AVPS is not evaluated against many cloud features. For example, scaling the system for multiple core IDS nodes is needed to bear the increase in the traffic due to heavy application requests. Moreover, the need to support the distributed nature of the cloud by protecting multiple applications in different cloud locations is absent. Additionally, the work in [10] introduces a maneuvering tactic to confront the denial of service attack on the cloud by moving the attacked virtual machine to a safe datacenter.

Our main aim is to provide a general defense strategy by protecting different levels of the cloud, and incorporating tailored signatures for various security threats. IDSaaS is intended to work in different cloud models and to provide flexible user control of security.

III. IDSAAS IN THE CLOUD

A. Overview

Cloud consumers should not have to only depend on the cloud provider's security infrastructure. They need to be able to monitor and protect their virtual existence by enforcing additional security methods with other network security technologies like firewalls, access control lists and data encryption within the cloud fabric. Consequently, cloud consumers require the capability to deploy IDSs within their virtual boundaries.

IDSaaS, which is shown in Figure 1, assists cloud consumers with securing their virtual machines by deploying an intrusion detection system in public clouds. It protects them against attacks initiated from any external source over the internet in addition to those originating from inside the cloud. Here, cloud consumers implement the applications they want to protect in the form of Virtual Machine Instances (VMI) within a secure virtual network (V-LAN). Concurrently, IDSaaS components can be placed in the same V-LAN to guard these valuable assets.

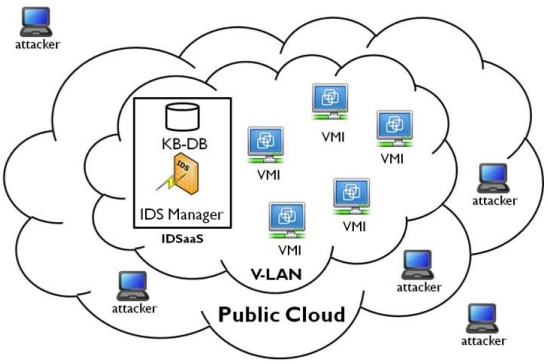


Figure 1. IDSaaS in the Cloud

B. IDSaaS Features

IDSaaS provides the following features to cloud consumers:

- *On-demand Elasticity:* Cloud consumers have the ability to scale IDSaaS core components that are responsible for discovering suspicious traffic based on the traffic volume for the protected business application.
- *Portability:* The IDSaaS model is implemented as a collection of Virtual Machine (VM) instances based on Xen virtualization [11]. Therefore, IDSaaS components can reside in public or private clouds or even in multiple regions within a single cloud.

 ISBN: 978-1-61208-216-5

- *Full-Control:* IDSaaS management, functionality and architecture are independent from the cloud provider. For example, an IDSaaS administrator can deactivate a particular IDS core node or enable a specific threat signature definition.
- *Customizable Signatures:* IDSaaS is equipped with predefined threat scenarios for faster and more accurate detection rates. These scenarios are represented in the form of rules. In addition, IDSaaS users can write customized signatures based on the nature of the defended application. These rules can protect the application (SaaS), the system (PaaS), and the network (IaaS) levels of the cloud model. The grammar and examples of threat signatures are given in Table I.

TABLE I. Signature Examples

	Signature
Grammar	*[action] [protocol] [src_ip] [src_port] [direction] [dist_ip] [dist_port] [option]*
Application Level	alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (uricontent:".php";pcre:"/(\%27)|(\')|(\-\-)|(%23)|(#)/"; msg:"SQL Injection Attack"; sid:1000005;rev:1;)
System Level	activate tcp any any -> $HOME_NET 22 (content:"/bin/sh"; activates:1; msg:"Possible SSH buffer overflow"; sid:1000023;) dynamic tcp any any -> HOME_NET 22 (activated_by:1; count:10;)
Network Level	drop tcp $EXTERNAL_NET any -> $DB_NET 3306 (msg:"Unauthorized Access to DB server"; content:"mysql -p"; nocase; sid:1000019;)

- *Reliability:* IDSaaS has the ability to backup the collected alerts with system configuration files and store them in an off-cloud location. This facilitates an efficient system recovery in the case of failure.

IV. IDSAAS ARCHITECTURE

IDSaaS, as shown in Figure 2, consists of five main components: the Intrusion Engine, the Output Processor, the Events Database, the Alerts Management, and the Rule-set Manager.

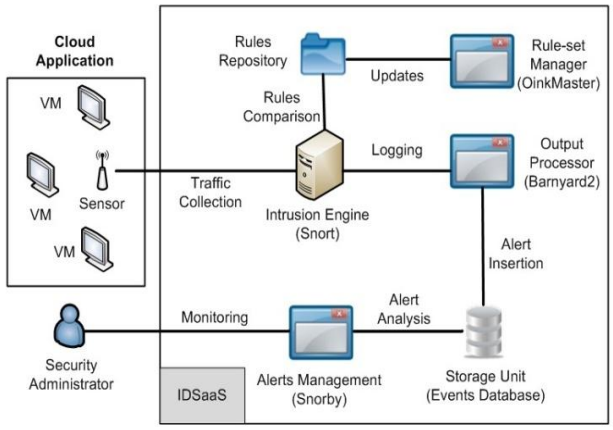


Figure 2. IDSaaS Components

A. Intrusion Engine:

Initially, the sensor taps into the network and collects network packets, which are decoded for the analysis step. The Intrusion Engine is the brain of the system. It preprocesses the incoming packets and examines their payload section looking for any matching pattern of a threat defined in the loaded attacking rules. The processed packet is logged only if it matches a rule. The output binary file is a collection of captured alerts. The signature-based detection model is selected because of its suitability to the cloud environment. Simplicity, flexibility and ease of sharing signatures are some of the advantages of this approach. Also, it will enforce the elasticity feature by eliminating the learning time for the system's behavior required for the anomaly-based approach.

B. Output Processor

The main purpose of the Output Processor is to increase the performance of the intrusion engine by formatting the output log files and inserting them into the Events Database. This allows the intrusion engine to focus on processing network packets and logging alerts while leaving the relatively slow process of database insertion to the Output Processor component.

C. Events Database

The Events Database stores the formatted events generated from the Output Processer component. Also, the database stores other relative information like sensor ID, event timestamp and packet payload details.

D. Alert Management

The Alert Management component is used as a GUI tool to view the generated alerts and correlate them. It allows the security administrator to extract events and relate them to predefined attacking situations. Moreover, it provides the ability to generate reports based on time, source of the attack, or types of threat.

E. Rule-set Manager

IDSaaS is a rule-based IDS system, and its rule base has to be frequently updated to reflect the new threats and attacking scenarios. The Rule-set Manager automatically downloads the most up-to-date set of rules from multiple locations. Rules are generally obtained either for free from the public community service or through a subscription service such as the SourceFire VRT [12].

V. PROOF-OF-CONCEPT IDSAAS

A proof-of-concept prototype of IDSaaS is implemented in Amazon web services using the EC2 cloud. Although it is tested on a public cloud, the IDSaaS framework can be applied to other types of clouds that support V-LAN implementation. All IDSaaS components are constructed and bundled in the form of Amazon Machine Images (AMI). The on-demand elasticity feature of IDSaaS can therefore be enforced by starting the AMI instances on the fly.

 ISBN: 978-1-61208-216-5

A. Tools

The prototype of IDSaaS is built using a collection of open source tools. Snort [13] is an open source network-based intrusion detection system that is used in the Intrusion Engine component. As for the Output Process component, Barnyard2 [14] is used to act as the middle tier between the Intrusion Engine and the Event Database. MySql is used as the relational database to store the generated alerts. Snorby [15] is used as a graphical interface for the system to display various information and statistics about the collected incidents. The Rule-set Manager is built using Oinkmaster [16], which is a simple Perl script that compares locally stored rules with the shared communities' rules repository and downloads updated rules based on user preferences.

B. Network Environment

The IDSaaS utilizes the Virtual Private Cloud (VPC) service from Amazon. This V-LAN setup has the advantage of creating a private network area that can only be controlled by the application owner within the public cloud borders. In the VPC space, both private and public subnets were created. The private subnet maintains the protected business application VMs. Any virtual machine that is placed in the private subnet is isolated from the cloud traffic except the traffic traveling from or to the public subnet of the VPC. The public subnet hosts various IDSaaS VMs. Figure 3 illustrates the general layout of the IDSaaS in the Amazon VPC.

C. IDSaaS VMs

1) IDSaaS Manager

The Manager VM is the security administrator access point where various supervision tasks can be performed. For instance, it hosts the Alert Management component that monitors traffic for any suspicious activity in the VPC. The Event Database also resides in the Manager VM. The Manager VM can be used as an access point to configure other VMs in both public and private subnets.

2) IDS Core

The IDS Core VM is the gatekeeper to the business application VMs in the private subnet. It inspects all incoming traffic using the Intrusion Engine component. Based on the threat rule matching process, a request to the business application VMs can be allowed or trapped by the IDS Core VM. As a result, the Output Processor will send generated alerts to the Event Database.

D. Security Groups (SG)

Security Groups are used to define permissible network services that can run on each VM in the Amazon EC2 cloud. These virtual firewalls can decide the nature of the traffic permitted for each VM in the form of inbound and outbound allowable ports. Any VM that is attached to a particular SG will comply with the services defined for that SG.

VI. IDSaaS Evaluation

We conducted several experiments to evaluate the effectiveness of our proof-of-concept prototype of IDSaaS in EC2. We first present an attacking scenario and then show the results of the experiments.

A. Attacking Scenario

In the scenario, a business application that consists of web and database servers are placed into a private subnet of the Amazon EC2 cloud in order to be accessed by the end users via the IDS Core VM. On the other hand, the IDSaaS VMs are placed on the public subnet. Figure 3 demonstrates the network setup and the deployment of IDSaaS components.

The business application can be accessed using the exposed URL or IP address assigned to the IDS Core VM. Experiments were conducted with different IDSaaS network setups. Each setup experienced two attacking locations; an External Attacker located outside the cloud and an Internal Attacker located inside the Amazon EC2 Cloud. Each attacker used two TCP protocols to attack the victim system. First, they issued a series of HTTP requests to access the registered users' information page of the business application. This area is restricted to the application administrator, so an alert is released for non-authorized access to this area. Second, they used the FTP protocol to upload a suspicious file to the target server through the file transfer service of the business application. Customized rules were enabled in the IDSaaS to capture such a harmful activity for each attacking type.

B. IDSaaS Components Overhead Experiment

The effectiveness of IDSaaS was evaluated by measuring the overhead added by the different IDSaaS components while protecting business applications in a public cloud. By providing an extra level of protection, IDSaaS improves the security element of the virtual machines on the Amazon cloud. Our results so far indicate acceptable increases in the response time for the business application after adding the IDSaaS components (Figure 4). For example, in the case of the FTP requests, IDSaaS imposes 10.60% and 9.27% increases in response time for traffic originating from outside and inside the cloud, respectively. Similarly, for HTTP requests, it imposes increases of 8.58% and 3.57% for traffic originating from outside and inside the cloud, respectively. We believe this size of increase in response time is justifiable given the additional ability to enforce tailor-made attacking rules. Table II shows the average response time for all network setups of the experiment.

 ISBN: 978-1-61208-216-5

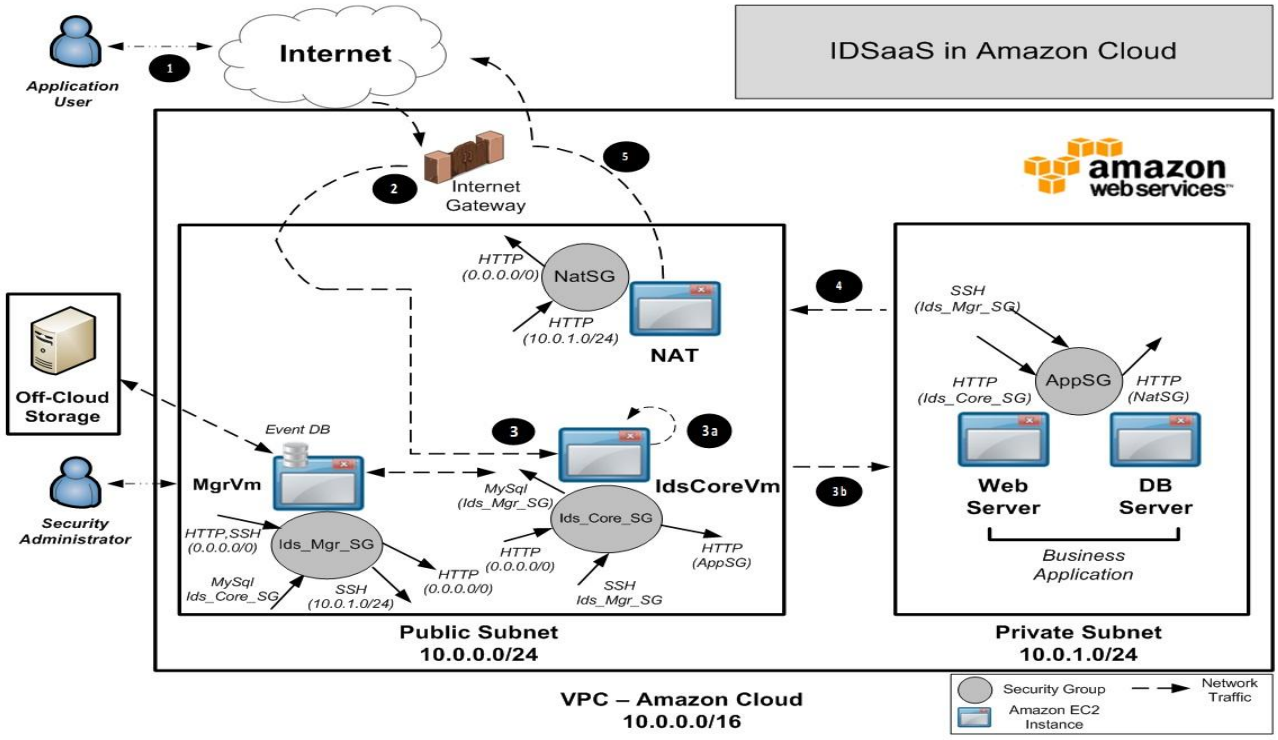


Figure 3. IDSaaS in Amazon Cloud

TABLE II. Components Overhead Experiment results

Network Setup		External Location	Internal Location
Base Network (No VPC, No IDSaaS)	HTTP Request	0.303 sec	0.224 sec
	FTP Request	16.731 sec	3.969 sec
VPC Network (No IDSaaS)	HTTP Request	0.310 sec	0.230 sec
	FTP Request	17.040 sec	4.098 sec
IDSaaS Network (VPC, IDSaaS)	HTTP Rule	0.329 sec	0.232 sec
	FTP Rule	18.505 sec	4.337 sec

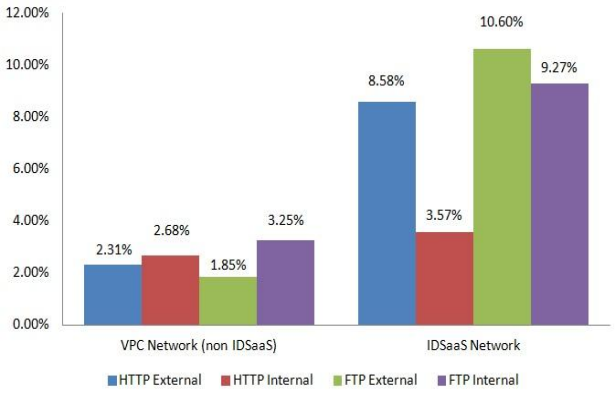


Figure 4. IDSaaS Components Overhead

C. IDSaaS Rules Overhead Experiment

The number of loaded attacking rules can also affect the efficiency of the IDSaaS in capturing many threats. In this experiment, we observed the performance of the intrusion engine (IDS Core VM) against different rule set stages. Stage one includes a complete set of rules (18,833 rules) addressing different attacking situations. This rule-set contains a collection of the intrusion engine's preinstalled rules as well as rules obtained from the public communities like Emerging Threats team [17]. Stage two decreases the rule set to 11 rules, which represents the Attack-Response (A-R) rules. Finally, the last stage incorporates a single rule to detect the Automatic Directory Listing (ADL) attack.

The IDS Core VM is used to compare the rules from the rule repository with the captured traffic in the form of a pcap file [18]. Intrusion engine performance was defined as the run time to process incoming packets, compare them with enabled rules and produce alerts in the form of binary logs. Therefore, the smaller the run time to analyze traffic packets, the better the performance of the intrusion engine.

Figure 5. IDSaaS Rule Overhead Experiment Results

Figure 5 shows that intrusion engine produced a single alert in an average time of 24.77 seconds by enabling the ADL rule (Stage 3). On the other hand, the intrusion engine took an average of 28.46 seconds to discover 68 threats by enabling the A-R subset. As a result, there was a 14.90% increase in the overhead for the extra rules enabled between stage 3 and stage 2. Similarly, stage 1 managed to capture 10,504 threats from the data sample within an average time of 48.72 seconds. This can be translated into an increase of 71.19% of the overhead compared to stage 2. For that reason, enabling all rules will degrade the performance of the intrusion engine and it will increase the percentage of false-positive alerts. Hence, fine-tuning the intrusion engine component to reflect the nature of the protected application is an important step when dealing with large number of rules.

D. Distributed IDSaaS Experiment

We implemented a distributed version of IDSaaS (D-IDSaaS) that has the ability to protect application VMs residing in multiple cloud regions. This is achieved by placing one or more IDS Core VMs in the same VPC as the business application VMs and placing the Manager VM in a centralized location. The security administrator can therefore monitor multiple business applications in different regions of the cloud from the central Manager VM.

We examined the cost of sending alerts from the IDS Core VM to the Manager component in three network configurations. Configuration 1, places the IDS Core VM and the Manager VM in the same VPC of the same cloud region. This typical IDSaaS setup is illustrated previously in Figure 3. Configuration 2, positions the Manager VM in a corporate network outside the cloud to meet with the privacy concerns of storing alerts in the cloud as well as reducing the storage costs of archiving historical alerts. In configuration 3, the IDS Core VM and the Manager VM are placed in different regions of the cloud. The business applications and the IDS Core VM are placed in the EU region of Amazon cloud and the Manager VM is positioned in the US East region of the Amazon cloud. Figure 6 displays the network layout for configuration 2 and 3.

The intrusion engine component was configured to read from a single pre-captured traffic file rather than from live network traffic. This standardized the input traffic to be analyzed by the intrusion engine for all network layouts. The used pcap file contained 30,000 network packets, which generates 145 alerts when enabling all installed rules (18,833 rules). Both the IDS Core VM and Manager VM were initialized using the small EC2 instances (OS Ubuntu, 1.7 GB memory, 1 virtual core CPU and 160 GB storage). However, the Manager VM in the off-cloud network (OS Ubuntu, 1.7 GB memory, 1 virtual CPU, 20 GB storage) was initialized using the VMware software [19] as a guest operating system.

The average dispatching time for 145 alerts from the pcap file using 100 trails was 2.35 seconds in configuration 1. In configuration 2, the same number of alerts was received on an average of 30.94 seconds. However, the highest dispatching time was obtained from configuration 3 with 119.70 seconds. The results are demonstrated in Figure 7. We believe this high value is due to alerts transmission time between the two Amazon regions.

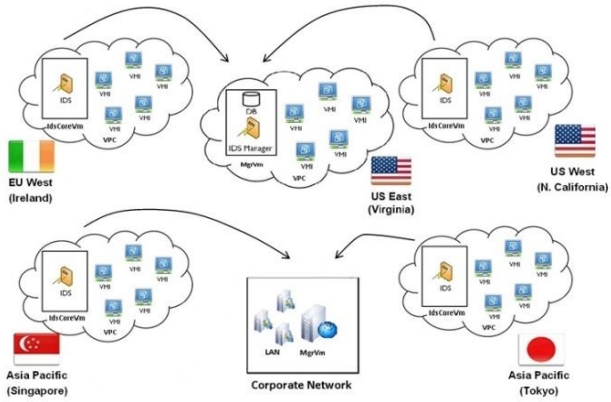


Figure 6. Distributed IDSaaS in Public Cloud

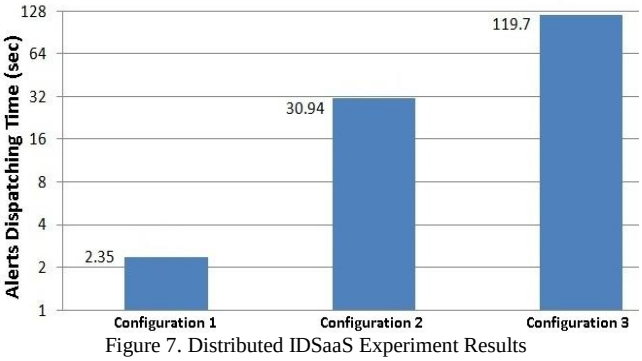


Figure 7. Distributed IDSaaS Experiment Results

VII. SUMMARY

In this paper, we introduced IDSaaS, which is a framework that enables consumers to protect their virtual

machines in public clouds. IDSaaS is compatible with many cloud features, such as portability, elasticity, on demand requests and pay-per-use service. The approach presented in this paper is implemented as a collection of virtual machines in order to comply with the cloud model.

Cloud consumers need to have customizable and controllable security solutions in the clouds. The major contribution for this work is a service to provide them with IDS capabilities regardless of the cloud model. With IDSaaS, users can define a virtual private area within the cloud space for their applications that can be secured with application-specific policies. Therefore, IDSaaS adds new levels of security onto those already supplied by cloud providers.

Increasing system availability is a feature to be implemented for future IDSaaS system. A replica of the IDS Core VM can be created to distribute the traffic load to prevent single point of failure situations. Therefore, a virtual load balancer node can increase the accessibility of the IDSaaS components in the cloud. Also, it can be responsible for balancing the traffic load between multiple IDS Core VMs.

ACKNOWLEDGMENT

This research is supported by the Ministry of Higher Education in the Kingdom of Saudi Arabia, the Saudi Arabian Cultural Bureau in Canada and Queen's University.

REFERENCES

[1] C. Burns, "Public cloud security remains MISSION IMPOSSIBLE," Network World. Oct, 2011. [Online]. Available: http://www.networkworld.com/supp/2011/enterprise5/101011-ecs-cloud-security-250973.html, [Retrieved: June, 2012].

[2] Amazon Web Services: Overview of Security Processes. [Online]. Available: http://aws.amazon.com/security, [Retrieved: June, 2012].

[3] B. Damele and A. Guimaraes, "Advanced SQL injection to operating system full control", Black Hat Europe 2009, April 2009

[4] Amazon Elastic Compute Cloud (Amazon EC2). [Online]. Available: http://aws.amazon.com/ec2/, [Retrieved: June, 2012].

[5] Amazon Virtual Private Cloud (Amazon VPC). [Online]. Available: http://aws.amazon.com/vpc, [Retrieved: June, 2012].

[6] W. Xin, H. Ting-lei, and L. Xiao-yu, "Research on the Intrusion detection mechanism based on cloud computing," Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference pp.125-128, 22-24 Oct. 2010

[7] C. Mazzariello, R. Bifulco, and R. Canonic, "Integrating a Network IDS into an Open Source Cloud Computing Environment," Sixth Internationl Conference on Information Assurance and Security (IAS), 2010

[8] S. Roschke, F. Cheng, and C. Meinel, "Intrusion Detection in the Cloud", In Proceedings of Workshop Security in Cloud Computing (SCC'09), IEEE Press, Chengdu, China, pp. 729-734 (December 2009).

[9] F. Sibai and D. Menasce, "Defeating the Insider Threat via Autonomic Network Capabilities," Communication Systems and Networks (COMSNETS), 2011 Third International Conference pp. 1-10, 4-8 Jan. 2011

[10] A. Bakshi and Y. Dujodwala, "Securing Cloud from DDOS Attacks Using Intrusion Detection System in Virtual Machine," ICCSN '10, pp. 260-264, 2010, IEEE Computer Society, USA, 2010

[11] Feature Guide: Amazon EC2 User Selectable Kernels. [Online]. Available: http://aws.amazon.com/articles/1345, [Retrieved: June, 2012].

[12] The official Snort Rule-set, Sourcefire Vulnerability Research Team (VRT). [Online]. Available: http://www.snort.org/vrt, [Retrieved: June, 2012].

[13] Sourcefire, Snort (version 2.9.5). [Online]. Available: http://www.snort.org, [Retrieved: June, 2012].

[14] The Barnyard2 Project. [Online]. Available: http://www.securixlive.com/barnyard2, [Retrieved: June, 2012].

[15] Snorby (version 2.2.6). [Online]. Available: http://www.snorby.org, [Retrieved: June, 2012].

[16] The OinkMaster Project. [Online]. Available: http://oinkmaster.sourceforge.net, [Retrieved: June, 2012].

[17] Snort Rules, Emerging Threats Project. [Online]. Available: http://rules.emergingthreats.net, [Retrieved: June, 2012].

[18] 1999 Training Data - Week 4, DARPA Intrusion Detection Evaluation. [Online]. Available: http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1999/testing/week4/index.html, [Retrieved: June, 2012].

[19] VMware Inc, Wmware Player (version 4.0.2). [Online]. Available: http://www.vmware.com/products/player, [Retrieved: June, 2012].

 ISBN: 978-1-61208-216-5

A Semantic Model to Characterize Pricing and Negotiation Schemes of Cloud Resources

Giuseppe Di Modica, Orazio Tomarchio
Department of Electric, Electronic and Computer Science Engineering
University of Catania
Catania, Italy
Email: {Giuseppe.DiModica, Orazio.Tomarchio}@dieei.unict.it

***Abstract*—Cloud computing technology has reached a good level of maturity. The market of cloud resources is still dominated by proprietary solutions for what concerns resource delivering, pricing models and Service Level Agreement. The research community is working hard to define specifications that try to standardize most of these aspects. When standards will get mature, customers will no more be locked-up to any proprietary technology, and full interoperability among clouds will be a reality. In the future cloud resource market the competition challenge will be played on the real capability of providers to accommodate customers' requests in a flexible way and to supply high and differentiated QoS levels. In this scenario a mechanism must be devised to support the match-making between what providers offer and what customers' applications demand. In this work we propose the definition of a semantic model to support the supply-demand matchmaking process in the future cloud markets. Leveraging on a semantic description of the cloud resources' features, customers will be able to discover cloud offers that best suit their own business needs. Tests conducted on an implementation prototype proved the viability of the approach.**

***Keywords*-Cloud computing; Price model; SLA negotiation; Ontology.**

I. INTRODUCTION

Cloud computing [1] has emerged as a paradigm able to offer resources in a flexible, dynamic, resilient and cost-effective way. Following the *service-oriented* paradigm, all cloud resources, both physical and logical, are virtualized and are offered "as-a-service". The real success of the cloud is mostly due to the considerable business opportunities that it produces for both consumers and providers of virtualized resources. On the one hand, the providers see in the cloud model a way to maximize the use of their computing assets and thus minimize the maintenance cost; on the other hand, the "pay-per-use" business model allows consumers to pay for only what they actually use, without any initial investment.

However, today we are still far from an open and competitive cloud and service market, where cloud resources are traded as in conventional markets. The main technological reason for this is the lack of interoperability of existing cloud technology [2], which is also leading to the phenomenon of *vendor lock-in*.

Another not technological, yet equally important reason is that, to date, cloud resources are offered according to strict pricing models and rigid Service Level Agreements (SLA). In a future open cloud market, users (customers) will demand for flexible pricing and resources' usage schemes to meet their specific computing needs; providers will have to negotiate with the customers for differentiated levels of quality of service.

In this paper we discuss about the need of more flexible charging models for cloud resources' usage, together with advanced negotiation protocols that could better support the public cloud model. We believe that, in order to build an effective matchmaking process between supply and demand, a structured model to describe resources' business features and applications' requirements is needed. To this purpose, we propose two ontologies for describing the resources offered by cloud providers on the one hand, and the application requirements expressed by customers on the other one. The final aim is to efficiently include pricing models, negotiation capabilities and service levels into resource publish/discovery mechanisms, that can then be enriched with tools to enable providers to easily characterize and advertise their resources, and customers to easily describe application requirements. A semantic matchmaking algorithm has been devised enabling customers to search for those cloud resources that best meet their requirements. A first prototype of the semantic discovery framework has been implemented. Experimental results show that semantic technologies are a powerful means that enhance the way resources' supply and demand are expressed and matched in the cloud markets.

The remainder of the paper is organized as follows. In Section II the background context is introduced and the issues inspiring this work are discussed. Section III describes the approach proposed for the definition of a cloud service discovery framework, and provides details on the mapping and the matching processes respectively. The implementation of a system prototype and results from tests are described in Section IV. In Section V recent works in literature are discussed, outlining the novelty of the idea proposed in this paper. We conclude the work in Section VI.

 ISBN: 978-1-61208-216-5

II. Analysis of current Cloud offering

The commercial success of cloud computing is witnessed by the individual success of few, very big companies that, by imposing their own proprietary solutions (e.g., Amazon's ".ami" and "EC2"), have made and are currently making huge profits by leasing their unused computational resources.

In a desirable scenario, the customer should be able to build up his own application independently of the specific cloud that it is going to run onto, define the application requirements in a standardized way, look for the cloud provider that best meets the requirements, negotiate for the service, deploy the application, monitor the application performance, move it to another cloud in the case that the service performance does not meet his expectation. However, the road that leads to cloud interoperability is long, because of several issues that still need to be addressed [2]. When such a target will be accomplished, a new scenario of business opportunities will open up to the old and the new stakeholders that will want to profit from the open market of cloud-based resources. Interoperability is the means by which also small companies can federate to each other to share their resources and propose themselves as an alternative to the big players. The European FP7 project Reservoir [3] is one of the first successful attempts to create an interoperable federation of cloud providers, spanning across different administrative domains, aiming at sharing their individual resources to respond to the customers' demand.

In the following, we analyze the panorama of the current cloud offerings by taking pricing models, negotiation protocols and service performance levels as key factors. We also take into account the customers' point of view, by analyzing how customers are used to characterize and specify application requirements under their business perspective.

A. Price Models

The main cloud paradigm's claimed strength is that resources (computing, storage, network) can be accessed on an *On-Demand* basis, and customers can be charged according to the actual resources' usage time. In particular, the CPU is usually charged by the hour, the data storage service is charged per GB/month, the data transfer service over the network is charged per GB. Providers also propose their customers an alternative pricing model based on *Resource Reservation*, which on the cloud provider's end provides an instant economic benefit (they receive an immediate payment for the reservation), and on the customer's end allows to save on the resource price provided that the resource itself is intensively used. Other cloud providers, instead of leasing "raw" computing resources, offers cloud-based services in the form of developing and execution platforms (PaaS) and applications (SaaS). Some decide to charge the customer according to the usage that the provided service make of the underlying raw resources. Others (mostly providing business oriented services) adopt a model that is more suited to those business applications that, once deployed, involve the interaction of many end users. The customer is then charged by month and by the number of end users that the application will have to serve (we refer to this model as *End-User-Based*). Finally, almost all commercial providers propose a *Free-Of-Charge* model, which is nothing but a try-before-buy strategy.

In the forthcoming cloud economy generation other pricing models might result more attractive for both the providers' and the customers' needs. In the process of optimizing the usage of internal resources, providers might want to encourage customers to access their resources during specific periods of underutilization (at night, or during the weekends), and thus would be willing to charge customers according to ad-hoc, time-oriented models. Again, in the same way like mobile phone operators do, providers might even offer their customers pre-paid packages of resources to be consumed as they like.

B. Negotiation Protocols

In the literature several proposals for the negotiation and management of SLA have appeared in the context of GRID and SOA, but many address the same issues in the cloud computing context too. Actually most of them provide limited or no support for dynamic SLAs negotiation, which we believe to be a strict requirement for the future cloud markets. As for the negotiation protocol, the OGF's WS-AgreementNegotiation [4] is the most notable standardization effort. The proposal is an extension of the former WS-Agreement recommendation, and is still in progress. It just supports the one-to-one negotiation scheme and the very simple offer/counter-offer dynamics. The approach is not efficient and flexible enough for complex application areas. Alternatives (such as **auctions**[5]) are also suggested as more appropriate for highly dynamic context. One of the objectives of the SLA@SOI European FP7 project [6] was to provide negotiation mechanisms for exchanging offers and counter offers between customers and providers in a SOA context. The implemented framework (SLAM) promises support for both one-to-one and one-to-many negotiations, allows for multiple rounds of negotiation, and can be adopted in agent marketplace as well as broker based architectures. The Vienna Service Level Agreement Framework (VieSLAF) architecture for cloud service management [7] introduces the concept of meta-negotiations to allow two parties to reach an SLA on what specific negotiation protocols, security standards, and documents to use before starting the actual negotiation.

In the actual market of virtualized resources, we notice that Amazon has launched the *Spot Instances* model, which can be seen as an example of a particular negotiation model that has been adopted to resolve the customers' competition on the provider's unused resources. Depending on the provider's business strategies and on the amount

 ISBN: 978-1-61208-216-5

of unused resources, other negotiation models might be employed. We claim that auction-based models would bring benefits to providers and to customers as well. The latter will have the chance to search for resources according to the associated negotiation scheme that best suite their own business strategies and needs.

C. Service Performance Levels

The performance features that cloud providers advertise are usually vague, just focusing on virtual machines computation speed.The only parameter which is quantitatively expressed and granted by all the commercial providers is the *availability* of resources. All providers guarantee a very high level of resource availability (from 99% upwards), prevent any user data loss by allocating extra back-up storage, support customers to face any technical issue. The competition among the providers is played on both the price at which the resources are sold and the capability of sustaining the promised, *guaranteed service levels*. Some providers further differentiate their service offer. Besides provisioning what we call a standard *basic service level*, which is the core activity of their business, some of them also offer a *premium service level*, which provides more guarantees than the basic and adds extra services.

In the future, in order to satisfy the customers' heterogeneous and dynamic business requirements, the cloud providers might be encouraged to propose new models. To cater for more fine-grained customer requirements, providers might want to propose customizable plans of service levels, that will enable customers to build their own desired service level provisioning.

D. Application Requirements

Every application needing some computing power could technically run on a cloud. Still security is a big concern that prevents service providers from unconditionally deploying their applications on the cloud. Generally speaking, before moving an application to the cloud a cost/benefit analysis must be carefully done. The decision concerns both whether to move onto the cloud or not, and to select the cloud offer that fits.

One should verify, according to the company's business objectives to be accomplished and to how much mission-critical the application is, whether the application to be deployed requires a guaranteed service level or a best effort is enough. If the former is to be chosen, again, depending on the business requirements of the application, a choice has to be made between a basic or a premium service level. Further on, the choice of the pricing model that best fits must be made according to the *application's profile*, i.e., the application's specific usage pattern: if such pattern is "dense" (resources are continuously used within a time frame), reserved-based solutions are to be preferred; otherwise, the on-demand pricing model will result more convenient.

All the choices must be made checking that the budget they require is compatible with the company's investment capability. For example, a service level might fit a given application's profile, but might not be affordable for the company; on the contrary, a more affordable service level would make the company save money, but might not fit the strict application's requirements. In most cases a compromise must be searched for.

III. Cloud service discovery framework

The previous analysis of current cloud offering has shown that provider and customer perspectives are quite different. The former seeks to maximize the profit and the utilization level of the IT asset that they have invested on. The latter just needs to make fine-tuned searches in the market in order to discover the service fitting their specific business needs. We have then designed a service discovery framework that exploits semantic mechanisms to favour the matchmaking of the providers' offer and the customers' demand. Two OWL-based ontologies have been developed to characterize respectively the provider and the customer perspectives. In particular, the first ontology semantically describes the features of the resources being offered by cloud providers (see Figure 1), the second one describes the application's business requirements demanded by customers (see Figure 2). For a detailed description of these ontologies refer to [8]. Since each ontology contains semantic concepts belonging to two different domains, we have devised a mapping process that transforms application requirements into "semantically" equivalent resource features, i.e., features that best represent the application requirements in the domain of resources. The mapping's purpose is to put application requirements and resource features on a common semantic ground (that of cloud resources) on which a semantic procedure will try to make the match.

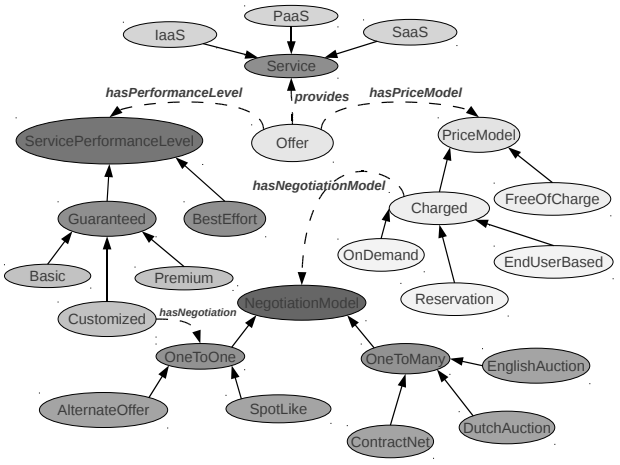


Figure 1. Resource features ontology

Figure 3 depicts the two semantic domains, along with the mapping and matchmaking processes. In the figure, the

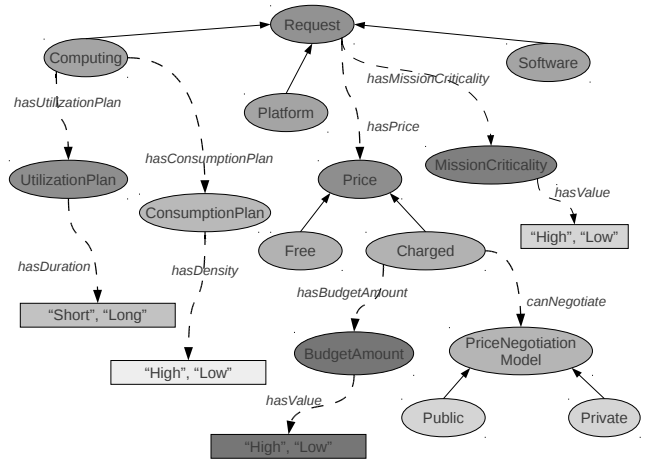


Figure 2. Application requirements ontology

filled circles represents, respectively, real requests issued by customers (within the application requirements' domain) and real offers advertised by service providers (resource features' domain). Through the mapping process the application requirement AR_4 is transformed into its "equivalent" resource feature offer RF_6 (empty circle) in the offers domain. Such resource feature does not necessarily coincide with a real offer, but rather represents the ideal offer that would perfectly match the considered application requirement. In the next step, the matchmaking procedure will explore the resource features' domain in order to search for concrete offers that show a *semantic affinity* to RF_6 (those covered by the gray area in the figure). The final outcome of the entire process will be a list of concrete offers, sorted by the semantic affinity degree, that may satisfy the needs represented by AR_4.

In the following subsections we provide some details on how the mapping and matchmaking processes work.

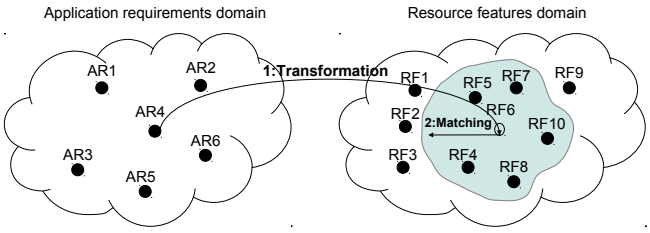


Figure 3. Mapping and matching

A. Mapping

The mapping process is a simple procedure that applies a list of mapping rules. Rules have been defined using the Semantic Web Rule Language (SWRL) [9]. SWRL was chosen since it is a W3C specification and it copes well with OWL-based ontologies. The objective of each rule is to transform a specific application requirement into the ideal, best matching resource feature.

A group of chained semantic rules drive the mapping from individuals of the Application requirements' ontology to individuals of the Resource features' ontology. A rule engine takes a request in input, applies the sequence of rules and, according on the rules that match, incrementally builds up the ideal offer. For the sake of brevity, we report only a significant subset of rules:

1) $request : Request(?request) \wedge offer : Offer(?offer) \rightarrow hasMatchedOffer(?request, ?offer)$
2) $hasMatchedOffer(?request, ?offer) \wedge request : Computing(?request) \wedge offer : provides(?offer, ?service) \rightarrow offer : IaaS(?service)$
3) $hasMatchedOffer(?request, ?offer) \wedge request : Platform(?request) \wedge offer : provides(?offer, ?service) \rightarrow offer : PaaS(?service)$
4) $hasMatchedOffer(?request, ?offer) \wedge request : Software(?request) \wedge offer : provides(?offer, ?service) \rightarrow offer : SaaS(?service)$

Rule 1 just states that, given a generic request in the application requirements' domain, a corresponding ideal offer exists in the resource features' domain. Rules 2 through 4 handle the different type of cloud services that can be requested. The rules are very intuitive, and states that a request for Computing resource is mapped onto an offer of the type IaaS, a request for a Platform resource is mapped onto a PaaS offer, and a request for a Software resource maps to an offer of the type SaaS.

B. Matchmaking

After the mapping process has elaborated the ideal offer, the matchmaking process will start exploring the domain of the real offers in order to find those ones whose features best meet the initial application requirements. In particular, for each offer advertised in the market, the matchmaking process will evaluate the *semantic affinity* between that offer and the ideal one. The semantic affinity will reveal how close a real offer is to the customer expectations. The semantic affinity will be a value in the range [0,1], being 1 the highest achievable affinity. The function that calculates the semantic affinity is the following:

$A = Serv_a * W_{serv} + Price_a * W_{price} + Perf_a * W_{perf} + Neg_a * W_{neg}$

The overall affinity between the ideal offer and a real offer is obtained by summing up the sub-affinities evaluated on each offer's feature: service, price model, performance level and negotiation model. So, for instance, the addendum $Price_a * W_{price}$ represents the sub-affinity evaluated on the price feature. In particular, $Price_a$ is the outcome of the semantic comparison between the price concepts exposed by the two individuals (the offers), while W_{price} is a weight factor. We plan to use the weight factor to let the customer tune the affinity algorithm according to customizable priority criteria.

 ISBN: 978-1-61208-216-5

We now provide some details on the semantic comparison of concepts. Let O_j be a generic offer, characterized be the semantic concepts: $Serv_{o-j}$, $Price_{o-j}$, $Perf_{o-j}$, Neg_{o-j}. In order to evaluate the overall semantic affinity of two offers O_{ideal} (the ideal offer that is the outcome of the mapping process) and O_{real} (a real offer in the market place), couples of homologous concepts must be compared.

The semantic affinity values for all the possible cases are shown in the following:

- 1, if the two concepts are semantically equivalent;
- 1, if $C_{o-ideal}$ is the father of C_{o-real};
- 0.5, if the two concepts are siblings and the father is the root concept in the considered branch;
- 0.75, if the two concepts are siblings and the father is a non-root concept in the considered branch;
- 0, if $C_{o-ideal}$ is not expressed;
- 0.5 in any other case.

The algorithm assigns the highest value to equivalent concepts, or to concepts that are in a father-son relationship. Instead, it penalizes two concepts that are direct descendants of a root concept, as in our ontology siblings concepts whose father is root usually represent opposite concepts (e.g., Charged vs FreeOfCharge, Guaranteed vs BestEffort). Conversely, siblings whose father is a non-root concept are considered different but someway "close" concepts (e.g., On-Demand vs Reservation, EnglishAuction vs DutchAuction), therefore they are given a higher grade of affinity.

IV. Implementation and tests

A prototype of the framework has been implemented and tested. The core of the framework consists of an *ontology mapper*, that makes use of a rules engine provided by the Jess library, a *matchmaker* supported by a semantic reasoner powered with the Pellet library, and a *repository* of advertised cloud offers. Customers are then provided with a front-end tool to build and submit cloud requests, while at this stage the repository of cloud offers was populated by hand. In the future we plan to implement a tool that will help providers to build their offers and push them to the repository. For the test purpose, we generated a complete set of offers spanning the whole semantic domain of resources' features. Afterwards, several different requests were generated, each of them asking for a specific cloud service. For each submitted request, the framework replied with a list of fitting offers. In the following we describe two sample requests and analyze the corresponding results provided by the discovery procedure. In the first request the customer asks for a service of type Platform, for whose price he is willing to negotiate in the context of a public auction:

$R1(Type : Platform, PriceModel : Charged,$
$NegotiationModel : Public)$

After submission, the mapping process transformed $R1$ into the following ideal offer:

$O1_{ideal}(Service : PaaS, PriceModel : EndUserBased,$
$ServicePerformanceLevel : Guaranteed,$
$NegotiationModel : OneToMany)$

For that offer, the matchmaking process produced the results depicted in Table I.

Table I
List of matching offers for request $R1$

Offer #	Service	PriceModel	SPL	NegModel	Affinity
39	PaaS	EndUserBased	Basic	ContractNet	1.0
49	PaaS	EndUserBased	Premium	EnglishAuction	1.0
48	PaaS	EndUserBased	Premium	DutchAuction	1.0
...	...	...	...	...	...
36	PaaS	EndUserBased	Customized	n.a.	0.875
35	PaaS	EndUserBased	Basic	n.a.	0.875
...	...	...	...	...	...
3	SaaS	FreeOfCharge	BestEffort	n.a.	0.5
1	IaaS	FreeOfCharge	BestEffort	n.a.	0.5

As the list is very long many results have been omitted. On the top of the list the perfectly matching concrete offers appear. The offers with an affinity value of 0.875, have a partial matching, as those offers do not provide any negotiation. The offers at the bottom do not match because of differences in both the service type and the service performance level. Here is the second request that we tested:

$R2(Type : Computing,$
$UtilizationPlan- > hasDuration : long,$
$ConsumptionPlan- > hasDensity : high)$

After submission, the mapping process transformed $R2$ into the following ideal offer:

$O2_{ideal}(Service : IaaS, PriceModel : Reservation,$
$ServicePerformanceLevel : Premium,$
$NegotiationModel : no)$

For that offer, the matchmaking process produced the results depicted in Table II.

Table II
List of matching offers for request $R2$

Offer #	Service	PriceModel	SPL	NegModel	Affinity
10	IaaS	Reservation	Premium	n.a.	1.0
...	...	...	...	...	...
33	IaaS	Reservation	Premium	DutchAuction	0.9375
32	IaaS	Reservation	Premium	ContractNet	0.9375
...	...	...	...	...	...
12	IaaS	OnDemand	Basic	ContractNet	0.8125
...	...	...	...	...	...

As expected, offers proposing the reservation-based price model have the best matching; in fact, they perfectly fit the application requirements concerning the utilization plan and the resource consumption plan. Offers that propose auctions are lightly penalized. Offers proposing a basic performance level get penalized even more.

V. Related work

Several standard organizations are working hard to propose specifications that will enable future scenarios of

interoperable cloud services. An exhaustive review of the research efforts dealing with interoperability issues in cloud computing systems was produced in the context of Cloud4SOA project [10]. Some attempts to design ontologies for the definition of cloud-related concepts and relationships have recently appeared in the literature. Still, there is no proposal that has reached a broad acceptance from the community, nor all features of the cloud domain have been fully covered, so far, by existing proposals. Some works([11], [12]) have tried to define taxonomies for cloud-based systems. They mostly identify and classify cloud delivery models, services and resources; some also deals with requirements like fault tolerance and security. One of the most complete cloud taxonomy is maintained and continuously updated by OpenCrowd([13]): in this project, existing cloud providers and cloud-related software are classified according to a specific scheme. In the aim of defining an open and standardized cloud interface for the unification of cloud APIs, the Unified Cloud Interface (UCI) Project [14] has proposed and released an RDF-OWL cloud data model mostly covering the definition of resources in the cloud domain. To our knowledge, the mOSAIC ontology([15]) is the most complete ontology that was proposed so far. It inherits most of the elements defined in other proposals (OCCI, NIST, IBM), and covers aspects like deployment models, service models, resources, services, actors, consumers, functional and non functional properties, languages, APIs. The ontology was developed in OWL and is used for semantic retrieval and composition of cloud services in the mOSAIC project.

The work discussed in this paper aims at discussing aspects of cloud interoperability not covered by any of the works cited above. The proposed perspective is that of a global market of cloud resources, where there is the need of a characterization of what is offered and demanded by actors in terms of business profit and utility respectively. The proposed ontology, therefore, covers a new portion of the cloud's domain of knowledge; nonetheless, it can be integrated to existing ontologies/taxonomies.

VI. Conclusion and future work

The future market of cloud services will have to provide novel and advanced matchmaking processes in order to account for the providers' and the customers' dynamic and heterogeneous business requirements, respectively in terms of profit and utility. The work presented here aims to define a cloud offer discovery framework based on semantic technologies. A matchmaking procedure has been devised to semantically search the offers' domain in order to provide the customer with a list of most profitable offers. Tests were run on a prototype of the framework and proved the viability of the proposed model. In the future, we are planning to enhance the semantic model by extending the ontologies and accordingly enriching the semantic rules.

References

[1] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, Sep. 2008, pp. 5 –13.

[2] A. Parameswaran and A. Chaddha, "Cloud Interoperability and Standardization," *SETLabs Briefings*, vol. 7, no. 7, pp. 19–26, 2009.

[3] Reservoir Consortium, "The Reservoir Project," http://www.reservoir-fp7.eu/, [Last Retrieved: July 2012].

[4] The Open Grid Forum, "WS-Agreement Negotiation Specification," http://forge.gridforum.org/sf/go/projects.graap-wg/, 2011, [Last Retrieved: July 2012].

[5] V. Krishna, *Auction Theory*. Academic Press, 2002.

[6] The SLA@SOI Consortium, "SLA@SOI Negotiation Platform," http://sla-at-soi.eu/publications/, [Last Retrieved: July 2012].

[7] I. Brandic, D. Music, P. Leitner, and S. Dustdar, "Vieslaf framework: Enabling adaptive and versatile sla-management," in *Proceedings of the 6th International Workshop on Grid Economics and Business Models*, ser. GECON '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 60–73.

[8] G. Di Modica and O. Tomarchio, "A semantic discovery framework to support supply-demand matchmaking in cloud service markets," in *2nd International Conference on Cloud Computing and Services Science (CLOSER 2012)*, Apr. 2012.

[9] Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., and Dean, M., "SWRL: A semantic web rule language combining OWL and RuleML," http://www.w3.org/Submission/SWRL/, 2004, [Last Retrieved: July 2012].

[10] N. Loutas, V. Peristeras, T. Bouras, E. Kamateri, D. Zeginis, and K. Tarabanis, "Towards a reference architecture for semantically interoperable clouds," in *IEEE Int. Conf. on Cloud Computing Technology and Science (CloudCOM 2010)*, 2010, pp. 143–150.

[11] B. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, Aug. 2009, pp. 44 –51.

[12] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Grid Computing Environments Workshop, 2008. GCE '08*, Nov. 2008, pp. 1 –10.

[13] OpenCrowd, "The OpenCrowd Cloud Taxonomy," http://cloudtaxonomy.opencrowd.com/, [Last Retrieved: July 2012].

[14] Cloud Computing Interoperability Forum, "UCI Cloud OWL Ontology," http://code.google.com/p/unifiedcloud/, [Last Retrieved: July 2012].

[15] F. Moscato, R. Aversa, B. Di Martino, T. Fortis, and V. Munteanu, "An analysis of mOSAIC ontology for Cloud resources annotation," in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, Sep. 2011, pp. 973 –980.

 ISBN: 978-1-61208-216-5

Controlling Data-Flow in the Cloud

Mandy Weißbach and Wolf Zimmermann
Institute of Computer Science, University of Halle
06120 Halle (Saale), Germany
Email: {weissbach, zimmermann}@informatik.uni-halle.de

***Abstract*—A big obstacle for using cloud services is that users have no control over the locations where their data are stored or processed, respectively. This paper presents a program analysis approach that enables clients to negotiate services with undesired locations. Clients may only use services that guarantee not to use (directly or indirectly) services on undesired locations for processing or storing the clients' data. In order to increase trust in the answers given by services during the negotiation process, a cryptographic approach similar to Web page certification is proposed. We show that a static data-flow analysis combined with a cryptographic approach ensures that clients' data do not reach undesired locations in the cloud.**

Keywords- data-flow; service-level agreement; cloud security.

I. INTRODUCTION

One major obstacle in using cloud services is that clients have no control where their data are being stored and processed. National data protection laws may require from clients to satisfy some standards. For example, EU-directives imply that it is illegal to pass personal data to environments where the access to data cannot be controlled [1]. Recently, there is an even stronger proposal [2]. This may apply towards storing data as well as to the results from processing data. However, if cloud servers located at different locations, they need to obey national laws on the server's location, and these might be rather different than the location of the cloud users and therefore there might be unauthorized access to clients' data that might be legal in the cloud servers country.

Unfortunately, encryption of data is only a solution when data are just stored (see, e.g., [3]), but it is currently not a solution when they are being processed [4] (some work on directly processing encrypted data exist, but it is just at the beginning and it is not clear whether this research will be successful at the end). Therefore, it is crucial that cloud service users can require that their data are stored in certain locations or exclude some locations for storing and processing their data. But, cloud service providers themselves prefer where to store the data of the service users. Even worse, they may use other cloud services which themselves may use other services and so on. Thus, it seems almost impossible to control where data are processed and stored. Thus, several authors see this issue as one of the major challenges in cloud computing [5], [6], [7].

In this paper, we propose a service-level agreement approach to ensure that the data of cloud service users are not processed or stored at undesired locations. Typical service-level agreement (SLA) approaches such as, e.g., reliability or response time can be measured by service users. If the chosen service violates its assured service quality, the service user is enabled to use alternative services. However, the problem in this work has different characteristics: (i) it is not measurable whether data are not being stored and processed at undesired locations, (ii) a violation cannot be observed by service users, and (iii) if a cloud service violates directly or indirectly the SLA, there already is a possible threat for the service user, i.e., the damage is sustained. Thus, service violations in the context of this work should be avoided, and service users have to trust the agreement.

We tackle the problem of avoiding storing or processing data at undesired location by data-flow analysis. In particular, this analysis ensures that either data do not leave the cloud server hosting the cloud service or the data are only transferred (possibly in processed form) to cloud services ensuring that the data received are neither stored nor processed at undesired locations. This approach enables the cloud service to provide the correct agreement. However, a malicious service may give the wrong answer. We propose cryptographic methods analogous to web page certification in order to increase the trust into the negotiated service-level agreement.

The paper is organized as follows: an example of a service model is provided and explained in Section II. In Section III, a data-flow analysis with respect to the given example is done. Section IV proposes a cryptographic approach in order to increase the trust in the answer given by the service model of Section II. On top of that, Section V presents an approach to choose dynamically a service that can be trusted. Section VI disscuses related work and Section VII concludes this work.

II. APPROACH

This section demonstrates the underlying approach. In our service model, we assume that each service A provides a set of functions, denoted by $Provided_A$. This might be given as a WSDL-description (Web Services Description Language). Furthermore each service A might use other services. We assume that this is not hard-coded in the implementation of A, but there is a variable I a where I contains the set of functions that is called on a, and a can be bound

 ISBN: 978-1-61208-216-5

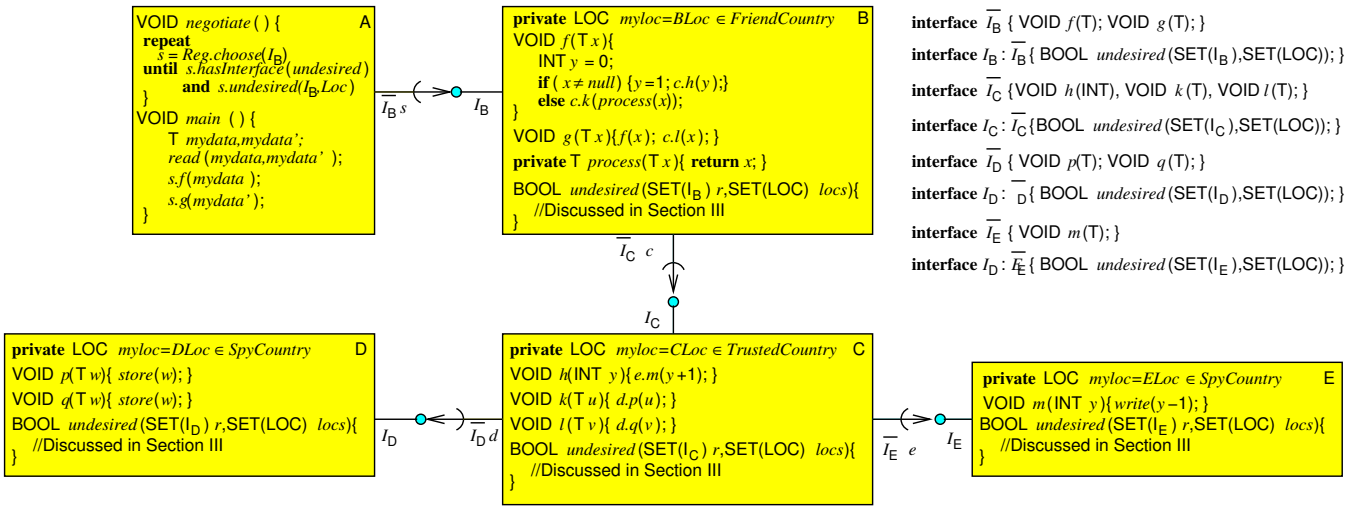


Figure 1. Storing Data at Undesired Locations

(dynamically) to a service X that provides at least I, i.e., $I \subseteq Provided_X$. Functions in I are called *required functions* of A w.r.t. a. The set of candidate services must be published and we assume that a registry Reg maintains all published services. For the purpose of this and the next section, we assume that the use structure is acyclic. Section IV shows how this assumption can be dropped.

Example 1: Consider services A and B in Fig. 1. $A.s$ is bound to service B and $B.c$ is bound to service C. The provided interface of B is $Provided_B = \{f, g, undesired\}$. The required functions of A w.r.t. s are $\{f, g\}$. The required functions of B w.r.t. c are $\{h, k, l\}$. ■

Remark: It is part of service-level agreement approaches that negotiations bind services to these variables. For simplicity, we only consider a set of functions for the selection of candidate services. However, this can easily be replaced by other match-making approaches, e.g., it can be based on contracts or adapters can be included. □

In the context of this paper a client would like to negotiate an agreement that a selected service guarantees to avoid data-flow from the client's data to a set Loc of undesired locations. This ensures that the client's data are not stored at undesired locations. For the purpose of negotiation, service A may offer a function $undesired \in Provided_A$ that returns true iff data flows via some operations o from the provided interface of A to services at undesired locations. It is sufficient to take into account only the set $S \subseteq Provided_A$ of operations used by the client. If A uses another service B, it needs to ask B (via B's function $undesired$) whether it can guarantee that A's data do not flow to a location in $l \in Loc$. Obviously, this needs only to be guaranteed for those operations of B where B passes (possibly processed) data of A. For simplicity, we assume that each service X knows its location and this location is stored in a constant $X.myloc$.

Example 2: Consider the services A, B, C, D, and E in Fig. 1. Service A would like to use service B. Service B is located in *FriendCountry*. B itself uses service C located in $TrustedCountry$ while C uses serviced D and E located in $SpyCountry$. For the example, we assume that all services (except possibly A) are published.

Suppose that client A wants to avoid storing its data neither in their original nor in processed form at servers in *SpyCountry*. Thus, before client A actually uses service B it would like to know whether data passed to B are never stored (neither in original nor in processed form) at a server in $SpyCountry$. Let Loc be the set of servers in $SpyCountry$. The procedure $negotiation$ searches for a published service B offering at least the operations specified in I_B where I_B is the set of functions of the required service that are called from A. For the purpose of negotiation, A calls $undesired(I_B, Loc)$ because A calls $b.f(mydata)$ and $b.g(mydata')$, if b is bound to service B. B calls functions $h, k, l \in Provided_C$ if c is bound to a service C. A call of $B.f$ implies that data of A flow to the calls $c.k(z)$ and $c.l(x)$, but there is no flow from data of A to the call $c.h(y)$. Thus, the call $undesired(I_B, Loc)$ must return $true$ only if $B.myloc \notin Loc$ and $undesired(\{k, l\}, Locs) = true$. Note that function h needs not to be considered because $mydata$ does not flow to y in the call $c.h(y)$. The functions $k, l \in Provided_C$ call $p, q \in Provided_D$ if $C.d$ is bound to a service D. The arguments of the calls $c.k(z)$ and $c.l(x)$ flow to the calls $d.p(u)$ and $d.q(v)$, respectively. Thus, there is a flow from the data of A to service D located in $SpyCountry$ which could store these data. Therefore, the negotiation must fail. $C.undesired(\{k, l\}, Locs)$ must return $false$ and therefore $B.undesired(I_B, Loc)$ returns $false$, i.e., A cannot use B.

Suppose there would be an alternative service D' with the same implementations of p and q, but its location

is in $MyCountry$. Then, if $C.d$ is bound to D' instead of D, $D'.undesired(\{p,q\}, Loc)$ returns $true$ because D' does not use other services. Thus, in this case $C.undesired(\{k,l\}, Locs)$ can return $true$ and therefore also $B.undesired(I_B, Loc)$ returns $true$. Hence, A can use B. Note, that $C.e$ is still bound to service E in $SpyCountry$ but there is no flow from the data of A to E because E is only used in h and there is no flow from data of A to the call $c.h(y)$ in B. ■

In general, if a service A negotiates with a service B for avoiding undesired locations Loc, A must additionally provide the set of functions $O \subseteq Provided_B$ used by A. If B calls functions provided by other services, and data from A flow to B then, B must ensure that these services are neither located in an undesired location nor passed directly or indirectly to a service located in an undesired country. Therefore, B has to negotiate with the required services for assuring that the data of A are never passed to services in an undesired location.

Remark: A cyclic use-relation would lead to non-terminating negotiations. One possibility to overcome this problem is that after a certain time, the negotiation with a service B is interrupted and another service is considered for negotiation, i.e., the function $undesired$ terminates after a certain time and returns $false$. □

III. Data-Flow Analysis

For the implementation of $undesired$, the data-flow from the provided functions to the required functions needs to be analyzed. For such a data-flow analysis, we refer to [8] (an interprocedural def-use-chain is needed). Let x be a parameter of a function $f \in Provided_A$ provided by a service A. The result of the program analysis is a predicate $DEP_{e,x}$ for each argument e of a call f of a required function of A. $DEP_{e,x}$ is true if the value of e depends on x.

Example 3: Consider services A and B in Fig. 1. The function $f \in Provided_B$ has a parameter x of type T. If x is not $null$ the function $h \in Provided_C$ is called. So $DEP_{x,y} = false$ because the value of the argument y does not depend on x.

Remark: The program analysis is *conservative*, i.e., the value of e might be independent of x although the program analysis computes $DEP_{e,x} = true$. However, if $DEP_{e,x} = false$, then it is guaranteed that the value of e is independent of x. An exact computation of $DEP_{e,x}$ would be undecidable. □

Let $T\ f(T_1\ x_1, \ldots, T_n\ x_n) \subseteq Provided_A$. Then, the *slice of* f consists of all set of required functions of A that are called with an argument depending on one of the parameters x_i, i.e.,

$Slice_f \triangleq \{p : \exists s.p(e_1, \ldots, e_k) \in A \bullet \exists i, j \bullet DEP_{e_i,x_j}\}$

The *slice of a set of functions* $S \subseteq Provided_A$ is defined by

$Slice_S \triangleq \bigcup_{f \in S} Slice_f$

Let B be a service that is used by A and $S \subseteq Provided_A$. Then

$Called_{S,B} \triangleq Slice_S \cap Provided_B$

is the set of functions of B called by A that may depend on a parameter of a function in S. For these functions, it must hold that B doesn't pass the data directly or indirectly to a service at undesired locations. Thus, the requirement for service B is

$UnDes_{B,S,L} \triangleq Called_{S,B} = \emptyset \vee B.undesired(Called_{S,B}, L) = true$

Example 4: Consider service B in Fig. 1. The $Slice$ of the function $f \in Provided_B$ and $g \in Provided_B$ are defined by

$Slice_f \triangleq \{k\}$ and $Slice_g \triangleq \{l\}$

because there is no dataflow from x to y ($DEP_{x,y} = false$), which means $h \notin Slice_f$. With $Slice_f$ and $Slice_g$, it is

$Slice_{f,g} = \{k, l\}$.

So only the functions k and l are called with data stemming from A by service B. Hence,

$Called_{\{f,g\},C} \triangleq \{k, l\}$.

In the next step the slices $Slice_k$ and $Slice_l$ are computed:

$Slice_k = \{p\}$ and $Slice_l = \{q\}$.

Therefore, p and q are called with data from A over B:

$Slice_{k,l} = \{p, q\}$.

Considering $Slice_{k,l}$ and the provided functions of the service D and E, only service D is called with data from A:

$Called_{\{k,l\},D} \triangleq \{p, q\}$ and $Called_{\{k,l\},E} \triangleq \emptyset$.

Now, we can compute if there exists a data-flow to an undesired location. Thus,

$UnDes_{D,\{k,l\},SpyCountry} \triangleq false$

because of

$D.myloc \in SpyCountry$ and $Called_{\{k,l\},D} \notin \emptyset$.

Hence,

$UnDes_{E,\{k,l\},SpyCountry} \triangleq true$

because of

$Called_{\{k,l\},E} = \emptyset$.

So,

$UnDes_{C,\{f,g\},SpyCountry} \triangleq false$

because of

$Called_{\{f,g\},C} \notin \emptyset$ and

$C.undesired(Called_{\{f,g\},C}, SpyCountry) = false$.

We can conclude that there is a data-flow from service B over C to D. And D is a service with an undesired location. This violation is produced by service D. Note that although E is called and $E.myloc \in SpyCountry$, this is not omitted, as no data flows from A to E.

Thus, for a service A, the function $undesired$ can be implemented as shown in Fig. 2.

Theorem 1: Let X be a service with an undesired location $X.myloc \in L$. If there is a data-flow from a parameter x of a function $f \in S \subseteq Provided_A$ to X, then $undesired(S, L) = false$.

```
/*@return : false-> data-flow to undesired location(s)
            true-> data-flow only to desired locations*/
BOOL undesired(SET(Provided_A) S, SET(Locations) L) {
   if myloc ∈ L return false;
   foreach service X used by A do
      if ¬UnDes_{X,S,L} return false;
   return true;
}
```

Figure 2. Implementation of *undesired*

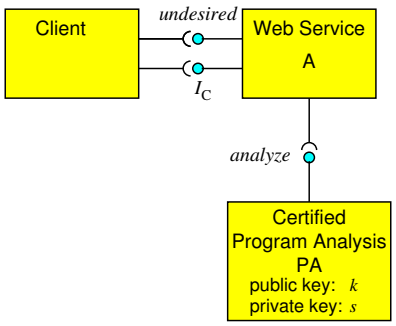


Figure 3. Trusted Agreement

Proof: Let X be a service with $X.myloc \in L$, i.e., the location of X is undesired. The maximal distance $d(Y, X)$ of a service Y to X is the length of the longest cycle-free path from Y to X w.r.t. the use relation. We prove the claim by induction on the maximal distance to X.
BASE CASE: $d(Y, X) = 0$. Then $Y = X$ and therefore $Y.myloc \in L$. For this case, *undesired* returns *false*.
INDUCTIVE CASE: $d(Y, X) > 0$. Let be $S \subseteq Provided_Y$ such that there is a data-flow from a parameter x of a function $f \in S$ to X. Thus, there must be a data-flow (internal to Y) to argument e of function call $z.g(\cdots e \cdots)$ where z is bound to Z and there is a data-flow from the corresponding parameter x of $g \in Provided_Z$ to X. Obviously $d(Z, X) < d(Y, X)$. Thus, by induction hypothesis it holds $undesired(S', L)$ returns *false* for each $S' \subseteq Provided_Z$ with $g \in S'$. By definition, it is $DEP_{e,x} = true$, thus, $g \in Slice_f$ and therefore $g \in Slice_S$. Since $g \in Provided_Z$, it holds $g \in Called_{S,Z}$. Thus, $Called_{S,Z} \neq \emptyset$ and therefore the induction hypothesis implies $UnDes_{Z,S,L} = false$. Since $undesired(S, X)$ only returns *true* if $Y.myloc \notin L$ and $UnDes_{Z,S,L} = true$ for all services used by A, it must return *false*. ■

IV. TRUSTED AGREEMENT

The approach of Sections II and III makes some idealistic assumptions: First, it assumes that each service is not malicious, i.e., it gives the correct answer according to Theorem 1. Second, there are no cycles in the use-relation. In this section, we present an approach to increase the trust in the answer given by a service that is also able to deal with cycles in the use-relation.

The main idea is similar to the verification of web pages, cf. Fig. 3: there is an independent certified program analysis service PA that performs the program analysis and computes the result of *undesired*. The following negotiation protocol increases the trust of the client to the analysis result:

Step 1: *Client* tells A that it would like to negotiate undesired locations
Step 2: A selects a certified program analysis PA and returns PA's public key k to *Client*.
Step 3: *Client* uses k to check whether PA is certified. If this is the case, *Client* encrypts its query $undesired(S, L)$ with k and passes it to A. If k does not belong to a certified program analysis, then *Client* may refuse to choose A or request another program analysis.
Step 4: A passes the encrypted query $undesired(S, L)$ together with its source text to PA. For security reasons, the source text is also encrypted with k.
Step 5: PA first decrypts the query and the source text of A. Then it performs the program analysis according to Section III. Finally, it signs the query $undesired(S, L)$ and the result with its private key s and passes it to A.
Step 6: A passes the signed result to *Client*.
Step 7: *Client* decrypts the signed result with the public key k of PA. Then *Client* verifies whether its query was being analyzed and whether the answer is *true*. If yes, then it accepts A, otherwise it refuses to choose A.

Since *Client* obtains the public key k of a program analysis, it can verify whether the program analysis can be trusted. Furthermore, the encryption of the query in Step 3 keeps it secret to A. Thus, A needs more effort to be malicious because the private key s of PA is required to decrypt the query, which is needed for the manipulation of the query. A possibility of A to be malicious would be that it creates its own (malicious) query q, encrypts it with k and passes it to PA. However, in Step 5 the analysis result together with the query is signed by PA's private key s. Since this key and the original query are secret to A, A is unable to replace the responded and manipulated query q of the PA by the original query. *Client* would discover such a manipulation at Step 7.

Remark: At first glance, it seems to be a severe restriction that A must pass its source text to PA. However, A can choose a program analysis PA that it trusts *before* offering PA to *Client*. □

Sections II and III demonstrate that PA might itself query services B used by A while performing the program analysis. In this case PA has the role of a client and B has the role of the service being queried. Hence, the above protocol can be used to negotiate with B. As PA is able to keep track of the analysis requests of A, it can check for cycles before processing the analysis request. In particular, it checks whether a query $undesired(S, L)$ for A is currently

```
BOOL undesired(SET(Provided_A) S, SET(Locations) L) {
    if myloc ∈ L return false;
    foreach service variable I_x x of A do
        while ¬x.hasInterface(undesired) ∨ ¬UnDes_{x,S,L} do {
            x = Reg.choose(I_x)
            if x = null return false;
        }
    return true;
}
```

Figure 4. Choosing an Adequate Service

being analyzed, i.e., whether there is an open analysis request $undesired(S', L)$ with $S' \subseteq S$. If yes, it can return immediately $true$. This is valid because if there is a data-flow from S' to an undesired location l, then there must be another call of a provided function to a service B with a data-flow to an undesired location.

Remark: We assume that the services are installed correctly and that we can use the advantages of trusted cloud federations [9], [10] in order to avoid that hackers change the implementation of the services. □

V. Dynamically Choosing a Service

In the scenarios of the previous sections, if a client requests a service A for avoiding data-flow to undesired locations and A itself may request for avoiding data-flow to undesired locations, then the chosen service B is not changed. However, service A might decide to choose an alternative service that fulfills the requirements for A. Thus, instead of returning $false$ if a possible data-flow to an undesired location is discovered, cf. Fig. 2, it can be searched for a service that guarantees that there is no data-flow to an undesired location, cf. Fig. 4. If there is no such service (i.e., $Reg.choose(I_x) = null$) then $false$ is returned. If $undesired(S, L)$ returns $true$ each service variable x of A is bound to a service X. Thus, for each call $x.f(\cdots) \in Slice_S$, there is no data-flow to an undesired location.

Remark: The search for an adequate service in the registry Reg might take a long time. An alternative would be to bound the number of tries to find an adequate service. □

The problem with this approach is that the service A needs to know the undesired locations. Thus, encrypting the analysis request with PA's public key prevents A from choosing alternative services according to Fig. 4. However, the program analysis PA could choose an alternative service on behalf of A. Thus PA could tell A which services it can choose. This dynamic choice can be achieved by changing the last step of Step 5 in Section IV: A positive answer is passed to A as in Step 5. However, if PA's answer is negative, it performs the procedure in Fig. 4. Any query $undesired(Called_{S,b}, L)$ to services b used by A is passed by A to the service bound to b. The result is passed back to PA which can tell A whether to bind the chosen service to b. If PA finishes the procedure in Fig. 4, it passes its final answer to A.

Thus, the answer is partially not being kept secret to the service being analyzed. However, the final answer is still kept secret and the client can still verify the final answer. The undesired locations and functions used by the client are still kept secret to the service being analyzed.

VI. Related Work

There is a lot of work on data security in the cloud. These works ensure data integrity ([11], [12], [13], [14], [15]), i.e., no malicious service or cloud attack changes the client's data or that this can be discovered by the client, respectively. These works assume (similar to our work) that there is an independent auditor. The works [12], [14] discuss privacy issues w.r.t. the auditor. [11] considers data-flow. They do not perform a static data-flow analysis but monitor data-flow between services in order to detect malicious services.

Works on privacy leaks on smart phones are closer to this work [16], [17]. These works analyze whether private data leave smart phone applications. While [16] uses a monitoring approach, [17] uses a static data-flow analysis approach. In contrast to our approach, they analyze the software executed on the smart phone, but they also forbid data leaving the smart phone that are stored in trusted locations.

Song et al. [18] investigates data-flow analysis in the context of service computing. In contrast to our work, they analyze data-flow correctness, e.g., whether each business process implementing a service receives the data it needs.

VII. Conclusions

In this work, we have shown how static data-flow analysis can ensure that clients data don't reach undesired locations (directly or indirectly) via software services. For this, an independent trusted program analysis is required. The approach turns out to be a negotiation approach similar to service-level agreements. The client sends a request to a service candidate whether it can guarantee to avoid data-flow to services on undesired locations. For this, an independent program analysis service signs the analysis result with its private key. Therefore, the client can verify whether a trusted program analysis has being performed. In order to prevent malicious services, the analysis request (in addition to the analysis result) is kept secret by encrypting it with the public key of the certified program analysis.

If we assume that no service is malicious (but possibliy erroneous), then there is no need for keeping the analysis request secret. If a service uses other services, it can look for alternatives that ensure themselves the avoidance of data-flow to undesired locations. In order to keep the analysis request secret to the services being analyzed, the service selection can be performed by the program analysis.

One might argue that a drawback of our approach is that the services must pass their source code to a certified program analysis. However, this certified program analysis is the only service that knows the source code and its the service that can choose the program analysis it trusts.

 ISBN: 978-1-61208-216-5

A more serious problem is that extremely malicious services send a source text to a certified program analysis that differs from their implementation. This could be prevented by two different approaches: first, there might be other program analyzes (e.g., that the service is doing something reasonable) and this analysis requests are also encrypted. If a service does not know what property is being analyzed, it is more difficult to prepare itself for cheating. A second possibility would be a combination of a monitoring approach (e.g., similar to [16]) with a randomized testing approach as used for checking data integrity (see, e.g., [15]): For the latter, the program analyzer can generate test cases (based on the source text it knows) and tests the service using these test cases. The monitoring approach monitors the data leaving the service and their corresponding destination services. Any difference between the data-flow analysis results and the destination services is a hint that the analyzed service is malicious, and the certified program analysis can give a negative answer to the client. It is subject to future work to detail these ideas. To check the performance of the proposed approach, the implementation of a tool is also a subject for future work.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] European Commission and others, "Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data," *Official Journal of the European Communities*, vol. 23, p. 31, 1995.

[2] European Commission, `http://eur-lex.europa.eu/LexUriServ/`, 2012, last accessed May 2012.

[3] R. Seiger, S. Groß, and A. Schill, "Seccsie: A secure cloud storage integrator for enterprises," in *13th IEEE Conference on Commerce and Enterprise Computing (CEC)*. IEEE, 2011, pp. 252–255.

[4] L. Wei, H. Zhu, Z. Cao, W. Jia, and A. Vasilakos, "Seccloud: Bridging secure storage and computation in cloud," in *Distributed Computing Systems Workshops (ICDCSW), 2010 IEEE 30th International Conference on*. IEEE, 2010, pp. 52–61.

[5] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 85–90.

[6] M. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud computing: Distributed internet computing for it and scientific research," *IEEE Internet Computing*, vol. 13, no. 5, pp. 10–13, 2009.

[7] S. Pearson, "Taking account of privacy when designing cloud computing services," in *ICSE Workshop on Software Engineering Challenges of Cloud Computing, 2009*. IEEE, 2009, pp. 44–52.

[8] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[9] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Federation establishment between clever clouds through a saml sso authentication profile," *International Journal on Advances in Internet Technology*, vol. 4, no. 12, pp. 14–27, 2011, ISSN: 1942-2652.

[10] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Evaluating a distributed identity provider trusted network with delegated authentications for cloud federation," in *PROCEEDINGS of The Second International Conference on Cloud Computing, GRIDs, and Virtualization (Cloud Computing 2011)*. IARIA, 2011, pp. 79–85, ISBN: 978-1-61208-153-3.

[11] J. Du, W. Wei, X. Gu, and T. Yu, "Runtest: assuring integrity of dataflow processing in cloud computing infrastructures," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 293–304.

[12] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[13] M. Tribhuwan, V. Bhuyar, and S. Pirzade, "Ensuring data storage security in cloud computing through two-way handshake based on token management," in *2010 International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom)*. IEEE, 2010, pp. 386–389.

[14] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1432–1437, 2011.

[15] Y. Liang, Z. Hao, N. Yu, and B. Liu, "Randtest: Towards more secure and reliable dataflow processing in cloud computing," in *2011 International Conference on Cloud and Service Computing (CSC)*. IEEE, 2011, pp. 180–184.

[16] W. Enck, P. Gilbert, B. Chun, L. Cox, J. Jung, P. McDaniel, and A. Sheth, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*. USENIX Association, 2010, pp. 1–6.

[17] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "Pios: Detecting privacy leaks in ios applications," in *Proceedings of the Network and Distributed System Security Symposium*, 2011.

[18] W. Song, X. Ma, S. Cheung, H. Hu, and J. Lü, "Preserving data flow correctness in process adaptation," in *Services Computing (SCC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 9–16.

 ISBN: 978-1-61208-216-5

A Security Architecture for Cloud Storage Combining Proofs of Retrievability and Fairness

Aiiad Albeshri*†, Colin Boyd* and Juan Gonzalez Nieto*
*Information Security Institute, Queensland University of Technology, Brisbane, Australia.
{c.boyd, j.gonzaleznieto}@qut.edu.au, a.albeshri@student.qut.edu.au
†Faculty of Computing and IT, King Abdulaziz University, Jeddah, Saudi Arabia

***Abstract*—We investigate existing cloud storage schemes and identify limitations in each one based on the security services that they provide. We then propose a new cloud storage architecture that extends CloudProof of Popa et al. to provide availability assurance. This is accomplished by incorporating a proof of storage protocol. As a result, we obtain the first secure storage cloud computing scheme that furnishes all three properties of availability, fairness and freshness.**

Keywords- Cloud Computing; Cloud Storage; Cloud Security.

I. INTRODUCTION

Cloud computing is essentially a large-scale distributed and virtual machine computing infrastructure. This new paradigm delivers a large pool of virtual and dynamically scalable resources, including computational power, storage, hardware platforms and applications, which are made available via Internet technologies. There are many advantages for private and public organisations that decide to migrate all or some of their information services to the cloud computing environment. Examples of these benefits include increased flexibility and budgetary savings through minimisation of hardware and software investments [7], [8], [15]. However, while the benefits of adopting cloud computing are clear, there are also associated critical security and privacy risks that result from placing data off-premises. Indeed, it has been observed that data owners who outsource their data to the cloud also tend to outsource control over their data [8].

Consumers have the option to trade the privacy of their data for the convenience of software services (e.g., web based email and calendars). However, this is generally not applicable in the case of government organisations and commercial enterprises [15]. Such organisations will not consider cloud computing as a viable solution for their ICT needs, unless they can be assured that their data will be protected at least to the same degree that in-house computing offers currently. Yet, none of today's storage service providers in the cloud (e.g., Amazon Simple Storage Service (S3) [2] and Google's BigTable [12]) guarantee any security in their service level agreements. Moreover, there have been already security breach incidents in cloud based services, such as the corruption of Amazon S3, due to an internal failure caused by mismatching files with customers' hashes [1].

This paper focuses on designing a secure storage architecture for cloud computing. As discussed below, important security requirements that a cloud storage architecture should satisfy are confidentiality, integrity, availability, fairness (or non-repudiation) and data freshness. Examination of the literature shows that there is no single complete proposal that provides assurance for all of these security properties. Also, some existing secure cloud storage schemes are designed only for static/archival data and are not suitable for dynamic data.

Proof of storage (POS) protocols are a key component in most secure cloud storage proposals in the literature. A POS is an interactive cryptographic protocol that is executed between clients and storage providers in order to prove to the clients that their data has not been modified or (partially) deleted by the providers [15]. The POS protocol will be executed every time a client wants to verify the integrity of the stored data. A key property of POS protocols is that the size of the information exchanged between client and server is very small and may even be independent of the size of stored data [8].

We investigated different types of existing cloud storage schemes and identified limitations in each one of them based on the security services that they provide. We identified a scheme by Popa *et al.* [18], called CloudProof, as one satisfying the majority of the security requirements. However, it does not provide assurance on data availability, i.e., it does not guarantee that the entire data is indeed stored by the cloud provider. Our goal then is to provide a cloud storage architecture that extends CloudProof in order to provide availability assurance, by incorporating a proof of storage protocol.

The rest of this paper is organised as follows: the second section elucidates the set of security properties that a secure cloud storage application must fulfill; the third section provides an analysis of existing secure cloud storage proposals from the literature; the fourth section introduces the proposed architecture; finally, in the fifth section, the paper draws some conclusions and points at future work.

II. SECURITY REQUIREMENTS

We consider a cloud storage scenario where there are four kinds of parties involved: the data owner, the cloud provider, clients and an optional third party auditor (TPA). The data owner pays for the cloud storage service and sets the access control policies. The cloud provider offers the data storage service for a fee. Clients request and use the data from the cloud. In the cloud environment we assume that there is no mutual trust between parties. Thus, several security properties

 ISBN: 978-1-61208-216-5

need to be assured when storing the data in the cloud, as discussed in many related works (e.g., [16], [22]).

a) Data Confidentiality: ensures that only authorised clients with the appropriate rights and privileges can access the stored information. The most effective way to ensure the confidentiality of the client's data is by using encryption, even though the cloud provider may still be able to predict some information based on monitoring the access patterns of clients [18]. Most existing secure storage proposals provide data confidentiality by allowing clients to encrypt their data before sending it to the cloud. However, critical issues such as key management may be problematic, especially when we have a multiple user scenario.

b) Data Integrity: ensures that the stored data has not been inappropriately modified (whether accidentally or deliberately). Data integrity becomes more challenging when adopting cloud computing where cloud customers outsource their data and have no (or very limited) control over their stored data from being modified by the storage service provider. Thus, cloud customers are aiming to detect any unauthorized modification of their data by the cloud storage provider.

c) Data Availability: ensures that users are able to obtain their data from the cloud provider when they need it. Cloud customers want to be sure that their data is always available at the cloud storage. To this end, a number of proof of storage protocols have been devised that allow the cloud provider to prove to clients that their entire data is being stored, which implies that the data has not been deleted or modified. Section III discusses some of these schemes.

d) Public Verifiability: means that service providers allow a TPA to perform periodical availability verifications on behalf of their customers. In cloud computing environments, customers may need to allow a TPA to verify the integrity of the dynamic data stored in the cloud storage [21]. Public verifiability allows the cloud customers (or their TPA) to challenge the cloud server for correctness of stored data. In fact, security requirements can be inter-related. For instance, when a TPA is delegated to perform verification, the confidentiality may be compromised. However, this issue could be resolved by utilising a verification protocol that allows TPA to verify without knowing the stored data [21].

e) Freshness: ensures that the retrieved data is fresh, i.e., it contains the last updates to the data. This is very important in shared and dynamic environments where multiple clients may simultaneously update data. Cloud customers need to ensure that the retrieved data is the latest version. To the best of our knowledge, CloudProof [18] is the only cloud storage scheme that addresses freshness.

f) Fairness: or non-repudiation ensures that a dishonest party cannot accuse an honest party of manipulating its data [24]. If a dispute arises between a client and storage provider regarding whether the correct data is stored then it may be necessary to invoke a judge to decide who is right. Fairness will typically be implemented by using digital signatures. Clients may want to have a signature from the provider acknowledging what data is stored. Providers may want signatures from clients whenever the stored data is altered, with deletion being an important special case.

III. Proof of Storage Schemes (POS)

Cloud storage schemes can be categorised into two types, static and dynamic. In static schemes, clients store their data and never change or update it. In dynamic schemes clients can update the stored data. In the following two subsections, we review existing proposals for POS protocols. Table I lists the schemes reviewed and indicates the security requirements that are satisfied by them. The entry with the dagger (†) indicates that the property is only partially satisfied. It can be seen that no single proposal encompasses all security requirements identified in Section II. The security requirements in the table are Confidentiality (C), Integrity (I), Availability (A), Public Verifiability (PV), Freshness (Fr) and Fairness (Fa).

Table I
Overview of the prominent proof of storage (POS) schemes.

POS Scheme	C	I	A	PV	Fr	Fa	Type
Proof of Retrievability (POR) [13]	✓	✓	✓	✓	✗	✗	Static
Provable Data Possession (PDP)[3]	✓	✓	✓	✓	✗	✗	Static
Compact POR [19]	✓	✓	✓	✓	✗	✗	Static
Tahoe [23]	✓	✓	✗	✗	✗	✗	Static
HAIL [5]	✗	✗	✓	✓	✗	✗	Static
POR (experimental test) [6]	✓	✓	✓	✓	✗	✗	Static
Framework for POR protocols [9]	✓	✓	✓	✓	✗	✗	Static
POS from HIP [4]	✓	✓	✓	✓	✗	✗	Static
DPDP [10]	✓	✓	✓	✗	✗	✗	Dynamic
POR with public verifiability [21]	✓	✓	✓	✓	✗	✗	Dynamic
Depot [17]	✗	✓	✓	✓	✗	✗	Dynamic
Wang *et al.* [20]	✗	✓	✓	✓	✗	✗	Dynamic
CloudProof [18]	✓	✓	✗	✓	✓	✓	Dynamic
Fair and Dynamic POR [24]	✓	✓	✓	✗	✗	✗†	Dynamic

A. POS for Static Data

There are several POS schemes that support storage of static data. Juels and Kaliski [13] introduced proof of retrievability (POR). In POR the *Encode* algorithm firstly encrypts all the data. Additionally, a number of random-valued blocks (sentinels) are inserted at randomly chosen positions within the encrypted data. Finally, an error correction code is applied to the resulting new data. Clients *challenge* the service provider by identifying the positions of a subset of sentinels and asking the service provider to retrieve the requested values. The *VerifyProof* process works because, with high probability, if the service provider modifies any portions of the data, the modification will include some of the sentinels and will therefore be detected. If the damage is so small that it does not affect any sentinel, then it can be reversed using error correction.

POR [13] only allows a limited number of executions of the *Challenge* algorithm (for the whole data). The verification capability of POR is limited by the number of precomputed sentinels embedded into the encoded file. This is improved by

 ISBN: 978-1-61208-216-5

the scheme of Shacham and Waters [19], which enables an unlimited number of queries and requires less communication overhead. In this scheme, in addition to encoding each file block, the client appends a special type of authenticator to each block. The encoded blocks and authenticators are stored on the server. The verifier challenges the service provider by sending a set of randomly selected block indexes. The response from the service provider is a compact proof that combines the challenge blocks and authenticators and which can be validated very efficiently by the verifier. Likewise, Bowers *et al.* [6], Ateniese *et al.* [4] and Dodis *et al.* [9] provided POR schemes which provide probabilistic assurance that a remotely stored file remains intact.

Table I lists other prominent POS examples. All POS schemes mentioned above were designed to deal with static or archival data only and are not suitable for dynamic environments. The efficiency of these schemes is mainly based on the preprocessing of the data before sending it to remote storage. Any modification to the data requires re-encoding the whole data file, so it has associated a significant computation and communication overhead.

B. POS for Dynamic Data

It is natural that clients may want to update their files while they are in storage without having to resubmit the whole data set to the server. Therefore, it is desirable to offer an option to update files in such a way that the proof of storage for the whole data still applies. POS for dynamic data is more challenging than static data. There are several dynamic POS schemes. Erway *et al.* [10] introduced what they called "Dynamic Provable Data Possession" or DPDP, which extends the static PDP [3]. Their approach uses a variant of authenticated dictionaries, which allows insertion and deletion of blocks within the data structure. A limitation of DPDP is that it does not allow for public verifiability of the stored data; in addition it does not consider data freshness or fairness. Wang *et al.* [21] improve on DPDP by adding public verifiability, thus allowing a TPA to verify the integrity of the dynamic data storage. Now the authenticated data structure employed is the classic Merkle Hash Tree (MHT). Still, data freshness and fairness are not considered.

Popa *et al.* [18] introduced CloudProof, which provides fairness by allowing customers to detect and prove cloud misbehaviour. This is achieved by means of digitally signed attestations. Each request and response for reading (get) and writing (put) data is associated with an attestation. This attestation will be used as proof of any misbehaviour from both sides. CloudProof [18] is the only POS scheme that provides assurance of data freshness by using hash chains. For each put and get attestation, the hash chain is computed over the hash of the data in the current attestation and the hash value of the previous attestation. More details are provided in Section IV.

In addition, CloudProof emphasises the importance of "fairness". If the cloud misbehaves, for example it deletes some user blocks, then the owner has the ability to prove to a judge that the cloud was at fault. At the same time, if the owner claims falsely that a file was deleted, the cloud can prove to the judge that the owner asked for this to be done.

It should be noted that fairness in CloudProof does not extend to the meaning normally expected in protocols for fair exchange. In particular, Feng *et al.* [11] have pointed out that a provider could omit sending its signature once it has received the signature of the client on an update. Consequently the provider has an "advantage" in the sense that it can prove to a judge that the client asked for an update but the client cannot provide any evidence that the provider received the update request. Arguably this advantage has limited consequences because the client can retain the update details pending the receipt of the provider's signature. If the provider does not send the signature then this is inconvenient for the client but he can recover from it; meanwhile, the client can seek other remedies. In any case, ensuring fairness in the stronger sense that neither party ever gets an advantage can only be achieved in general using an online trusted third party which is likely to be too costly to justify.

Zheng and Xu [24] have a rather different definition of fairness for their dynamic scheme. They require only that clients are not able to find two different files which both will satisfy the update protocol. The idea is that a malicious client can then produce a different file from that which the server can produce and claim that the server altered the file without authority. Zheng and Xu do not require that the update protocol outputs a publicly verifiable signature so a judge can only verify this fact by interacting with the client using public information. In addition, they do not consider the situation where a server does maliciously alter the file - for example deletes it. In this case, the client may no longer have anything to input to the verification equation.

In fact, the security model for CloudProof is quite weak. Auditing is only done on a probabilistic basis to save on processing. The data owner (or TPA) assigns to each block some probability of being audited, so an audit need not check every block. Thus, for parts that are rarely touched by users, this means that it could be a long time before it is noticed if something has been deleted. Whether or not a block will be audited is known to any user who has access to it, but is hidden from the cloud. Blocks which are not audited can be changed at will (or deleted) by the cloud. Popa *et al.* [18] state that "We do not try to prevent against users informing the cloud of when a block should be audited (and thus, the cloud misbehaves only when a block is not to be audited)". This seems too optimistic - if even a single user can be corrupted by the cloud, then the cloud can delete all the blocks to which that user has access without any chance of detection. it is clear therefore that CloudProof does not provide the availability assurance. However, as seen in Table I, it is the scheme that provides the most security services. In the next section, we extend CloudProof to provide availability of the whole stored data. We do so by combining CloudProof with the dynamic POR of Wang *et al.* [21].

IV. Proposed Architecture

We now describe a new architecture which combines the idea of CloudProof [18] and Dynamic Proofs Of Retrievability (DPOR) [21] as it provides data availability for dynamic data along with must of other security requirements. The proposed

 ISBN: 978-1-61208-216-5

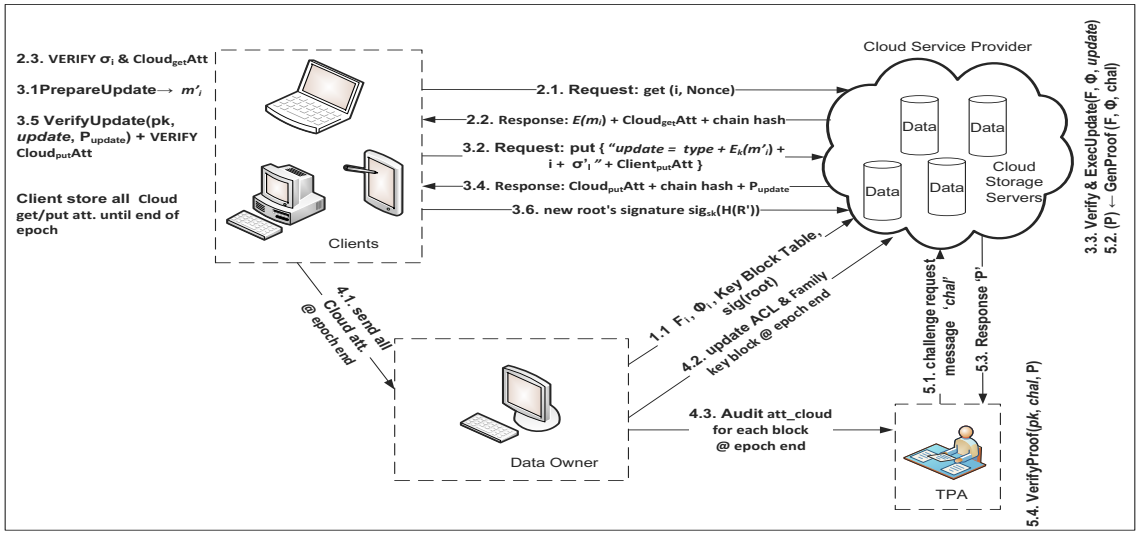


Figure 1. Proposed Architecture

POS architecture tackles the limitations of both schemes. Thus, the limitation of CloudProof of being unable to check data availability at the whole data set level is overcome by employing DPOR.

DPOR consists of the following protocols/algorithms:

1) *KeyGen*: is a randomized algorithm that is used to generate cryptographic key material.
2) *Encode*: is used for encoding data before sending it to the remote storage.
3) *GenProof*: the service responds to the client's challenge request by generating a proof which is sent to the verifier.
4) *VerifyProof*: upon receiving the proof from the service provider, the client executes this protocol to verify the validity of the proof.
5) *ExecUpdate*: this protocol is used in dynamic schemes and is executed by the cloud provider. This protocol may include a proof by the service provider of the successful update of the data, so that the customer can verify the update process.
6) *VerifyUpdate*: this is executed by the client in order to verify the proof sent by the service provider after an update.

As in CloudProof, we consider different time periods or *epochs*. At end of each epoch the data owner or TPA performs a verification process to assure that the cloud storage possesses its data. In this way we obtain a design that satisfies all the desirable properties discussed in Section II. Figure IV describes the proposed architecture and identifies its parties and the different protocols that are executed between them.

g) Key Management: we assume that the data owner will divide the plaintext data file into blocks $F'' = \{m''_1, m''_2, ..., m''_n\}$. Each data block is assigned to an ACL (set of users and groups) and blocks with similar ACL are grouped in a single block family. In addition, for each block family there is a family key block that contains a secret (signing) key sk (known only to clients with write access in the ACL), read access key k (known only to clients with read access in the ACL), public (verification) key pk (known to all clients and the cloud provider), version of pk and k keys, block version, and signature of the owner. The data owner will create the family key block table in which each row in this table corresponds to an ACL (Fig. 2). The data owner maintains the key production while the key distribution process is offloaded to the cloud service provider but in a verifiable way. The key distribution process involves two cryptographic tools; **broadcast encryption** E_F which is used to encrypt the secret key ($E_F(sk)$) and read access key ($E_F(k)$). $E_F(k)$ guarantees that only allowed clients and groups in the ACL's read set can decrypt the key and use it to decrypt the blocks in the corresponding family. sk is used to generate update signatures for blocks. $E_F(sk)$ guarantees that only users and groups in the ACL's write set can decrypt the key and use it to generate update signatures for blocks in the corresponding family. The **key rotation** scheme is another cryptographic tool which is used to generate a sequence of keys using an initial key with a secret master key [14]. Thus, only the owner of the secret master key can produce the next key in the sequence. Also, by using key rotation, the updated key allows computing of old keys. Thus, there is no need to re-encrypt all encrypted data blocks [18]. The data owner will keep the family key block table and every time there is a change of membership, the data owner will re-encrypt the key and update the family key block table.

h) Pre-Storage Processing: the data owner encodes each block in the data file F'' using Reed-Solomon error correction $F' = encode_{RS}(F'')$. Then, each block in F' is encrypted using the corresponding k of that block family; $F = E_k(F') = \{m_1, m_2, ..., m_n\}$. The data owner creates a Merkle Hash Tree (MHT) for each block family. The MHT is constructed as a binary tree that consists of a root R and leaf nodes which are an ordered set of hashes of the family data blocks $H(m_i)$. MHT is used to authenticate the values of the data blocks. As in DPOR [21], the leaf nodes are treated in the left-to-right sequence thus, any data block (node) can be uniquely identified by following this sequence up to the root (Fig. 4).

 ISBN: 978-1-61208-216-5

DPOR [21] uses BLS or RSA in such a way that multiple signatures verification can be done very efficiently. Thus, for each block family F, the data owner runs the signature generatour algorithm $(\Phi,\ sig_{sk}(H(R))) \longleftarrow SigGen(sk, F)$ which takes the signing key of the family (sk) and the encrypted block family F and generates the signature set for this family $\Phi = \{\sigma_1, \sigma_2, ..., \sigma_n\}$; where $\sigma_i \leftarrow (H(m_i) \cdot u^{m_i})^{sk}$ for each family block m_i; $u \leftarrow G$ is a random element choosed by the data owner. In addition, a signature of the root that associated MHT is generated $sig_{sk}(H(R))$. Then, each block m_i will be associated with its signature σ_i and some metadata such as block version and version of k and pk; $b_i = \{m_i||block\,Ver||k\,Ver||pk\,Ver||\sigma_i\}$ (Fig. 2). Finally, the data owner sends to the cloud storage the block family $\{b_1, b_2, ..., b_n\}$, its signature set Φ, the family key block table, and the root signature of this block family $sig_{sk}(H(R))$ (Message 1.1 of Fig. IV).

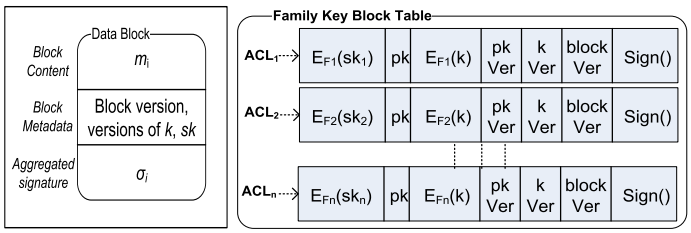


Figure 2. Data block and family key block table sent to the cloud

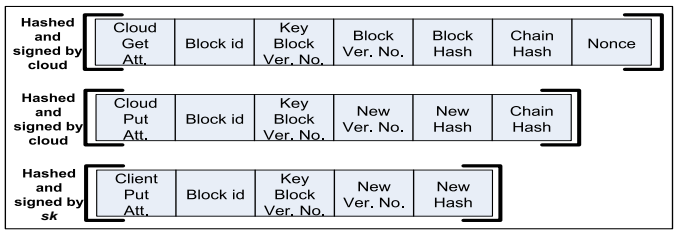


Figure 3. Attestations of Popa *et al.* [18]

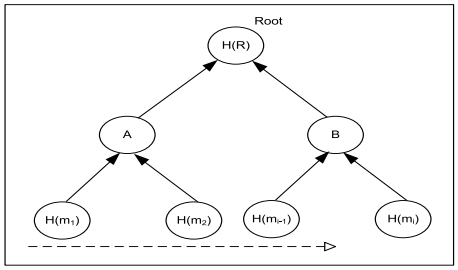


Figure 4. Merkle Hash Tree

i) Attestations: As in CloudProof [18] we build a hash chain from all data changes and requires signing from both parties on all updates. Thus, any misbehaviour could be detected and proved by exchanging attestations for each request or response between data owner, clients and cloud provider. The structure of exchanged attestations includes metadata such as the block version and current hash which are used to maintain the write-serialisability (when each client placing an update is aware of the latest committed update to the same block) and the hash chain value which is used for freshness (Fig. 3). The hash chain is computed over the hash of the data in the current attestation and the chain hash of the previous attestation. Thus it is a sequence of hashes which contains current attestation and all history of attestations of a specific block as follows: *chain hash* = *hash*(*data, previous hash chain value*). Thus, if the sequence of attestations is broken this means there is a violation of freshness property. In addition, during each epoch clients need to locally store all received attestations and forward them to the data owner for auditing purposes at end of each epoch (Fig. IV). For simplicity, in our proposal we assume that all data blocks will be audited, however, in practice a probabilistic approach as in CloudProof would be advantageous.

j) Get block: in the get (read) request for a specific data block, clients need to send to the cloud provider the block index (i) for that block and a random nonce (Message 2.1 of Fig. IV). The cloud provider will verify the client by checking the ACL and make sure that only clients with *read/access permission* (of the block) can gain access to this block. If the client is authorised then it will respond by sending the requested block (b_i) with its signature (σ_i), the cloud get attestation $Cloud_{get}Att$ and signature of the attestation $Sign(Cloud_{get}Att)$ (Message 2.2 of Fig. IV). The client will verify the retrieved attestation and make sure that it was computed over the data in the block and the nonce. Also, the client will verify the integrity signature (σ_i) of the received block. Clients need to locally store these attestations and their signatures and forward them at the end of each epoch for auditing purposes.

k) Put block: suppose the client wants to update a specific block (m_i) into (m'_i). First, the client needs to generate the corresponding signature σ'_i. Also, the client prepares the update (put) request message $update = (type,\ i,\ m'_i,\ \sigma'_i)$; where *type* denotes the type of update (Modify M, Insert I or Delete D). In addition, the client will use sk to compute its put attestation ($Client_{put}Att$) and sign it $sign_{sk}(Client_{put}Att)$. Then client sends $update$ message, $Client_{put}Att$ and $sign_{sk}(Client_{put}Att)$ to the cloud servers (Message 3.2 of Fig. IV). On the cloud side, cloud provider will verify the client by checking the ACL and make sure that only clients with *write permission* (of the block) can update this block. In addition, cloud provider will verify the client's attestation. If the client is authorised then it runs $(F',\ \Phi',\ P_{update}) \leftarrow ExecUpdate(F,\ \Phi,\ update)$ which replaces the block m_i with m'_i and generates the new block family F'; and replaces the signature$\sigma_i$with σ'_i and generates new signature set of the family Φ'; and updates the $H(m_i)$ with $H(m'_i)$ in the MHT and generates the new root R' (in MHT scheme as a new block added into or deleted from a file these new nodes are added to MHT as described in DPOR [21] and the tree is rearranged according to this update). The cloud responds to the update request by sending a proof for the successful update ($P_{update} = \{\Omega_i,\ H(m_i),\ sig_{sk}(H(R)),\ R'\}$; where Ω_i is used for authentication of m_i). Also, the cloud constructs the put attestation ($Cloud_{put}Att$) and signs it $sign_{sk}(Cloud_{put}Att)$ and send

them to the client (Messages 3.3 and 3.4 of Fig. IV). In addition, the cloud provider will store the received client attestations to be used if any misbehaviour detected. The client verifies the cloud put attestation and check the chain hash. Also, client verify the received update proof by running this algorithm: $\{(TRUE,\ sig_{sk}(H(R'))),\ FALSE\} \leftarrow VerifyUpdate(pk,\ update,\ P_{update})$ which takes pk, the old root's signature $sig_{sk}(H(R))$, the update message request ($update$), and the received proof (P_{update}). If verification succeeds, it generates the new root's signature $sig_{sk}(H(R'))$ for the new root $R'$and send it back to the cloud (Messages 3.5 and 3.6 of Fig. IV). In addition, client need to store all received cloud put attestation ($Cloud_{put}Att$) and forward them to the data owner for auditing purposes.

l) Auditing: the auditing process is carried out at the end of each epoch and consists of two parts. In the first part the attestations produced within the given epoch are verified as per CloudProof. In the second part, the integrity of the whole data set as in DPOR [21]. For each family block the TPA picks random c-element subset $I = s_1, ..., s_c$. For each $i \in I$, the TPA selects a random element $v_i \leftarrow Z$. Then TPA sends the message $chal$ which identifies which blocks to be checked ($chal = \{(i, v_i)\}_{s_1 \leq i \leq s_c}$). When the cloud provider receives the $chal$ message, prover will compute: 1. $\mu = \sum_{i=s_1}^{s_c} v_i m_i \in Z$; and 2. $\sigma = \prod_{i=s_1}^{s_c} \sigma_i^{v_i} \in G$. The prover runs $P \leftarrow GenProof(F, \Phi, , chal)$ algorithm to generate the proof of integrity $P = \{\mu, \sigma, \{H(m_i), \Omega_i\}_{s_1 \leq i \leq s_c}, sig_{sk}(H(R))\}$; where Ω_i is the node siblings on the path from the leave i to the root *R* in the MHT. The verifier will verify the received proof by running this algorithm $\{TRUE, FALSE\} \leftarrow VerifyProof(pk,\ chal,\ P)$. This way we are able to check data availability at the whole file level.

V. Conclusion and Future Work

We have investigated the different type of existing cloud storage schemes and identified limitations in each one of them based on the security services that they provide. We have then introduced a cloud storage architecture that extends CloudProof in order to provide availability assurance. This is accomplished by incorporating a proof of storage protocol such as DPOR. The proposed POS architecture overcomes the weaknesses of both schemes. In this way we obtain a design that satisfies all the identified desirable security properties.

Both schemes are considered secure and work efficiently individually and it is reasonable to assume that they should work in a secure and an efficient way when combined. However, it may be interesting to perform a detail performance and security analysis of the proposed architecture.

References

[1] Amazon S3 availability event: July 20, 2008. http://status.aws.amazon.com/s3-20080720.html. [retrieved: Feb, 2012].

[2] Amazon Web Services. Amazon simple storage service FAQs, Mar 2011. Available at: http://aws.amazon.com/s3/faqs. [retrieved: Dec, 2011].

[3] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 598–609, New York, NY, USA, 2007. ACM.

[4] Giuseppe Ateniese, Seny Kamara, and Jonathan Katz. Proofs of storage from homomorphic identification protocols. In *Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '09, pages 319–333, Berlin, Heidelberg, 2009. Springer-Verlag.

[5] K.D. Bowers, A. Juels, and A. Oprea. HAIL: A high-availability and integrity layer for cloud storage. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 187–198. ACM, 2009.

[6] Kevin D. Bowers, Ari Juels, and Alina Oprea. Proofs of retrievability: theory and implementation. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, CCSW '09, pages 43–54, New York, NY, USA, 2009. ACM.

[7] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616, 2009.

[8] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina. Controlling data in the cloud: outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 85–90. ACM, 2009.

[9] Yevgeniy Dodis, Salil Vadhan, and Daniel Wichs. Proofs of retrievability via hardness amplification. In *Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, TCC '09, pages 109–127, Berlin, Heidelberg, 2009. Springer-Verlag.

[10] Chris Erway, Alptekin Küpçü, Charalampos Papamanthou, and Roberto Tamassia. Dynamic provable data possession. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 213–222, New York, NY, USA, 2009. ACM.

[11] J. Feng, Y. Chen, D. Summerville, W.S. Ku, and Z. Su. Enhancing Cloud Storage Security against Roll-back Attacks with A New Fair Multi-Party Non-Repudiation Protocol. In *The 8th IEEE Consumer Communications & Networking Conference*, 2010.

[12] Google. Security and privacy FAQs, Mar 2011. Available at: http://aws.amazon.com/s3/faqs. [retrieved: Jan, 2012].

[13] Ari Juels and Burton S. Kaliski, Jr. PORs: proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 584–597, New York, NY, USA, 2007. ACM.

[14] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 29–42, 2003.

[15] Seny Kamara and Kristin Lauter. Cryptographic cloud storage. In Radu Sion, Reza Curtmola, Sven Dietrich, Aggelos Kiayias, Josep Miret, Kazue Sako, and Francesc Sebé, editors, *Financial Cryptography and Data Security*, volume 6054 of *Lecture Notes in Computer Science*, pages 136–149. Springer Berlin / Heidelberg, 2010.

[16] R.L. Krutz and R.D. Vines. *Cloud Security: A Comprehensive Guide to Secure Cloud Computing*. Wiley, 2010.

[17] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish. Depot: Cloud storage with minimal trust. In *Proc. OSDI*, 2010.

[18] R.A. Popa, J.R. Lorch, D. Molnar, H.J. Wang, and L. Zhuang. Enabling security in cloud storage slas with cloudproof. *Microsoft TechReport MSR-TR-2010*, 46:1–12, 2010.

[19] Hovav Shacham and Brent Waters. Compact proofs of retrievability. In *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '08, pages 90–107, Berlin, Heidelberg, 2008. Springer-Verlag.

[20] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Ensuring data storage security in cloud computing. In *Cloud Computing*, pages 1 –9, july 2009.

[21] Qian Wang, Cong Wang, Jin Li, Kui Ren, and Wenjing Lou. Enabling public verifiability and data dynamics for storage security in cloud computing. In *Proceedings of the 14th European conference on Research in computer security*, ESORICS'09, pages 355–370, Berlin, Heidelberg, 2009. Springer-Verlag.

[22] M.E. Whitman and H.J. Mattord. *Principles of Information Security*. Course Technology Ptr, 3rd edition, 2009.

[23] Z. Wilcox-O'Hearn and B. Warner. Tahoe: the least-authority filesystem. In *Proceedings of the 4th ACM international workshop on Storage security and survivability*, pages 21–26. ACM, 2008.

[24] Q. Zheng and S. Xu. Fair and dynamic proofs of retrievability. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 237–248. ACM, 2011.

 ISBN: 978-1-61208-216-5

The Optimal Resource Allocation Among Virtual Machines in Cloud Computing

Marjan Gusev
Faculty of Information Sciences and Computer Engineering
Ss. Cyril and Methodius University
Skopje, Macedonia
Email: marjan.gushev@finki.ukim.mk

Sasko Ristov
Faculty of Information Sciences and Computer Engineering
Ss. Cyril and Methodius University
Skopje, Macedonia
Email: sashko.ristov@finki.ukim.mk

***Abstract*—Virtualization is a key technology for multi-tenant cloud computing enabling isolation of tenants in one or more instances of virtual machines and sharing the hardware resources. In reality, modern multi-core multiprocessors also share the last level cache among all cores on one chip. Our goal will be to enable an optimal resource allocation by avoiding cache misses as much as possible, since this will lead to performance increase. In this paper, we analyze the performance of single and multi-tenant environments in cloud environment installed on a single chip multi core multiprocessor with different resource allocation to the tenants. We realize a series of experiments with matrix multiplication as compute intensive and memory demanding algorithm by varying the matrix size to analyze performance behavior upon different workload and variable cache requirements. Each experiment uses the same resources but it is orchestrated differently. Although one might think that virtualization and clouds include software overhead, the results show how and when cloud computing can achieve even better performance than traditional environment, both in a single-tenant and multi-tenant resource allocation for certain workload. The conclusions show that there are regions where the best performance in the cloud environment is achieved for cache intensive algorithms allocating the resources among many concurrent instances of virtual machines rather than in traditional multiprocessors using OpenMP.**

***Keywords-Cache memory*; *Cloud Computing*; *Matrix Multiplication*; *Shared Memory*; *Virtualization*.**

I. INTRODUCTION

Cloud Service Providers (CSPs) rent on-demand scalable hardware resources. The customers can use CPU, memory, and storage with arbitrary size and type in virtual machines (VMs) whenever they need. This flexibility results in dynamic resource workload. CSPs foster it even more by consolidating VMs on smaller number of physical servers in order to save power consumption. In such dynamic environment, customers' VMs are not totally isolated. They share same physical resources, especially CPU, memory and network. This paper focuses on CPU utilization when sharing among many concurrent VMs.

Cache memory is the CPU's key element in compute and memory intensive algorithms. Due to the performance impact of the cache, we define these algorithms as *cache intensive algorithms*. Matrix multiplication is an example of such algorithm that today's computations are using. This algorithm is compute intensive $O(n^3)$ and memory demanding $O(n^2)$.

Producers of modern multiprocessors must adopt caches for cloud computing especially in the multitenant, multiprocess and multithreading dynamic environment. For example, Intel introduces Intel Smart Cache [1] to improve the performance. Sharing the last level cache among multiprocessor's cores allows each core dynamically use the cache up to 100%. This technology can be used to increase the overall performance in cloud computing multi-tenant environment. Machina and Sodan in [2] developed a model that describes the performance of the applications as a function of allocated cache size, even if the cache is dynamically partitioned.

The fundamental driver for Multi-tenancy is Virtualization. It introduces additional layer and can provide better performance. The cache intensive algorithms run faster in distributed than shared cache memory virtual environment. Gusev and Ristov in [3] found that matrix multiplication algorithm can run faster in virtual environment compared to traditional, both by sequential and parallel executions (for problem sizes that fit in distributed L1 and L2 caches correspondingly). However, virtualization produces huge performance drawback for shared cache memory, even if it is dedicated per chip in multi chip multiprocessor. In this paper, we continue the performance analysis in cloud solution, compared to both virtual environment in guest operating system and traditional operating system. We expect that there are regions where the experiments will prove that cloud virtualization produces better performance and achieves better performance.

Koh et al. [4] describe the phenomenon that running the same VM on the same hardware at different times among the other active VMs will not achieve the same performance. They predict the performance scores of the applications under performance interference in virtual environments. VM granularity has a significant effect on the workload's performance for small network workload [5].

The experiments performed in this paper address several VM instances in a cloud system using different number of CPUs (assuming all cores are utilized). The introduction of a virtualization in the cloud is supposed to decrease the performance [6]. Our plan is to check validity of the

 ISBN: 978-1-61208-216-5

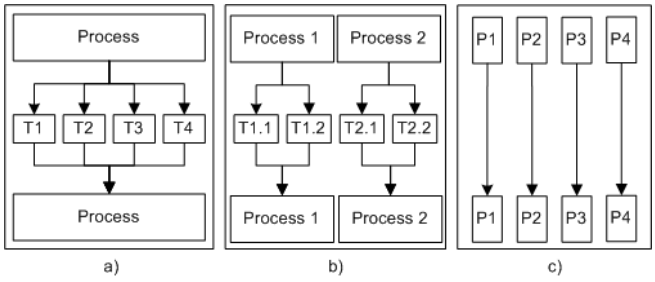


Figure 1. Test Cases in Traditional Environment

following hypotheses:

- Is there a region where cloud environment achieves better performance than traditional and virtual environment, and
- What is the performance of cloud computing with multi-VM environment in comparison to allocation of all resources to only one VM?

The rest of the paper is organized as follows: The testbed for three workload environments is described in Section II. Sections III and IV present the results of the experiments performed to determine the best environment for cache intensive algorithm and best resource allocation among process, threads and tenants correspondingly, while Section V presents the performance when the algorithm is executed sequentially on a single core. The results of the cache misses analysis are presented in Section VI to prove the causes for better / worse performance in L2 / L3 region for traditional and cloud environment. The final Section VII is devoted to conclusion and future work.

II. The Workload Environments

This section describes the testing methodology and defines the workload environments for experiments. Matrix multiplication algorithm is used as test data for both sequential and parallel execution. For all different environments, we plan to use the same hardware and operating system. The only difference is inclusion of virtual machines and enabling cloud environment.

A. Traditional Environment

This environment consists of Linux Ubuntu Server 11.04 installed on Dell Optiplex 760 with 4GB DDR2 RAM and Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz [7]. The multiprocessor has 4 cores, each with 32 KB 8-way set associative L1 cache dedicated per core and 8-way set associative L2 cache with total 6 MB shared by 3MB per two cores.

Three different parallel executions are defined as test cases 1.1, 1.2 and 1.3 in this environment, as depicted in Fig. 1. The sequential execution is determined as test case 1.4.

1) Case 1.1: 1 process with 4 (max) threads on total 4 cores: In this test case the matrix multiplication is executed by one process using 4 parallel threads as presented in Fig. 1 a). Each thread runs on one core multiplying the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.

2) Case 1.2: 2 different processes with 2 threads per process on total 4 cores: In this test case two concurrent processes execute matrix multiplication. Each process uses two parallel threads as shown in Fig. 1 b). Each process multiplies the whole matrix $A_{N \cdot N}$ and a half of matrix $B_{N \cdot N/2}$ divided vertically. Each thread multiplies matrix $A_{N \cdot N}$ and half of $B_{N \cdot N/2}$, i.e., $B_{N \cdot N/4}$.

3) Case 1.3: 4 different processes with 1 thread per process (sequentially) on total 4 cores: In this test case 4 concurrent processes execute matrix multiplication as depicted in Fig. 1 c). Each process multiplies the whole matrix $A_{N \cdot N}$ and a quarter of matrix $B_{N \cdot N/4}$ divided vertically.

4) Case 1.4: 1 process sequentially on 1 core: In this test case, one process executes matrix multiplication sequentially on one core, i.e., three cores are unused and free. The process runs on one core multiplying the whole matrix $A_{N \cdot N}$ with the whole matrix $B_{N \cdot N}$.

B. Virtual Environment

This environment consists of the same hardware and operating system as described in Section II-A. Additionally new VM is installed with same Linux Ubuntu Server 11.04 using VirtualBox and Kernel-based Virtual Machine virtualization standard (KVM). All available resources (4 cores) are allocated to the only one VM for parallel execution and only one core for sequential execution.

Two test cases are performed in this environment one with parallel and the other with sequential execution.

1) Case 2.1: 1 VM with 1 process with 4 (max) threads on total 4 cores: In this test case one process executes matrix multiplication by 4 parallel threads, all in the VM. Each thread runs on one core multiplying the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.

2) Case 2.2: 1 VM with 1 process sequentially on total 1 core: In this test case one process executes matrix multiplication sequentially in VM on one core, i.e., three cores are unused and free. The process runs on one core multiplying the whole matrix $A_{N \cdot N}$ with the whole matrix $B_{N \cdot N}$.

C. Cloud Virtual Environment

Cloud virtual environment is developed using OpenStack Compute project [8] deployed in dual node as depicted in Fig. 2. KVM virtualization standard is also used for VMs. One Controller Node and one Compute Node are used.

This cloud virtual environment consists of the same hardware and operating system as described in Section II-A for Compute Node server. Virtual Machine described in Section II-B is instantiated in one or more instances for the four test cases that are performed in this environment.

 ISBN: 978-1-61208-216-5

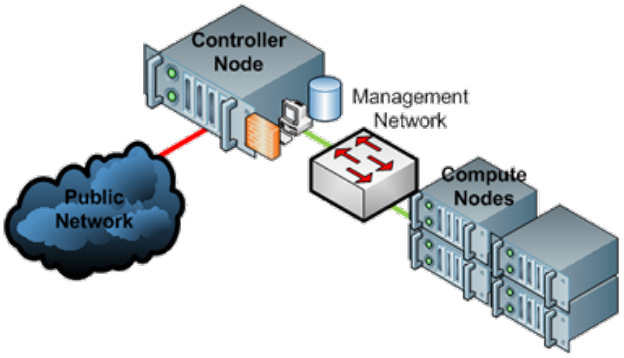


Figure 2. OpenStack dual node deployment [9]

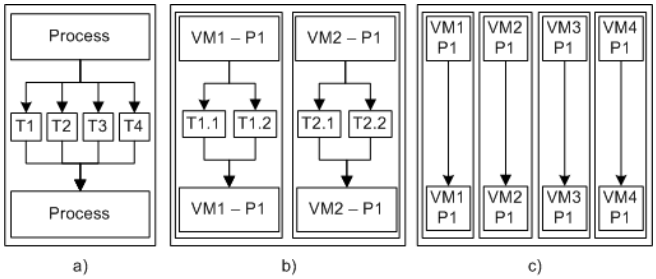


Figure 3. Test Cases in Cloud Virtual Environment

Three test cases 3.1, 3.2 and 3.3 are performed as parallel executions in this environment depicted in Fig. 3. The test case 3.4 for sequential execution is defined as one instance of VM with one sequential process.

1) Case 3.1: 1 instance of VM with 1 process with 4 (max) threads per process on total 4 cores: This case is similar as cases 1.1 and 2.1, i.e., one instance of VM is activated in the Cloud allocated with all 4 cores as depicted in Fig. 3 a). One process in VM executes matrix multiplication with 4 paralllel threads. Each thread runs on one core multiplying the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.

2) Case 3.2: 2 concurrent instances of VM with 1 process per VM with 2 threads per process on total 4 cores: In this test case two concurrent instances of same VM are activated in the Cloud allocated with 2 cores per instance as depicted in Fig. 3 b). One process in each VM executes matrix multiplication concurrently with 2 parallel threads per process (VM). Each process (in separate VM) multiplies the whole matrix $A_{N \cdot N}$ and a half of matrix $B_{N \cdot N/2}$ divided vertically. Each thread multiplies matrix $A_{N \cdot N}$ and half of $B_{N \cdot N/2}$, i.e., $B_{N \cdot N/4}$.

3) Case 3.3: 4 concurrent instances of VM with 1 process per VM with 1 thread per process (sequentially) on total 4 cores: In this test case, 4 concurrent instances of same VM are activated in the Cloud allocated with 1 core per instance as depicted in Fig. 3 c). Each process (in separate VM) multiplies the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.

4) Case 3.4: 1 instance of VM with 1 process sequentially on total 1 core: This case is similar as test case 3.1. The difference is that only one core is dedicated to the only VM, i.e., three cores are unused and free. The process runs on one core multiplying the whole matrix $A_{N \cdot N}$ with the whole matrix $B_{N \cdot N}$.

D. Test Goals

The test experiments have two goals:

- The first goal is to determine if the additional virtualization layer in cloud drawbacks the performances compared to traditional or virtualized operating system when all the resources are dedicated to only one tenant and multi-threading is used.
- The second goal is to determine which resource allocation among tenants and threads provides best performance in the traditional environment and in the cloud.

Different sets of experiments are performed by varying the matrix size changing the processor workload and cache occupancy in the matrix multiplication algorithm.

III. Traditional vs Virtual vs Cloud Environment Performance with all Resources Allocated

This Section presents the results of the experiments performed on three workload environments when all the resources (CPU cores) are rented to one tenant, i.e., test cases 1.1, 2.1 and 3.1 as described in Section II.

Fig. 4 depicts the speed in gigaflops that matrix multiplication achieves for different matrix size N when executing one process concurrently using 4 threads on 4 cores on three same hardware resources, but different system environments as described in Section II. The curves are identified by V(4)T for traditional environment, V(4)V for environment with virtual and V(4)C with cloud environment. Fig. 5 shows only the differences of achieved speeds in Fig. 4 using relative presentation of the ratio to the default speed value obtained by traditional environment.

Two regions with different performance for all three test cases are clearly depicted in Fig. 4; the left one with higher speed and the right one with lower speed. The first region is the L2 region as defined in [3] (the region for such matrix size N that will enable storage of all memory requirements in L2 cache and avoid generation of cache misses for reusing the same data on L2 level). The second region is the region where the matrices can not be stored completely in the L2 cache and many L2 cache misses will be generated due to re-using of data, but memory requirements will fit in the L3 cache (if it exists). This region is called the L3 region. We must note that those matrices that fit in L1 region are too small to produce higher speed.

 ISBN: 978-1-61208-216-5

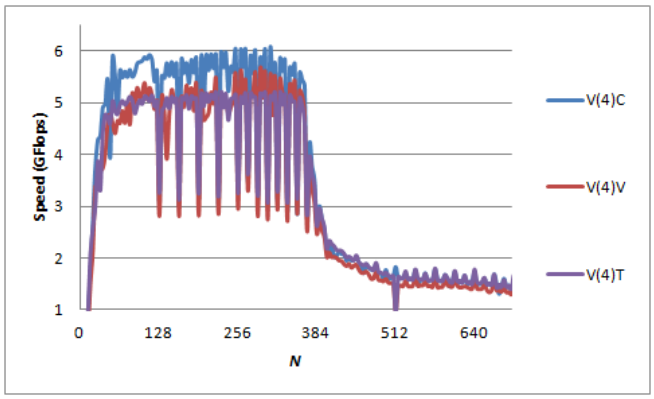


Figure 4. Speed comparison for traditional / virtual machine allocated with all hardware resources (4 threads)

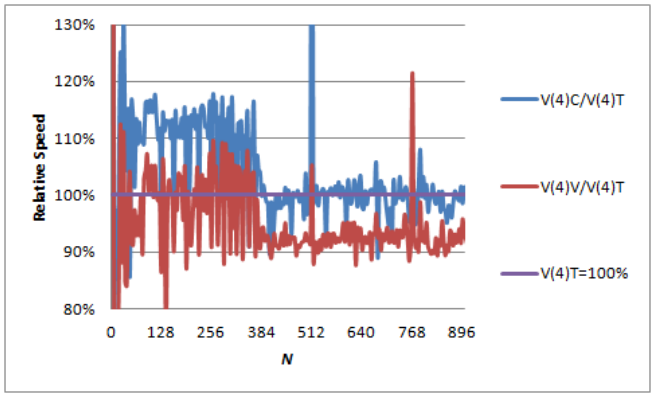


Figure 5. Relative speed comparison for Fig. 4.

Analyzing the performance by comparing the three curves in figures 4 and 5, we can conclude that cloud virtualization performs the algorithm better than other two environments in the L2 region. Virtualization also performs better than traditional environment in the same L2 region, but produces worse performance in points where performance drawbacks appear due to cache set associativity described in [10]. Cloud and traditional environments provide similar performance in L3 region, i.e., shared main memory, much better than virtual environment. The conclusion is that in this region virtualization provides the worst performance and cloud environment achieves the best performance.

Another important conclusion is the fact that the speed increases in the L2 region where the cache memory is dedicated per core (group of 2 cores) for virtual and cloud environments. However, the speed decreases in the shared memory L3 region when matrix size N increases demanding more memory requirements, generating higher cache miss penalty and increasing the overall memory access time.

Based on results of these experiments, we can conclude that cloud virtual environment achieves better performance compared to traditional environment for cache intensive algorithms in the L2 region using dedicated L2 cache per core and shared L3 cache and main memory. Section VI describes the causes for this phenomenon.

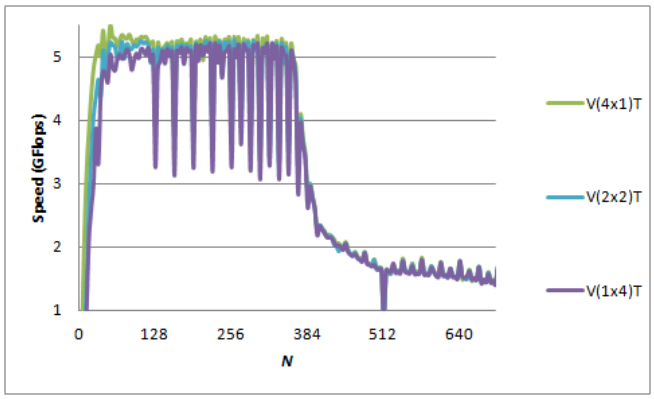


Figure 6. Speed comparison for traditional machine allocated with different resources per thread

IV. Multiprocess, Multithread and Multitenant Environment Performance

This section presents the results of the experiments performed on traditional and cloud workload environment when the resources (cores) are shared among processes, threads and tenants in different ways.

A. Multiprocessing and Multithreading in Traditional Environment

This Section presents the results of the experiments that run test cases 1.1, 1.2 and 1.3 described in Section II, i.e., different resource allocation per process in traditional environment.

The achieved speed for the matrix multiplication algorithm is presented in Fig. 6 in gigaflops for different matrix size N executing with 1, 2 and 4 processes using total 4 threads on all 4 cores on the same traditional environment. By V(1x4)T, we denote the results obtained for environment defined in the test case 1.1, V(2x2)T the test case 1.2 and V(4x1)T the test case 1.3.

The same two regions (L2 and L3) are depicted in Fig. 6 identified by different speed performance for all 3 test cases.

The relative ratio of achieved speeds in comparison to the traditional environment defined in test case 1.1 with 1 process and 4 parallel processes is presented in Fig. 7.

Comparing the obtained curves in figures 6 and 7 we can conclude that environment for test case 1.3 is the leader in the speed race in front of case 1.2 and 1.1 for the L2 region. All test cases provide similar performance in the L3 region where the best performance is achieved by test case 1.3.

The fact that the speed is almost linear in the L2 region where cache memory is dedicated per core (group of 2 cores) is also an important conclusion. However, the speed decreases for all 3 test cases in the shared memory L3 region

 ISBN: 978-1-61208-216-5

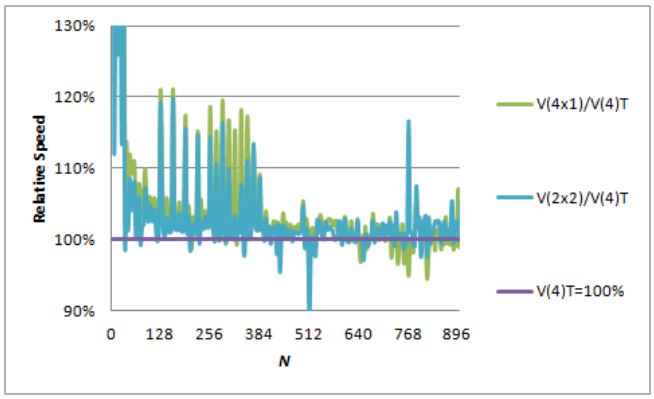


Figure 7. Relative speed comparison for Fig. 6.

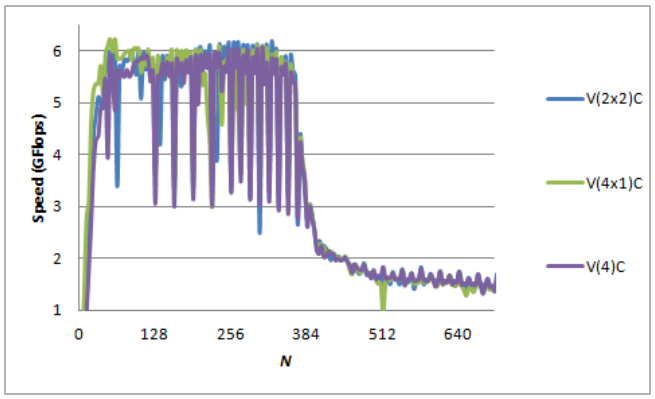


Figure 8. Speed comparison for virtual machine(s) in cloud allocated with different resources per machine and per thread

when the matrix size N is increased and higher cache miss penalty is generated.

We can conclude that dividing the problem in separate processes is the best solution for cache intensive algorithms in the L2 region. The OpenMP handles better in the L3 region by allocating all the resources to one process that executes concurrently with maximum number of threads equal to the number of cores.

B. Multi-tenant / Multi-threading in Virtual Cloud Environment

This section presents the results of the experiments that run test cases 3.1, 3.2 and 3.3 described in Section II with different resource allocation per tenant in cloud virtual environment.

The speed achieved for the matrix multiplication algorithm is presented in Fig. 8 for different matrix size N of the matrix multiplication executing on one, two and four VM using total 4 threads on all 4 cores on the same cloud virtual environment. The curves are identified by V(4)C for test case 3.1, V(2x2)C for test case 3.2 and V(4x1)C for test case 3.3. The relative differences to the default speed V(4)C are presented in Fig. 9.

Fig. 8 presents that the same two regions L2 and L3 can be identified by different performance for all 3 test cases.

Analyzing the performance behavior presented in figures 8 and 9 we can conclude that the environment defined by test case 3.3 is the leader in the speed race in front of the test cases 3.2 and 3.1 for the left part of the L2 region, and the environment for test case 3.2 is the leader for the speed race in front of the test cases 3.3 and 3.1 in the right part of the L2 region. All test cases provide similar performance in the L3 region with test 3.1 as a leader.

We can also conclude that the speed increases in the L2 region where cache memory is dedicated per core (group of 2 cores) for all three test cases. However, the speed decreases for all test cases in the shared memory L3 region when the matrix size N is increased enough and higher cache miss

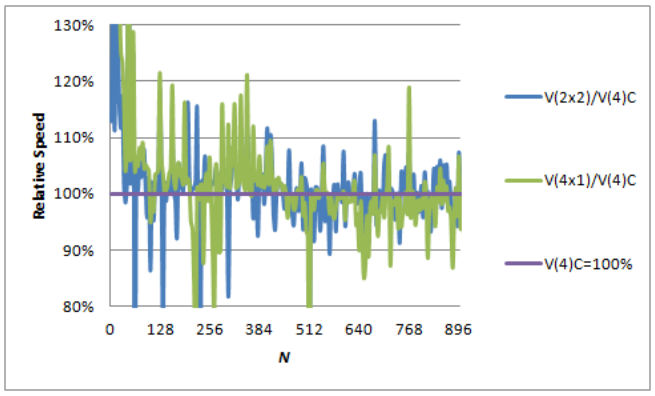


Figure 9. Relative speed comparison for Fig. 8

penalty is generated increasing the overall memory access time.

Dividing the problem in separate concurrent VMs is the best solution for cache intensive algorithms in the L2 region for dedicated L2 caches. The best solution for the L3 region with shared main memory is to allocate all the resources to one process (VM) to be executed concurrently with maximum threads as number of cores.

V. Traditional vs Virtual vs Cloud Environment Performance for Sequential Execution

This section presents the results of the experiments performed on three workload environments for sequential execution, i.e., test cases 1.4, 2.2 and 3.4 as described in Section II.

The achieved speed for execution of the matrix multiplication algorithm is shown in Fig. 10. The figure depicts the speed in gigaflops for different matrix size N when executing one process sequentially on one core on three different system environments as described in Section II. The curves are identified by V(1)T for traditional environment,

 ISBN: 978-1-61208-216-5

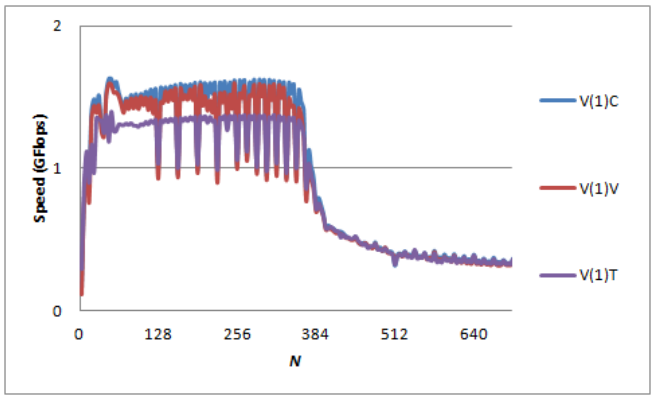


Figure 10. Speed comparison for sequential execution in the three environments

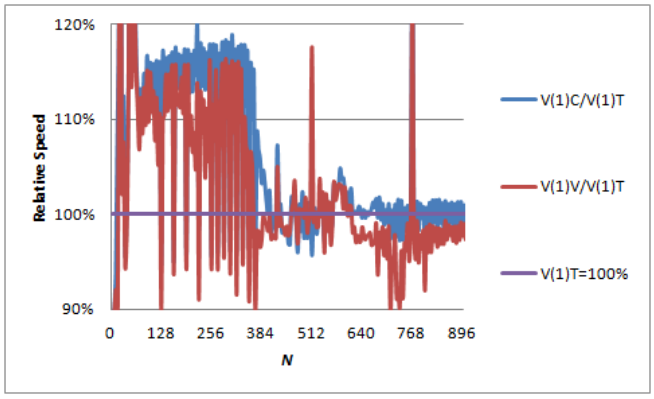


Figure 11. Relative speed comparison for Fig. 10.

V(1)V for environment with virtual and V(1)C with cloud environment.

The performance analysis of the curves in figures 10 and 11 shows that cloud virtualization achieves better performance for the algorithm execution in the L2 region. Virtualization also performs better than traditional environment in the same L2 region. Cloud and traditional environments provide similar performance in the L3 region, better than virtual environment. The conclusion is that in this region virtualization provides the worst performance and cloud environment the best performance.

VI. Cache Miss Analisys

This section presents the results of the experiments realized using Valgrind [11] to prove why the algorithm runs better in cloud environment in L2 region and runs better in traditional environment in L3 region. L1 and L2 cache misses are analyzed for both L2 and L3 regions for sequential execution in traditional and cloud environment. Table VI presents the results of these experiments. L1DT and L1DC identifies the number of L1 data cache misses for traditional and cloud environment correspondingly, and

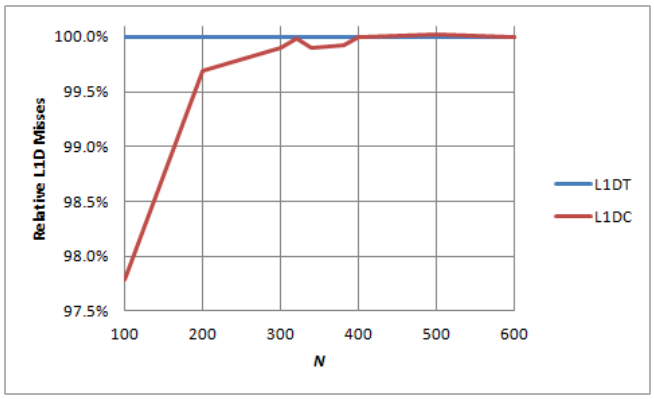


Figure 12. Relative comparison for L1 data cache misses

L2DT and L2DC for the number of L2 data cache misses for both environments correspondingly.

Table I
Number of L1 and L2 data cache misses in cloud and traditional environment in some points in L2 and L3 regions

N	L1DT	L1DC	L2DT	L2DC
100	145,572	142,344	11,553	9,176
200	1,039,954	1,036,676	22,807	20,432
300	3,448,511	3,445,020	41,580	48,158
320	33,600,548	33,597,359	46,329	72,598
340	5,011,438	5,006,561	51,501	83,472
360	5,941,929	5,936,645	60,225	94,021
380	7,093,110	7,087,590	100,675	106,888
400	68,438,786	68,435,517	113,818	119,676
500	113,364,842	113,385,000	187,027	666,141
600	244,545,355	244,541,890	765,609	27,234,248

A. L1 Data Cache Misses

The relative ratio of L1 data cache misses in comparison to the traditional environment is depicted in Fig. 12. We can conclude that cloud environment achieves smaller number of L1 data cache misses than the traditional environment in the L2 region, and comparable number of L1 data cache misses in the L3 region.

B. L2 Data Cache Misses

The relative ratio of L2 data cache misses in comparison to the traditional environment is depicted in Fig. 13. We can conclude that cloud environment achieves smaller number of L2 cache misses than the traditional environment in the L2 region, but much more than traditional environment in the L3 region.

VII. Conclusion and Future Work

Several experiments including sequential and parallel executions are performed with different resource allocation in traditional, virtual and cloud environments on the same multiprocessor. The testing methodology addresses each environment with full utilization to all CPU cores with

 ISBN: 978-1-61208-216-5

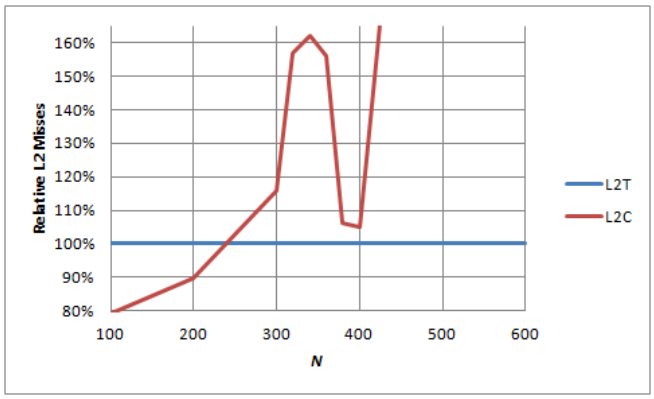


Figure 13. Relative comparison for for L2 data cache misses

different techniques: mono-process with multi-threading, multi-processes with multi-threading and multi-processes with single threads.

Cache intensive algorithm is the algorithm which is computationally intensive and memory demanding, i.e., utilizes the cache with data reuse to perform several computations. Simple matrix multiplication algorithm is used for sequential execution and 1D blocking matrix B for parallel execution to efficiently utilize cache performance. Our goal is not to create a new algorithm which exploits super linear speedup in cloud environment, but to examine the cache memory usage phenomenon and its performance impact in cloud.

The conclusions brought from the experiments performed in this paper are summarized to the impact of the resource allocation. Dividing the algorithm to parallel cores enables usage of more L1 and L2 cache for parallel version in comparison to the traditional environment, phenomenon explained in [3] for L1, L2 and L3 regions for multiprocessors using shared L2 cache and distributed L1 cache.

The experiments performed in this paper address several virtual machine instances in a cloud system using different number of CPUs (assuming all cores are utilized). Each experiment orchestrates the CPU cores differently. The contribution of the paper can be summarized as:

- The experiments prove that there is a region (L2 region) where cloud environment achieves better performance than traditional and virtual environment, both for parallel and sequential process execution, and
- The experiments prove that cloud computing provides better performance in a multi-VM environment, rather than allocating all the resources to only one VM.

The best resource allocation for traditional environment for cache intensive algorithms is the usage of multiple processes with single threads. Multiple VMs with single threads is the best resource allocation for cloud environment. Comparing the environments, cloud computing provides the best performance.

Future multiprocessors will have more cores and cache on chip with different cache types and results of this research will have higher impact. Our plan for further research is to continue with performance analysis of cloud computing on different hardware and cloud platforms with different hypervisors to analyze CPU behavior with different cache organization and the best platform for cache intensive algorithms. Experiments with MPI for inter VM instance communication are planned as future research.

REFERENCES

[1] Intel. Intel smart cache. [retrieved: May, 2012]. [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/intel-smart-cache.html

[2] J. Machina and A. Sodan, "Predicting cache needs and cache sensitivity for applications in cloud computing on cmp servers with configurable caches," in *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, ser. IPDPS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2009.5161233

[3] M. Gusev and S. Ristov, "Matrix multiplication performance analysis in virtualized shared memory multiprocessor," in *MIPRO, 2012 Proc. of the 35th Int. Convention*, 2012, pp. 264–269.

[4] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An analysis of performance interference effects in virtual environments," in *Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE International Symposium on*, april 2007, pp. 200 –209.

[5] P. Wang, W. Huang, and C. Varela, "Impact of virtual machine granularity on cloud computing workloads performance," in *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, oct. 2010, pp. 393 –400.

[6] B. Xiaoyong, "High performance computing for finite element in cloud," in *Future Computer Sciences and Application (ICFCSA), 2011 International Conference on*, June 2011, pp. 51 –53.

[7] cpu world. Intel(r) core(tm)2 quad cpu q9400. [retrieved: May, 2012]. [Online]. Available: http://www.cpu-world.com/sspec/SL/SLB6B.html

[8] Openstack. Openstack compute. [retrieved: May, 2012]. [Online]. Available: http://openstack.org/projects/compute/

[9] ——. Openstack dual node. [retrieved: May, 2012]. [Online]. Available: http://docs.stackops.org/display/documentation/Dual+node+deployment

[10] S. Ristov and M. Gusev, "Achieving maximum performance for matrix multiplication using set associative cache," in *Next Generation Information Technology (ICNIT), 2012 The 3rd International Conference on*, 2012, pp. 542–547.

[11] Valgrind. Valgrind. [retrieved: May, 2012]. [Online]. Available: http://valgrind.org/

 ISBN: 978-1-61208-216-5

A Framework for the Flexible Deployment of Scientific Workflows in Grid Environments

Javier Fabra, Sergio Hernández, Pedro Álvarez, Joaquín Ezpeleta
Aragón Institute of Engineering Research (I3A)
Department of Computer Science and Systems Engineering
University of Zaragoza, Spain
Email: {jfabra,shernandez,alvaper,ezpeleta}@unizar.es

***Abstract*—Scientific workflows are generally programmed and configured to be executed by a specific grid-based system. The integration of heterogeneous grid computing platforms in order to build more powerful infrastructures and the flexible deployment and execution of workflows over them are still two open challenges. Solutions based on meta-scheduling have been proposed, but more flexible and decentralized alternatives should be considered. In this paper, an alternative framework based on the use of a tuple-based coordination system and a set of mediation components is proposed. As a use case, the First Provenance Challenge has been implemented using two different workflow technologies executed over the framework, Nets-within-Nets and Taverna, and transparently deployed on two different computing insfrastructures. The proposed framework provides users with scalability and extensibility mechanisms, as well as a complete deployment and scheduling environment suitable for a wide variety of scenarios in the scientific computing area.**

Keywords – middleware for integration; scientific workflow deployment; grid-based systems.

I. INTRODUCTION

Grid computing emerged as a paradigm for the development of computing infrastructures able to share heterogeneous and geographically distributed resources [1]. Due to their computational and networking capabilities, this type of infrastructure has turned into execution environments suitable for scientific workflows. Scientific workflows are a type of workflow characterized for being composed by a large number of activities whose execution requires a high computation intensity and complex data management.

Currently, many efforts are being carried out in the field of scientific computing to execute their experiments taking full advantage of grid technologies. Two important open challenges in this area are the integration of heterogeneous grid computing platforms in order to build more powerful infrastructures and the flexible deployment and execution of workflows over them. Some authors have proposed solutions based on the use of meta-schedulings without considering dynamic behaviours or workloads. However, in order to tackle with the nature of grids, it is required to consider more flexible and decentralized alternatives.

In this paper, a framework able to tackle the previous challenges is proposed. As shown in [2], [3], the use of a broker based on the Linda coordination model [4] and a set of mediators facilitates the flexible integration of heterogeneous grid computing environments, addressing the challenge of creating more powerful infrastructures. These components encapsulate and handle specific features of various computing environments integrated into our framework, being programmers unaware of this heterogeneity. As a result, the tasks that compose a workflow can be executed in a flexible way using different computing environments. Unlike current proposals the framework is not based on the use of a meta-scheduler to perform global scheduling decisions, but each computing environment competes to execute jobs according to the availability of its own grid resources. In order to implement this alternative scheduling model, each one of these computing environments is represented in the broker by a specific mediator able to achieve suitable scheduling decisions. Hybrid computing environments could be easily integrated implementing new mediators. On the other hand, scientific workflows can be programmed independently of the execution environment in which they will be executed. The Net-within-Nets paradigm [5] and the Renew tool [6] have been used for programming this type of workflows. This is also compatible with other existing workflow programming languages. Indeed, Taverna workflows can be programmed using the framework services or translated to our programming language and then executed.

The remainder of the paper is organized as follows. Section II introduces some related work. In Section III, the architecture of the framework is presented. The role of the Linda-based broker, its implementation details and task dispatching mechanisms are described in Section IV. The flexible integration of heterogenous grid middlewares and grid management components with the broker is then detailed in Section V. The features and new capabilities are shown by means of an example that implements the First Provenance Challenge in Section VI. Finally, conclusions are depicted in Section VII.

II. RELATED WORK

A considerable progress has been made in the understanding of the particular nature of scientific workflows and the implementation of grid-based systems for their specification, scheduling, and execution. A detailed survey of existing grid workflow systems is presented in [7], [8]. The comparison of several systems shows relevant differences in the building and

 ISBN: 978-1-61208-216-5

execution of workflows that causes experiments programmed by scientists and engineers to be strongly coupled to the underlying grid-based execution system. This coupling forces grid administrators to perform relevant configuration and integration efforts in most of the scientific workflow deployments. Therefore, some interesting challenges are still open: the ability to program scientific workflows independently of the execution environment, the portability of scientific workflows from one execution environment to another, or the integration of heterogeneous execution environments to create more powerful computation infrastructures, for instance. Consequently, research efforts should concentrate on the definition of new high-level programming constructs independent of specific grid technologies and also on the provision of execution infrastructures able to interface multiple providers. This type of infrastructure should integrate software adaptation layers for translating generic management operations to provider-specific APIs. Additionally, new strategies of resource brokering and scheduling should be integrated into these execution environments to facilitate the utilization of multiple-domain resources and the allocation and binding of workflow activities to them.

Let us briefly resume some of the current proposals for provisioning flexible and extensible execution infrastructures. On the one hand, different grid-based systems built on a *meta-scheduler* have been proposed [9], [10], [11]. A meta-scheduler is a middleware component that provides advanced scheduling capabilities on a grid consisting of different computing platforms. The software architecture of all these solutions is very similar and is composed of the following components: a resource monitoring system to collect information from integrated computing platforms, a meta-scheduler to distribute jobs among grid resources using different scheduling policies [12] and, finally, a set of adaptation components to achieve mediation between middleware components and computing platforms. On the other hand, architectures based on the integration of meta-schedulers have been adapted for taking advantage of Cloud technologies [11], [13], [14]. Resulting computing environments comprise of virtualized services usage-based payment models in order to achieve more efficient and flexible solutions, where the supported functionality will be no longer fixed or locked to underlying infrastructure.

III. An Open Framework for Programming and Executing Scientific Workflows

In short, the main goals of our approach are:

- To execute scientific workflows programmed using a High-level Petri nets formalism or other standard languages widely accepted by the scientific community.
- To simultaneously work with different and heterogeneous grid middlewares or with middlewares implemented using different technologies (e.g., Web services). At this respect, workflow execution engines must be uncoupled from specific grid technologies.
- To allow the addition or removal of resources without previous announcement.
- To support different scheduling strategies and policies in the execution environment. The use of a particular scheduling strategy or policy should depend on the characteristics and requirements of each workflow application.

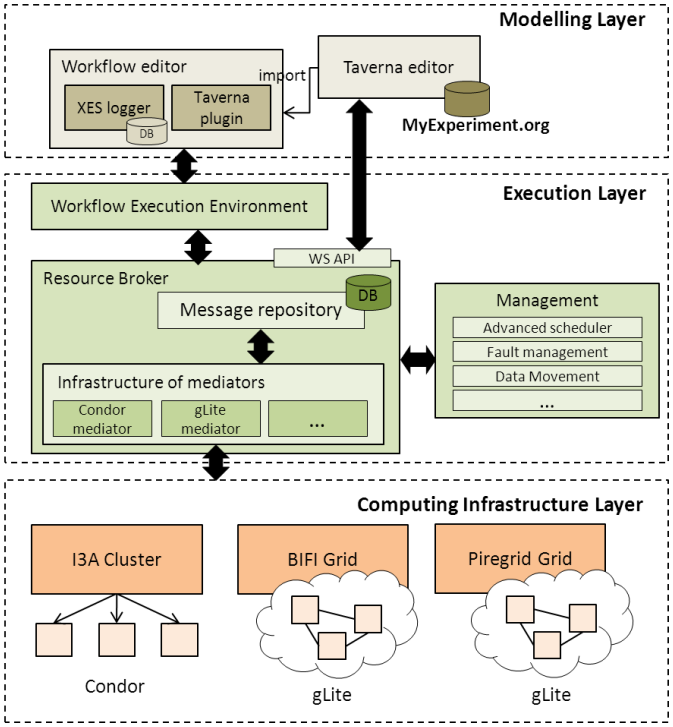


Fig. 1. Architecture of the execution environment.

Figure 1 shows the high-level architecture of the proposed framework. As shown, the architecture consists of three layers: the *modelling layer*, the *execution layer* and the *computing infrastructure layer*. In the following, each layer as well as its main components and interfaces are described in detail.

Firstly, the *modelling layer* consists of a set of tools for the programming of workflow applications. A workflow can be developed using the broker services, which are exposed through its Web service interface, using a workflow modeling tool such as Taverna [15], for instance. Also, we propose the use of Reference nets, a subclass of Petri nets, to implement workflow applications from the perspective of the Nets-within-Nets paradigm [5]. Nevertheless, other high-level programming languages for workflows could be also used by scientific communities (e.g., physicists, biologists or astronomers) for programming their workflows. With respect to this issue, plugins can be added to the modelling layer to support existing or new modelling approaches, such as the Taverna plugin shown in Figure 1, for instance. This plugin allows to import workflows programmed with Taverna, which are automatically translated to the workflow format in the workflow editor and then directly executed. A good repository for these type of workflows is the scientific community hosted at *MyExperiment.org*. In this work, Renew [6] is used as a workflow editor. Renew is an academic open-source tool that

 ISBN: 978-1-61208-216-5

allows the direct execution of Reference nets without any additional coding process and which represents a worth benefit for the final user.

Secondly, the *execution layer* is composed of the core components. The *workflow execution environment* is responsible for controlling the execution of workflows and submitting tasks to the *resource broker* when they must be executed. Internally, the broker consists of a *message repository* and a set of *mediators*. Messages are used to encapsulate any information that is passed through the components of the system. A message can describe a task to be executed or the result of its execution, for instance. Mediators encapsulate the heterogeneity of a specific grid middleware, having a complete knowledge of its capabilities. This knowledge is used for making dispatching decisions (which specific computing infrastructure will execute a pending task?). Subsequently, the grid middleware of the selected computing platform will schedule the set of resources needed for executing the task. As a result, the broker uncouples the workflow execution environment from the specific details about the grid-based computing infrastructures where tasks will be executed. This design avoids the need for a close integration of the workflow execution environment with specific grid middlewares used for the execution of tasks.

Let us now go deeper into the description of the two components of the broker. On the one hand, the Linda coordination model [4] has inspired the implementation of the message repository. Messages are encoded as tuples and stored into a tuple space. The interface of the repository provides a set of operations for accessing the tuples stored in the tuple space according to the semantics of Linda. In Section IV, we will depict the advantages of using a Linda-based repository and provide details about its implementation. On the other hand, *mediators* are required for achieving the aforementioned uncoupled integration. In general, a mediator is an entity that directly communicates with the tuple repository, matches and retrieves special-tagged tuples and processes them. In our approach, each grid middleware is represented by a mediator. Internally, this mediator is responsible for: i) having a complete information of the grid resource it represents; ii) interacting with the tuple repository to find at run-time tasks that could be executed by the set resources of its middleware; iii) dispatching the task to the middleware for its execution and controlling the input and output data transference; and, finally, iv) storing the results of the executed task in the tuple repository as tuples. Mediators of different and heterogeneous grid middlewares could compete for the execution of a specific task. Currently, as it will be described in Section V, different mediators have been implemented for the grid middleware we have access to (Condor and gLite) and then integrated into the *infrastructure of mediators*.

On the other hand, a set of *management components* has also been integrated into the execution layer to support the execution of workflow applications: the fault management component, the data movement component or the advanced scheduling component, for instance. The integration procedure of these components is similar to the one used by mediators. A management component interacts with the tuple repository in order to match and retrieve special-tagged tuples and then processes them. Therefore, the action of these components can be triggered as a result from the previous processing, which allows to dynamically compose complex action chains. In Section V the component for the fault management subsystem and its integration will be detailed.

Finally, the *computing infrastructure layer* is composed of different and heterogeneous computing platforms. The interaction with these platforms is managed by the corresponding grid middlewares. Currently, three computing platforms are integrated in the framework we manage: the HERMES cluster hosted by the Aragón Institute of Engineering Research (I3A), which is managed by the Condor middleware; and the two research and production grids managed by the gLite middleware and hosted by the Institute for Biocomputation and Physics of Complex Systems (BIFI) belonging to the European Grid Initiative (EGI), namely AraGrid and PireGrid.

To sum up, the open nature of the proposed solution is provided by the resource broker, composed of a Linda-based repository and a set of mediators, providing scientists with a high level of abstraction and flexibility when developing workflows. On the one hand, workflow programmers must concentrate on the functional description of workflow tasks and corresponding involved data. Specific details about the computing platforms where these tasks will be executed are ignored from the programmer perspective. On the other hand, the message repository facilitates the integration of mediators and management components and the scalability of the overall framework. Currently, its dispatching model is based on the functional capabilities of the computing platforms managed by the set of mediators. And, finally, these mediators are responsible for encapsulating the technological heterogeneity of the different types of grid middlewares and resource-access technologies (e.g., Web services). New mediators may be easily added in order to integrate new middlewares or technologies.

IV. LINDA-BASED TASK DISPATCHING

As previously stated, the resource broker is composed of a message repository and a set of components (mediators) that interact through this space by means of the exchange of messages. In this section, the role of the Linda-based message repository and the corresponding task description and dispatching mechanisms are presented.

Linda [4] is a coordination model based on two notions: tuples and a tuple-space. A tuple is something like [”Gelernter”, 1989], a list of untyped values. The tuple space is a collection of tuples stored in a shared and global space that can be accessed with certain operations, that allow processes to read and take tuples from and write them into it in a decentralized manner. For instance, the operation `in(x,["Gelernter", ?])` tries to match the *template* `["Gelernter", ?]`, which contains a wildcard, with a tuple in the shared space. If there is a match, a tuple is extracted from the tuple space and

 ISBN: 978-1-61208-216-5

assigned to variable `x`; otherwise, the process blocks until a matching tuple appears. The matching is free for the wildcard, but literal for constant values. The Linda matching mechanism allows easily programming distributed synchronization processes.

Linda-based coordination systems have been widely used for communicating and coordinating distributed processes. Their success in distributed systems is due to a reduced set of basic operations, a data-driven coordination and a space and time uncoupled communication among processes that can cooperate without adapting or announcing themselves [16].

Let us now introduce how tuples are describe and dispatched in our appraoch. Tuples are used to code the information needed for submitting a job to a grid middleware or recovering the result (or an exception) of an executed job. A tuple structure based on the *Job Submission Description Language* standard, JSDL [18], has been adopted. From the job submission point of view, this representation includes the specification of the application to be executed, the references to input and output data (represented by the corresponding URIs), a description of the host required for its execution (operating system, CPU architecture and features, memory, network bandwidth, etc.), QoS parameters and, optionally, the grid middleware responsible for its execution. In case the target grid platform is not specified, different mediators compete for the job execution in base to certain policies. On the other hand, a result tuple contains a reference to the original request, a reference to the output data and the execution log (grid and host used for the job execution, execution costs and QoS results, mainly). If an error occurs, the result tuple will contain the information about it. The fault handling component, which handles these faults, will be depicted in Section V.

Once the tuple representing a job has been created, the workflow execution environment puts it into the message repository by means of an `out` operation. Each grid computing platform is connected to the platform by means of a mediator, which knows the applications that could be locally executed by its grid and the description of the available internal resources. Each mediator is then waiting for tuples that encode such job requests able to be executed by its grid. Obviously, this waiting will depend on the availability at run-time of the grid and its capabilities. An `in` operation is invoked by the mediator in order to retrieve a tuple of its interest, using the Linda matching mechanism. Then, the retrieved tuple is locally processed by the mediator to perform the corresponding invocation to the grid middleware it represents.

If many grid computing platforms are able to execute a job, their mediators will compete to retrieve the job request tuple. The Linda matching mechanism is non-deterministic and, therefore, it does not offer any further guidance about which mediator will retrieve the job request tuple. In this work, the use of WS-PTRLinda, an extension of a previous distributed Linda-based implementation of a message broker, called *DRLinda* [17], is proposed. As DRLinda, WS-PTRLinda was developed using Nets-within-Nets and the Renew tool, the same technologies we used for programming workflow applications. WS-PTRLinda provides a new Web-service based interface (SOAP 1x. SOAP2 and REST), support for persistence of the tuple space (for high-availability demanding environments), and a timeout mechanism useful for failure detection. Currently, a basic and non-deterministic scheduling is being used for dispatching job requests to grid mediators. In [17], we proposed and implemented some alternative matching mechanisms to solve specific problems. Similarly, new grid-oriented matching mechanisms could be defined to extend the scheduling policies of the broker (e.g., a QoS-based scheduling policy). Let us finally comment on two relevant advantages of this Linda-based brokering. Firstly, the cooperation is uncoupled because the execution environment does not have any prior knowledge about mediators and vice versa. The interaction style is adequate enough to be used in environments where it is very important to reduce as much as possible the shared knowledge between different components. Also, writing and reading components can cooperate without adapting or announcing themselves. New mediators could be added/removed without affecting the rest of components integrated into the framework.

V. FLEXIBLE INTEGRATION OF GRID MIDDLEWARES

Following the presented approach, different types of resources and components (execution engines, management components or mediators, for instance) can be integrated in an easy and uncoupled way. The only requirement for these components is to implement the Linda coordination API in order to put and remove tuples. Besides, components can be added or removed dynamically and transparently to the rest of the system, facilitating this way the scalability and adaptation of the framework.

In this section, two different types of integrated components are presented. The first one is a mediator able to interact with the Condor middleware, whereas the second one is a fault management component. When a fault is detected during the execution of a job, this component will re-schedule the job according to different policies. Our aim is to illustrate how this solution is able to interact with grid computing platforms managed by heterogeneous grid middlewares.

A. Interaction with the Condor middleware

As previously described, the framework is able to interact with several underlying grid infrastructures. Let us depict how a mediator has been developed to integrate a Condor middleware. Specifically, this mediator is responsible for the interaction with the HERMES cluster. Figure 2 shows the functional components of the mediator required for supporting such interaction. Additionally, this mediator can be reused for interacting with any computing platform managed by Condor.

The *Job Manager* interacts with the Linda-based broker depicted in the previous section in order to read job requests and write their results. Obviously, all request types that could be fulfilled by the cluster must be known by the manager. For this purpose, the *Internal Resource Registry* knows the list of applications that could be locally executed and the

 ISBN: 978-1-61208-216-5

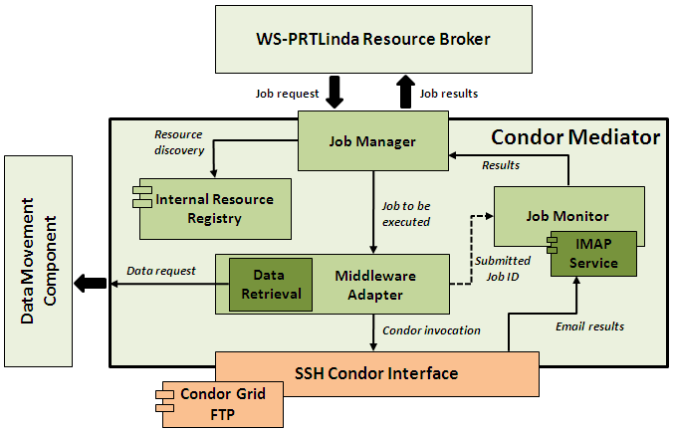


Fig. 2. Components of the Condor mediator.

description of available internal resources. This registry should monitor the cluster and dynamically update its information, but at this first implementation of the Condor mediator this information is static. Once a job request has been retrieved, the manager sends it to the *Middleware Adapter* component that is responsible for translating the request into a Condor job. Before submitting the job to the cluster via the SSH protocol, the adapter internally carries out two important tasks. First, it assigns an identifier to the job (*Job ID*) and sends it to the *Job Monitor* component. This ID will be used to correlate jobs and tuples. In case the input data required by a job are stored in an external computing platform, the adapter interacts with the Data Movement component for moving them (or making a copy) into the Condor cluster. After that, the adapter submits the job to the Condor middleware.

Internally, Condor can schedule the execution of submitted jobs depending on the local state of its resources. The goal is to achieve the best possible throughput. Therefore, a double scheduling can be done in the approach, similarly to the hierarchical scheduling model described in [19]. Once the job execution has been completed, results are sent through a logging mechanism (in our case, SMTP-IMAP) service integrated in the *Job Monitor*. This component maps received results with job requests and forwards them to the job manager. Finally, results are written in the broker so they can be then taken by the workflow application that submitted the original request.

This design and implementation is quite flexible and provides reusability. For instance, we have also developed a mediator to interact with the gLite middleware used in AraGrid. Its design is similar to the previous one. In fact, most internal components have been reused, as the job manager and the internal resource registry, and others components have been adapted, as the middleware adapter or the job monitor, for instance.

B. Fault handling

When dealing with scientific workflows, failures can arise at several levels. In this work, we will focus on those faults and exceptions that happen at the execution level. When the execution of a job fails, the corresponding mediator captures the fault and puts an error tuple into the message repository. This tuple, which will be processed by the *Fault management component*, contains information about the cause of the fault that will be used by the manager to take a decision with respect to the job execution. Different decisions could be taken: to submit the job again to the same grid computing platform, to submit the job to an alternative and reliable grid computing platform or to notify the error to the workflow executing environment in case the error persists, for instance. In the last case, most grid solutions offer two different ways to manage the fault: corrective actions or alternative workflows.

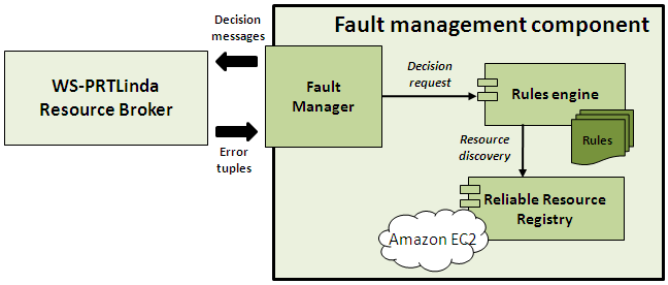


Fig. 3. Components of the fault management component.

Figure 3 shows the internal design of the fault management component. A *Fault Manager* interacts with the message repository in order to retrieve error tuples and to write the corresponding decision tuple. When an error tuple is found, the fault manager processes it and creates a decision request that is sent to a decision maker. We have used a *rules engine* as the decision maker. Rules are encoded in RuleML (the standard Web language for rules using XML markup [20]) and describe the corrective actions that will be executed in case of each type of error. These actions can be changed and modified at runtime, providing adaptation capabilities based on specific scenarios. Normally, the job will be sent again for a new execution on the corresponding infrastructure. However, in case it fails again or even if the error tuple contains some critical information, a usual action is to send the job request to a reliable grid middleware (our ultimate goal is the successful execution of job requests). Reliable grid middlewares have special characteristics (number of nodes, throughput, rejection rate, etc.), which turn them into more suitable candidates for a difficult job execution. For this purpose, a *Reliable Resource Registry* has been implemented and integrated in the fault management component. The current version of the registry contains a list of reliable grid middlewares. This list is used by the rules engine to decide in which middleware the failed job request will be executed. Finally, the fault manager puts a new job request tuple into the broker, specifying the grid middleware responsible for its execution.

VI. A CASE STUDY: THE FIRST PROVENANCE CHALLENGE

As a case study we present a workflow implementing the First Provenance Challenge [21].

 ISBN: 978-1-61208-216-5

The goal of the First Provenance Challenge (FPC) workflow is to create a *brain atlas* from an archive of four high resolution anatomical data and a reference image. Some image processing services are required for the workflow execution. These services have been deployed into heterogenous grid middlewares (more specifically, into the Condor cluster hosted by the I3A Institute and the gLite grids hosted by the BIFI Institute). In this example we show the flexibility of our proposal: some jobs are programmed to be executed by a specific computing platform, and other jobs may be executed by any available computing platform able to invoke the required service.

The workflow requires seven input parameters, whose specific values are implemented as the initial markings of places `Grid_Environment`, `Reference_image`, `Input_image_{1..4}`, and `Images_directory`. Their meanings are, respectively: the URL of one of the clusters where the workflow is going to be executed (more specifically, the cluster hosted by the I3A), the URI of the reference image, the URIs of the four images to be processed and the directory where the intermediate and final image files will be stored.

Figure 4 shows the implementation of the workflow using the Renew tool. Due to space limitations, only the first image processing flow is detailed in the figure, although the remaining branches for anatomy `Image2`, `Image3` and `Image4` are similar. Alternatively, Figure 5 depicts the implementation of the same workflow using Taverna. Job requests and results are encoded as Linda tuples. A request tuple is a nested tuple composed of four elements: the application or service to be executed and the URIs of the input and output data, the file descriptors for standard streams, QoS parameters and the computing platform where the request is going to be executed, respectively. Let us explain a tuple example, specifically the tuple depicted in transition `Align_warp_1(out)`. By putting that tuple in the message repository, the `Align_warp` service is invoked by the corresponding mediator using as input data an anatomy image, a reference image and their headers. The output is a warped image. For the sake of simplicity, file descriptors and QoS parameters are omitted in the tuple. Finally, the initial marking of the `grid_environment` place determines the value of the `grid` variable and, therefore, the computing platform selected for the job execution (the first field of this last tuple contains the access information required by the platform).

Tuples are either built and put into the message repository by means of the `Broker.out` action (as in the `Align_warp_1 (out) transition`, for instance) or withdrawn from the broker by means of the `Broker.in` action (as in the `Align_warp_1 (in) transition`, for instance). The sequential execution of these couple of transitions for a given image corresponds to an asynchronous call to the `Align_warp` service: first, the tuple with the information is put into the message broker, then the corresponding mediator takes it and invokes the service, putting the invocation result into the broker as a tuple and finally the result is captured and put into the workflow net by means of

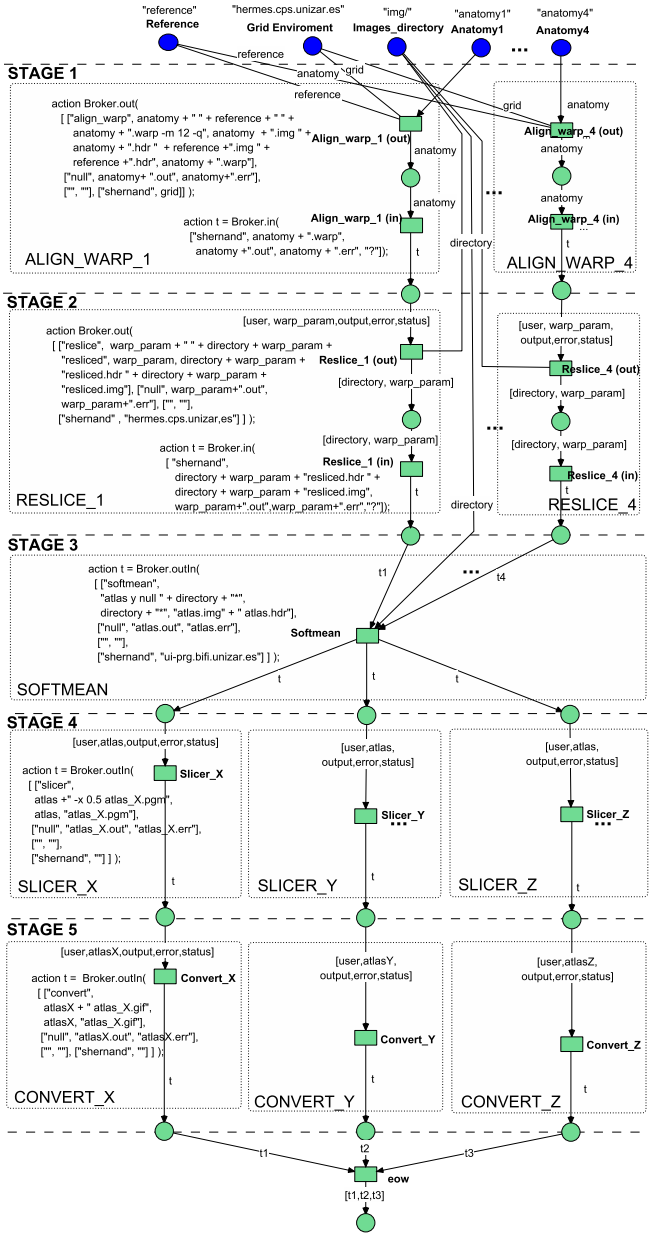


Fig. 4. Nets-within-Nets based implementation of the First Provenance Challenge workflow.

the second transition. Given the semantics of Petri nets, the processing of the input images can be done in any interleaved way, since tuples are put/removed into/from the broker as soon as resources are available. In this first stage the job request is executed in the cluster specified by the initial marking (the `grid` variable is an input parameter of the request submitted to the broker by the `Align_warp (out)` transition).

Once stages 1 and 2 are finished, Stage 3 takes the whole set of images from the directory specified by the parameter `Images_directory`, and executes the `softmean` method with these images as an input. At this stage the service deployed in one of the grids hosted by the BIFI institute is explicitly invoked. The last job request and its result are

 ISBN: 978-1-61208-216-5

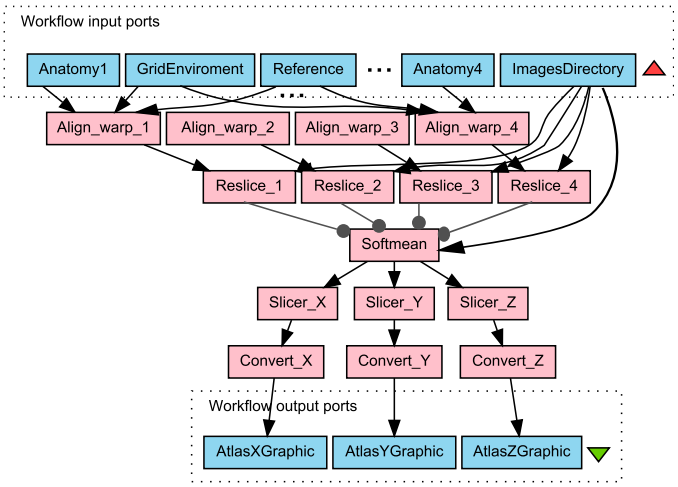


Fig. 5. Taverna implementation of the First Provenance Challenge workflow.

carried out by means of the `Broker.outIn` action: from the workflow point of view this corresponds to a synchronous call to the service described in the tuple. Then, softmean results are distributed so that stages 4 and 5 could be executed in parallel to compute the atlas data set for each dimension in axis x, y and z. The slicer and convert jobs could be executed by any available computing platform. Therefore, different executions of the workflow could invoke services deployed in different platforms. Finally, firing of transition `eow` (*end-of-workflow*) terminates the workflow. The resulting images will have been stored in the images directory.

Figure 5 depicts the workflow implemented with Taverna (some flow symbols in the top of the figure have been removed to improve readability). As shown, the structure is similar to the Nets-within-Nets implementation, although in this case the workflow is composed of several subworkflows, each of them implementing the previous invocations to the broker in order to put and withdraw tuples. Due to space limitations, the description of these subworkflows is left out of this paper.

A. Flexible deployment and execution

In order to analyze and test the transparency and flexibility of the proposed approach, the First Provenance Challenge workflow was executed using the framework. The target computing infrastructure for the execution of each stage (which can be specified in out transitions at each stage in Figure 5) was left unset, meaning that the mediators compete for each submitted task. At this respect, both HERMES and AraGrid were setup to separately allow the execution of the FPC workflow. However, as the aim of this experiment was to improve the overall execution cost of the workflow, the advanced scheduling component was programmed to perform a meta-scheduling process considering the load of the underlying computing infrastructures and the history of previous executions. Therefore, at every moment the best suitable candidate is estimated, avoiding the dispatching of a task to an overloaded infrastructure. This means that each task is first captured by the advanced scheduling component and then the target infrastructure is set (so, the corresponding mediator will retrieve the task for its execution). However, the whole process is transparent from the user's perspective.

To do that, the advanced scheduler also considered the average load of each infrastructure at every moment. Figure 6 depicts the daily average load (% of the maximum load) in the HERMES and AraGrid computing infrastructures. As it can be observed, both computing infrastructures have different load models. Their trends during the day as well as the previous execution time are used to decide the most suitable candidate for each task deployment.

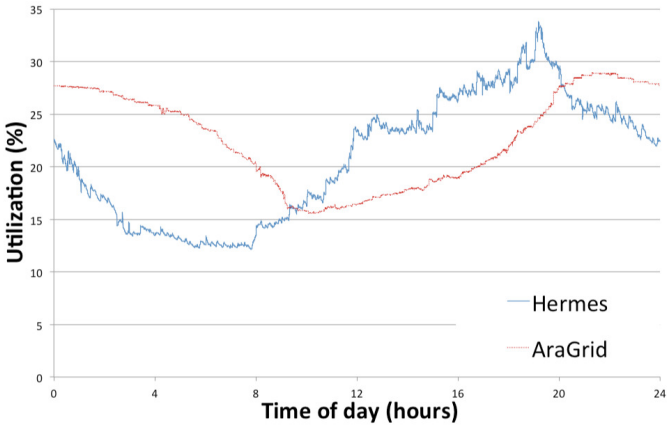


Fig. 6. Hermes and AraGrid daily utilization (in percentaje).

Figure 7 depicts the results obtained for 900 executions of the FPC workflow deployed on the framework. Average execution times (in seconds) are shown for each separated infrastructure (HERMES and AraGrid) and also for the framework for each stage of the First Provenance Challenge workflow. The overall execution time (average) is better when using the framework. This is due to the best candidate selection performed by the advanced scheduler (in most cases). The analysis of each separated stage depics that most of the time (70%) the HERMES cluster computing infrastructure gets a better execution time that AraGrid, which is related to the fact that the framework execution time is closed to the HERMES one.

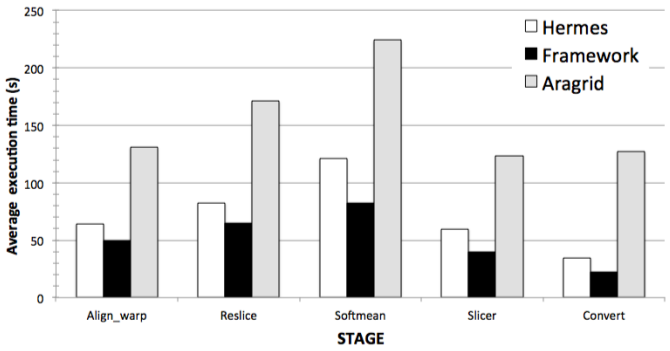


Fig. 7. Experimental results for the First Provenance Challenge workflow.

If we consider the average execution times for the complete workflow, AraGrid got the worst results with 777 seconds,

 ISBN: 978-1-61208-216-5

HERMES got 362 seconds and the framework got 260 seconds. Obviously, using the most adequate infrastructure to get the better execution time is not a trivial process from the researcher's point of view. However, by means of the use of the framework, this is done in a flexible and transparent way. Other possibilities are to reduce access costs (for instance, if each computing hour has an asssociated cost), resource usage, etc. Regarding the time to move data between the two infrastructures (as output from a stage is used as input of the following one), the average time for each workflow execution was less than 55 seconds (so the average framework execution time goes to 315 seconds).

VII. Conclusion and future work

In this paper, a framework to solve some of the open challenges in the application of grid-based solutions to scientific workflows has been presented. This framework is uncoupled from specific grid technologies, able to work simultaneously and transparently with different and heterogeneous grid middlewares, providing scientists with a high level of abstraction when developing their workflows. The integration of the execution environment with different grid middlewares has been carried out by means of a resource broker composed of a Linda-based coordination system and a set of mediators. Thanks to the aforementioned broker, this integration is flexible and scalable. On the other hand, regarding the workflow programming point of view, the proposal is also open and flexible. As it has been shown, workflows programmed using standard languages or existing service-oriented workflow management systems (e.g., Taverna) can also be executed in the framework.

Currently, the proposed framework is being applied to solve some complex and high time-consuming problems, such as the behavioural analysis of semantically-annotated scientific workflows, or the analysis of existing data connections into the Linked data cloud, for instance. These solutions will allow improving the capabilities of the presented approach and also analyzing its performances.

We are also working on the integration of Cloud-related solutions, such as using the *Amazon Elastic Cloud Computing Simple Queue Service* (Amazon EC2 SQS) in order to have an alternative message repository, as well as providing specific high-performance computing capabilities (indeed, currently Amazon EC2 offers a mechanism to virtualize a HPC machine, able to handle critical and complex computation tasks). Related to this last point, we are adding some external reliable computing platforms by means of virtualization technologies. In [2] we sketched the implementation of a similar mediator able to support the execution of business tasks. Similarly, a new mediator able to submit job requests to the EC2 interface with the required policies has been implemented.

Acknowledgment

This work has been supported by the research project TIN2010-17905, granted by the Spanish Ministry of Science and Innovation.

References

[1] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a new computing infrastructure*. Second edition, Morgan Kaufmann Publishers, 2004.

[2] J. Fabra, P. Álvarez, J.A. Bañares, and J. Ezpeleta, *DENEB: A Platform for the Development and Execution of Interoperable Dynamic Web Processes*. Concurrency and Computation: Practice and Experience, Vol. 23, Issue 18, pp. 2421-2451, 2011.

[3] R. Tolosana-Calasanz, J.A. Bañares, P. Álvarez, and J. Ezpeleta. *Vega: A Service-Oriented Grid Workflow Management System*. In 2nd International Conference on Grid computing, High Performance and Distributed Aplications (GADA'07), vol. 4805, pp. 1516-1523, 2007.

[4] N. Carriero and D. Gelernter, *Linda in context*. Communications of the ACM, Vol. 32, Num. 4, pp. 444-458, 1989.

[5] O. Kummer, *Introduction to petri nets and reference nets*. Sozionik Aktuell, Num. 1, pp. 19, 2001.

[6] O. Kummer and F. Wienberg, *Renew - the Reference Net Workshop. Tool Demonstrations*. In 21st International Conference on Application and Theory of Petri Nets (ICATPN 2000), pp. 87-89, 2000.

[7] M. Rahman, R. Ranjan, and R. Buyya, *A Taxonomy of Autonomic Application Management in Grids*. In 16th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2010), 2010.

[8] J. Yu and R. Buyya, *A taxonomy of workflow management systems for Grid Computing*. Journal of Grid Computing, Vol. 3, Issues 3-4, pp. 171-200, 2005.

[9] E. Huedo, R.S. Montero, and I.M. Llorente, *A Framework for Adaptive Execution on Grids*. Software - Practice and Experience , Vol. 34, Issue 7, pp. 631-651, 2004.

[10] M. Heidt, T. Dornemann, J. Dornemann, and B. Freisleben, *Omnivore: Integration of Grid meta-scheduling and peer-to-peer technologies*. In 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008), pp. 316-323, 2008.

[11] J. Tordsson, R.S. Montero, R. Moreno-Vozmediano, and I.M. Llorente, *Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers*. Future Generation Computer Systems, Vol. 28, pp. 358-367, 2012.

[12] J. Yu and R. Buyya, *A novel architecture for realizing Grid Workflows using Tuple Spaces*. In 5th IEEE/ACM International Workshop on Grid Computing (GRID 2004), Pittsburgh, USA, pp. 119-128, 2004.

[13] G. Mateescu, W. Gentzsch, and C.J. Ribbens, *Hybrid computing: Where HPC meets grid and Cloud computing*. Future Generation Computer Systems, Vol. 27, pp. 440-453, 2011.

[14] Y. Zhang, C. Koelbel, and K. Cooper, *Hybrid re-scheduling mechanisms for workflow applications on multi-cluster grid*. In 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009), pp. 116-123, Shanghai (China), 2009.

[15] *Taverna: an open source and domain independent Workflow Management System*. Available at http://www.taverna.org.uk/ [retrieved: May, 2012]

[16] P. Álvarez, J.A. Bañares, and P.R. Muro-Medrano, *An Architectural Pattern to Extend the Interaction Model between Web-Services: The Location-Based Service Context*. In 1st International Conference on Service Oriented Computing (ICSOC 2003), Lecture Notes in Computer Science, Vol. 2910, pp. 271-286, 2003.

[17] J. Fabra, P. Álvarez, and J. Ezpeleta, *DRLinda: A distributed message broker for collaborative interactions among business processes*. In 8th International Conference on Electronic Commerce and Web Technologies (EC-Web 2007), Lecture Notes in Computer Science, Vol. 4655, pp. 212-221, 2007.

[18] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, *Job Submission Description Language (JSDL) Specification, Version 1.0*. Available at http://www.gridforum.org/documents/GFD.56.pdf [retrieved: May, 2012]

[19] R. Sharma, V.K. Soni, M.K. Mishra, and P. Bhuyan, *A survey of job scheduling and resource management in grid computing*. World Academy of Science, Engineering and Technology, Issue 64, pp. 461-466, 2010.

[20] H. Boley, *Rule Markup Language, RuleML Specification. Version 1.0.*. Available at http://ruleml.org/ [retrieved: May, 2012]

[21] L. Moreau, B. Ludscher, I. Altintas , R.S. Barga, S. Bowers, and S. Callahan, *The First Provenance Challenge*. Concurrency and Computation: Practice and Experience, Vol. 20, Issue 5, pp. 409-418, 2008.

 ISBN: 978-1-61208-216-5

Semi-shared storage subsystem for OpenNebula

Sándor Ács, Péter Kacsuk, Miklós Kozlovszky
MTA SZTAKI Computer and Automation Research Institute,
H-1518 Budapest, P. O. Box 63, Hungary
[acs, kacsuk, m.kozlovszky]@sztaki.hu

***Abstract*— To address the limitations of OpenNebula storage subsystems, we have designed and developed an extension that is capable of achieving higher I/O throughput than the prior subsystems. The semi-shared storage subsystem uses central and distributed resources at the same time. Virtual machine instances with high availability requirements can run directly from central storage while other virtual machines can use local resources. As I/O performance measurements show, this technique can decrease I/O load on central storage by using local resources of host machines.**

Keywords - cloud computing; OpenNebula; storage subsystem; I/O performance.

I. INTRODUCTION

Cloud computing opens a new way of thinking about distributed information technology (IT) infrastructures [1]. The paradigm is based on virtualization technologies (server, storage, network, etc.) and it uses multiple experiences gathered from grid and cluster computing as well. In the three layered cloud model (Software/Platform/Infrastructure as a Service), the IaaS is the bottom layer that provides fundamental computing resources to consumers [2]. IaaS can be built from traditional IT hardware components and cloud middleware software.

OpenNebula [3] is an open source software stack, born from a research project and became one of the best-known IaaS cloud solution. The main components of OpenNebula are the front-end, compute nodes, image repository and networking infrastructure. The front-end machine is responsible for the core services (user authentication, scheduling, etc.) and provides an entry point for consumers. Compute nodes are hosts of virtual machines (VMs). The image repository handles virtual disk images and its storage subsystem contains physically the images. Compute nodes reach disk images directly via shared storage or copied through the network. If compute nodes use shared storage, VMs will consume the same resource that can cause decreased I/O performance for VMs. If compute nodes use non-shared storage, they will suffer from some disadvantages (e.g., slower VM deployment).

There are several open issues in cloud computing and one of them is related to the virtualized I/O performance [4]. Related studies [5] expose that the storage subsystem can play the key role from efficiency point of view in a cloud.

The main contribution presented in this paper is the concept of semi-shared storage subsystem that tries to alleviate the negative effects and find a trade-off between shared and non-shared storages. The semi-shared storage subsystem can provide benefits from both of the storage subsystems at the same time. It can share disk images between compute nodes for fast and flexible deployment and it can decrease the load with distributed non-shared resources.

We designed, implemented and tested the semi-shared storage subsystem for OpenNebula. I/O performance of the prototype is investigated in a local cloud installation and its values are compared to results of other existing storage subsystems. We present a technique that is able to achieve higher I/O throughput in OpenNebula than its prior solutions.

This paper is organized as follows: first, we introduce the related research results in Section II. Then, we present image management and features of the storage subsystems in OpenNebula in Section III. Next, we detail the semi-shared storage subsystem that helps to reduce the load in a cloud infrastructure. Afterwards, in Section V, we present the test infrastructure and results of the performance benchmarks. In Section VI, a production use case is introduced. Finally, we conclude our research in Section VII.

II. RELATED WORK

As related works have been already started to investigate the I/O performance of cloud infrastructures. Goshal et al. [5] introduced the Magellan project that explored some IaaS clouds from High Performance Computing (HPC) suitability point of view. The paper discloses that the performance of communication intensive applications is degraded by the virtualized I/O subsystem. Benchmarks were used on different types of clouds (e.g., Amazon EC2) and compared the results with local infrastructure measurements. Their results pointed out the major performance bottleneck which can be caused by virtualized environment.

Lihtium [6], a distributed storage system, was designed in order to avoid the limitation of centralized shared storage systems of cloud infrastructures. This solution is complex and specialized for virtualization workloads aimed at the large-scale cloud infrastructures and data-centers. The semi-shared storage solution for OpenNebula is lightweight and it can enhance the I/O throughput in small and middle-scale cloud infrastructures as well.

Ousterhoutet et al. [7] presented that the disk-oriented storage systems are problematic in a dynamic cloud environment. A new storage system was designed in order to achieve lower access latency and higher bandwidth. The solution is based on the main memories aggregation of the nodes. This new approach called RAMCloud, where all

 ISBN: 978-1-61208-216-5

information (disk images as well) is kept in DRAM. This solution promises 100-1000x faster throughput than disk-based systems and 100-1000x lower access latency. Using RAM based storages for improving the I/O performance of clouds has many benefits, however traditional disk based storages cost much less for the same capacity.

Sheepdog [8] is a distributed storage system that is integrated into QEMU/KVM [8]. It provides block level storage volumes redundantly based on distributed resources. Sheepdog supports volume management features such as snapshot and it can be scaled up without single point of failure to several hundred nodes. However, it cannot guarantee high bandwidth and low latency storage.

III. STORAGE SUBSYSTEMS AND DISK IMAGES

The image repository, accessible by the compute nodes, serves as a store for disk images in IaaS. The compute nodes can create copies from the disk images or they can use the images directly in order to create virtual machine instances.

1) Storage subsystem

In OpenNebula, the compute nodes can reach the disk images in different ways: (i) via shared storage or (ii) by copying it through the network from the image repository.

a) Shared storage

In Fig. 1, a compute node and an image repository with virtual disk images can be seen, where shared storage is available from the compute node, which can start the virtual machine instances.

With shared storage, the VMs can be started without copying it through the network and live migration is available for instances. The live migration is a procedure when a VM instance is moved from one host to the other without outage which can be sensed by end users.

The disadvantage of the shared storage is that all of the deployed virtual machines could use the same resource (storage subsystem of image repository) concurrently. The decreased I/O throughput causes performance loss for VM instances.

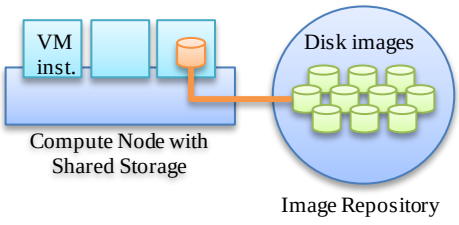


Figure 1. Compute node with shared storage

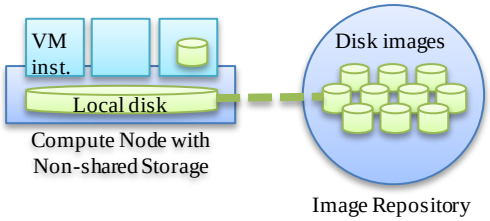


Figure 2. Compute node uses local copy from disk images

b) Non-shared storage

In Fig. 2, shared storage is not available, so the compute node cannot attach disk images directly from the image repository. The disk images should be copied through the network (broken line in Fig. 2) and stored in local storage. The virtual machine instances are created from local copies. The non-shared storage can cause peaks on I/O load while disk images are copying, however these peaks can be ignored if the VM instances are used long-term (days). In this paper, we investigate this option.

This storage subsystem can reduce the load on image repository with distributed resources however VM deployment and image sharing (copying and saving) takes more time and the live migration is not available.

2) Disk images

OpenNebula uses two types of disk images from volatility point of view. The state of the disk images can be persistent or non-persistent. If a virtual machine runs with persistent disk, the changes will be stored after shutdown. If a virtual machine uses non-persistent disk, the disk image will be deleted after shutdown.

Persistent and non-persistent disks can be used with shared- (Fig. 1) and non-shared (Fig. 2) storages as well. These options are detailed in the next two sections.

3) Disk images with shared storage

*a) Persistent disk:*The Virtual machines deployment is fast (compared to the overall process time of copy image from repository to local disk and from local disk until deployment). It is not needed to write back the changes after shut down because the disk image is attached directly to the image repository. The live migration is available in this option.

*b) Non-persistent disk:*These disk images are copied from the image repository, that takes more time than in option (a), however it is still faster than using non-shared storage. After shut down, the disk images are deleted, except if they were forced to be saved that means a copy from the instances in this case. The live migration is also available.

Summarized: With shared storage the fast VM deployment and live migration can be achieved. On the other hand, many VM instance with I/O intensive workloads can cause heavy load on the image repository.

 ISBN: 978-1-61208-216-5

4) Disk images with non-shared storage

*a) Persistent disk:*The Disk images are copied two times (for starting and saving) in the life of a virtual machine instance. The procedure of moving the VM instance from a compute node to another, takes more time than acceptable for live migration.

*b) Non-persistent disk:*These disk images are copied through the network from the image repository as well. They are deleted after shut down (except if they were forced to be saved by the user). The live migration is not available because of the non-shared storage.

Summarized: There is an overhead on the disk image sharing however the I/O workloads of the VM instances are distributed on the compute nodes. However, in this case just the slower cold migration is available instead of live migration for VM instances.

IV. THE SEMI-SHARED STORAGE SUBSYSTEM

As related works pointed out in Section II, the shared storage can be a bottleneck in a cloud and it can cause decreased I/O performance for VM instances. In this paper, we focused on the disk I/O. As presented in Section III, non-shared storage subsystem can be used to decrease the load on the image repository and to increase the VMs' disk performance because the VM instances use (distributed) local copies from the disk images instead of the shared storage.

In order to avoid the high load on image repository and increase the performance of the virtual disks, we propose the notion of semi-shared storage. As a proof of concept it was elaborated and implemented to OpenNebula.

The basic ideas were the following: the image repository component practically has more reliable storage subsystem than compute the nodes. Some VMs (e.g. database or firewall servers) may need to be migrated without outages. These VM instances should have persistent disk images based on shared storage because it takes time to copy the disk images trough the network and resume the operation of the VM instance. The loss of the fast start and live migration opportunities can cause that the non-shared storage is not sufficient to be used in high available production systems. However not all of the VM instances require features like live migration, fast deployment and having persistent disk images. These instances can be used with non-shared storage. (Of course, the non-persistent disk can be saved as well by the users.)

Our contribution to OpenNebula is the Semi-shared storage subsystem, which uses shared storage for persistent disk images and local copies with non-persistent disk images for creating VM instances. The benefit of this solution is that the shared- and non-shared files-systems can be used at the same time on the same compute node. The semi-shared storage subsystem can satisfy high availability requirements (like the original shared storage subsystem).

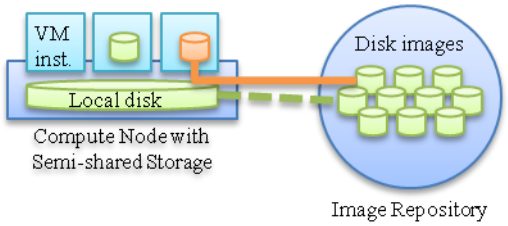


Figure 3. Semi-shared storage using local and shared resources concurrently

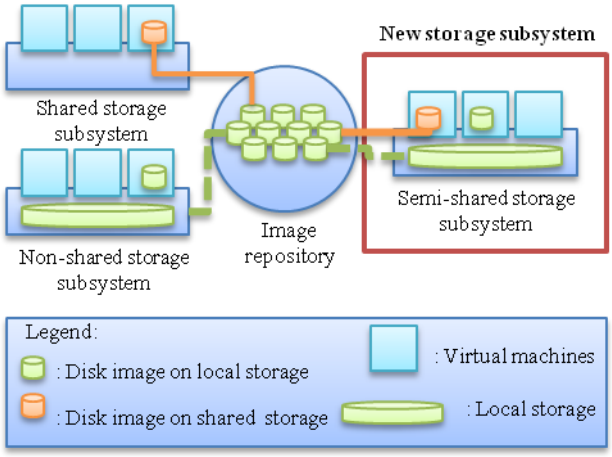


Figure 4. Semi-shared storage is using local- and shared resources at the same time

Moreover, it is able to decrease the load on image repository by using local storages of compute nodes. These may increase the performance of the disk images, especially in an over-provisioned cloud (like the original non-shared storage).In Fig. 3, the compute node uses shared- and non-shared storage at the same time. Shared storage is used for VM instances deployed with persistent disk image and local copies (non-shared storage) are used for VM instances deployed with non-persistent image. Fig. 4 summarizes the original (left side) and the new (right side) storage subsystems for OpenNebula.

V. TEST INFRASTRUCTURE AND PERFORMANCE BENCHMARKS

Experiments were carried out on an installation of OpenNebula (version 3.2) that consists of two compute nodes and one image repository. Technical details are summarized in "TABLE I".

TABLE I. CONFIGURATION OF THE TEST BED

Components of the test infrastructure				
Role	***Type***	***CPU***	***HDD***	***MEM***
Image-Repository Front-end	Sun Fire X2200 M2	2xQuad-Core Opteron 2.3G	Seagate ST32500N SATA	12G DDR2
2XCompute Nodes	Sun Fire X2200 M2	2xDual-Core Opteron 1.8G	WDC WD2500JS SATA 250G	8G DDR2

1) Testing of semi-shared storage

In order to prove that higher I/O performance is achievable by using the semi-shared storage subsystem than the prior (shared and non-shared) solutions could provide, I/O benchmarks were performed on the test cloud. The benchmarks were sequential read throughput tests because sequential read is a typical storage parameter [6]. At the same time, 8 exactly the same virtual machine instances were used to stress and load the I/O subsystem, while the performance was measured inside the virtual machine and directly on the physical block device with the iostat tool. Iostat is an I/O performance monitoring tool for Linux based systems. During the tests, the virtualization hypervisor was KVM and caches as well as buffers were disabled on every layer (files-system, hypervisor, etc.) for more accurate results [5]. The first diagram (Fig. 5) presents the results when a shared storage server and one compute node use semi-shared storage for benchmarking. The available I/O performance was measured in the image repository, compute node and individually in the VM instances as well. The benchmark values are the sequential read throughputs when all the 8 virtual machines are running. The first test batch has 9 pairs of columns. The pairs are distributions of VM instances between local and remote resources. In a pair, the first column is the aggregated I/O performance of the VMs and the second is the aggregated I/O performance of image repository and compute node.

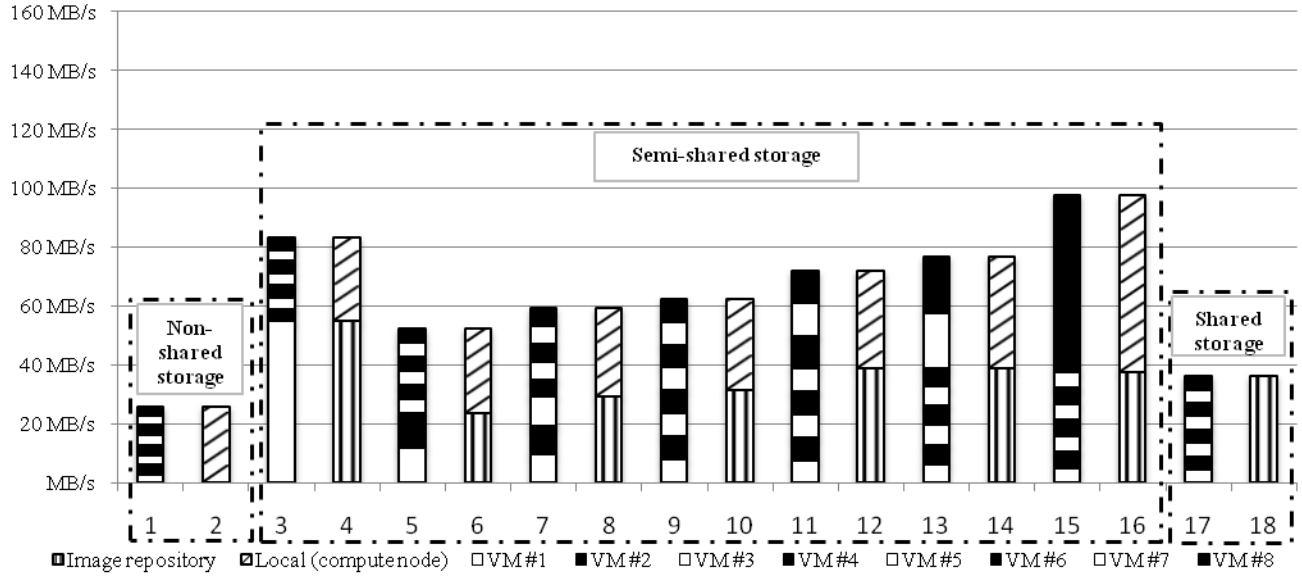


Figure 5. I/O benchmark with the image repository and one compute node

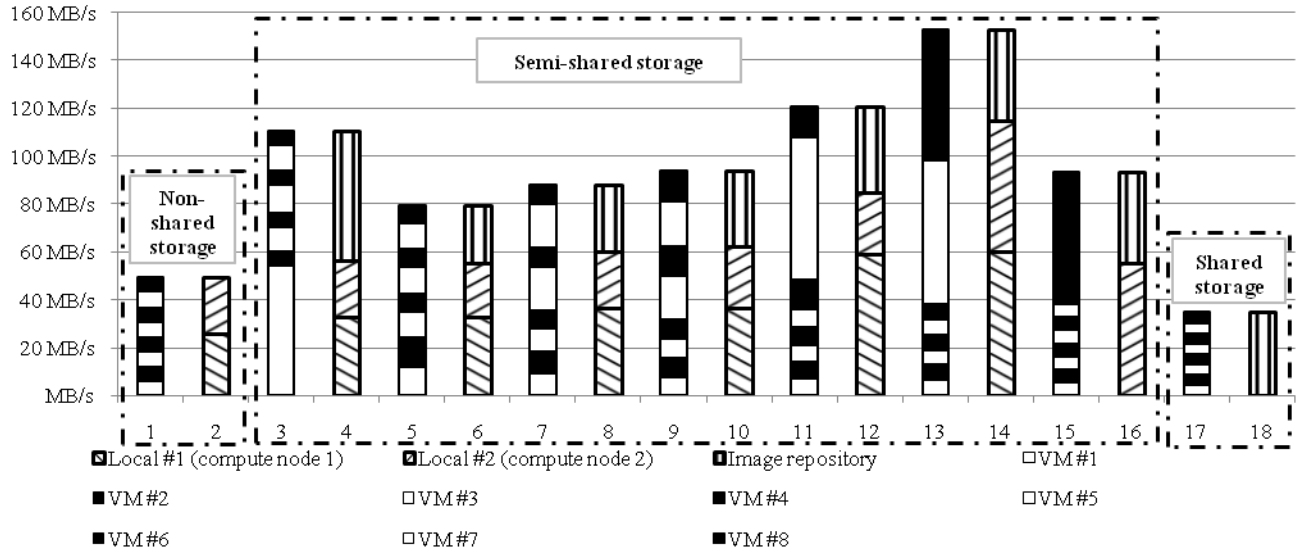


Figure 6. I/O benchmark with the image repository and two compute nodes

In the first test (column one and two), all of the VM instances are using the local storage of the compute node, which is special because it is the default distribution when non-shared storage subsystem is used. In the second test (7 VM instances running from local storage and one instance directly from image repository), the aggregated I/O performance is increased almost by three times compared to the first test, because the shared storage (image repository) was dedicated to only one VM. The last test in the batch is special as well, because all the VM instances running from image repository which is default when shared-storage subsystem is used for the OpenNebula cloud. The results show that the semi-shared storage subsystem can serve higher aggregated I/O performance than the original storage subsystem solutions in OpenNebula.

In the second test batch (Fig. 6), one image repository server (as shared storage) and two compute nodes are used and benchmarked. In the diagram, it can be seen that highest aggregated and individual (from VM instance point of view) I/O performance can be achieved if the VM instances can use exclusively a local- or the shared storage. If more computing nodes were added to test-bed, bigger performance gap would be measured between the shared- and semi-shared storage subsystems. These points to the fact that the non-persistent VM instances (running from local disk of compute nodes) are preferred and the number of the persistent VM instances (running from image repository) should be kept low if the image repository consists of a single machine. In this paper, we do not investigate and discuss the clustered or distributed storage technologies which can expand the capacities of the image repository.

VI. PRODUCTION USE CASE

Some early tests with OpenNebula showed us that I/O throughput can be problematic if VMs generate I/O intensive workloads. In order to protect our production services, we wanted to isolate production, developer and tester VMs. Separated clouds can be build for these purposes, however the utilization of the cloud components would be worse in that case. Some of our VMs require live migration which excludes to use non-shared storage subsystem for OpenNebula.

After tests were successfully running in the test-bed, semi-shared storage subsystem was put production in MTA SZTAKI. Our second cloud installation has 64 CPU cores, 152GB RAM and ~5 TB storage. Usually, there are 40-60 VM instances are running concurrently. ~10 instances of them are in production, about 20 instances are used by developers and the others are running for testing purposes. Production VMs are using only persistent disk images and the testing VMs are using only non-persistent disk images. (Developers are using both of them.) With this distribution, we managed to solve the high utilization of our resources without compromising the production services.

VII. FUTURE WORK

For IOPS-critical server workload, flash based storages are preferred to use, like SSDs or traditional DRAM [10]. We already have performed some experiments with VMs running in DRAM. We considered expanding OpenNebula with DRAM based storage solution and combined it with semi-shared storage subsystem as well. Going forward, we are planning to explore different file-system solutions for image repository. Ceph [11] could enhance scalability and provide software based redundancy for image repository of OpenNebula. In this paper, we did not investigate the I/O performance from a networking point of view, however we plan to explore the effects on the network by using different types of storage subsystem for OpenNebula.

VIII. CONCLUSION

In this paper, the performance of virtualized I/O subsystems was discussed, which is one of the most considerable limitations of cloud infrastructures. The investigation is focused on OpenNebula and its storage subsystem solutions. In this cloud middleware, we experienced scalability and I/O throughput problems. To relieve the problem, OpenNebula provides distributed storage option however fast VM deployment and live migration features are lost with that option. We presented the semi-shared storage subsystem that is able to achieve higher I/O throughput than other storage solutions do in OpenNebula by using central and local resources at the same time. Finally, test results and the production use case showed that we managed to expand I/O performance related bottlenecks in OpenNebula.

ACKNOWLEDGMENT

The authors would like to thank Márk Gergely, Péter Kotcauer, Zsolt Németh and Gábor Kecskeméti for their input and comments that helped to shape and to improve this paper. This work is supported in part by the EDGI EU FP7 project (RI-261556).

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging itplatforms: Vision, hype, and reality for deliveringcomputing as the 5th utility. Future GenerationComputer Systems, 2009.

[2] P. Mell and T. Grance. NIST definition of cloud computing. National Institute of Standards and Technology. Oct. 2009.

[3] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster. Virtual Infrastructure Management in Private and Hybrid Clouds. IEEE Internet Computing, vol. 13, no. 5, pp. 14-22, Sep. 2009.

[4] V. O. Póserné: Comparing the webservers of the opensource and the closed source operation systems, in Proc of the 5th International Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania, 2009, pp. 169-172

[5] D. Ghoshal, R. S. Canon and L. Ramakrishnan. I/O Performance of Virtualized Cloud Environments. DataCloud-SC '11, Nov. 2011.

[6] J. G. Hansen and E. Jul. Lithium: virtual machine storage for the cloud. In SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing, pp. 15–26, New York, NY, USA, 2010.

 ISBN: 978-1-61208-216-5

[7] J. Ousterhout , P. Agrawal , D. Erickson , C. Kozyrakis , J. Leverich , D. Mazières , S. Mitra , A. Narayanan , G. Parulkar , M. Rosenblum , S. M. Rumble , E. Stratmann and R. Stutsman. The case for RAMClouds: scalable high-performance storage entirely in DRAM, ACM SIGOPS Operating Systems Review, v. 43 n. 4, Jan. 2010.

[8] Sheepdog project. http://www.osrg.net/sheepdog/.

[9] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. KVM: the Linux virtual machine monitor. In Ottawa Linux Symposium, 2007.

[10] A. M. Caulfield , L. M. Grupp , S. Swanson. Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. Proceeding of the 14th international conference on Architectural support for programming languages and operating systems, Washington, DC, USA, Mar. 2009.

[11] Sage Weil, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, Carlos Maltzahn, Ceph: A Scalable, High-Performance Distributed File System, Proceedings of the 7th Conference on Operating Systems Design and Implementation (OSDI '06), Nov. 2006.

 ISBN: 978-1-61208-216-5

A Fast Virtual Machine Storage Migration Technique Using Data Deduplication

Kazushi Takahashi and Koichi Sasada
Graduate School of Information Science and Technology
The University of Tokyo
Dai-building 1301, Sotokanda 1-18-13,
chiyoda-ku, Tokyo, Japan 101-0021
Email: {kazushi, ko1}@rvm.jp

Takahiro Hirofuchi
National Institute of Advanced Industrial
Science and Technology (AIST)
Central 2, Umezono 1-1-1, Tsukuba, Japan 305-8568
Email: t.hirofuchi@aist.go.jp

***Abstract*—In this paper, we proposed a fast virtual machine (VM) storage migration technique. Virtual machine storage migration can migrate entire VM states to other hosts, including VM disk images. It is widely used for cross-data-center load management. However, VM disk images generally have large file sizes (typically 1-30GB). Thus, storage migration was time-consuming. To address this problem, we introduce the deduplication technique into traditional VM storage migration. We focused on the fact that it is possible to return VMs from the new VM hosts. For example, a VM first migrates from host A to host B. Next, the VM returns from host B to host A. There will then be additional reusable disk pages in host A. Consequently, to expedite the operation, we only return to host A the disk pages that have been updated while in host B. We implement this idea to a QEMU/KVM (kernel virtual machine). To track the reusable disk pages, we developed a DBT (Dirty Block Tracking) mechanism and a new diff format to store the tracking result. In this paper, we discuss the design and implementation of our prototype. Our technique successfully reduced the transfer time for storage migration from 10 minutes to about 10 seconds in some practical workloads.**

***Keywords*-** *Virtual Machine Monitor; VMM; VM Live Migration; VM Storage Migration*

I. INTRODUCTION

Live migration involves the migration of virtual machines (VMs) from one physical host to another even while the VM continues to execute. Live migration has become a key ingredient for data center management activities such as load balancing, failure recovery, and system maintenance.

Recently, a new live VM migration mechanism has been developed. Live VM (virtual machine) storage migration allows us to move entire VM states, including VM disk images to another physical machine without stopping the VM. Traditional VM live migration mechanisms transport only machine states such as CPU registers and NIC registers, and require file sharing systems such as NFS [1] and iSCSI, to share VM disk images between the source and destination hosts. However, VM storage migration achieves VM live migration without the file sharing systems.

VM storage migration enables flexible VM deployment across physical computers. First, VM storage migration is widely used in many data centers because it does not require file sharing systems. For example, cross-data-center load management, and VMs can evacuate quickly to other data centers in other regions when the data center is unavailable due to maintenance. Second, we believe that storage migration will be used for personal computer environments in future. Some researchers [2] have studied virtual machine migration for personal computing. However, since they used traditional virtual machine migration, as opposed to storage migration, the system forces users to use file sharing systems that have network connections. However, by introducing VM storage migration, VMs become portable without the need for file sharing systems. VM storage migration has significant potential for future computing.

However, VM disk images are large (typically 1-30GB in size) and VM storage migration is time-consuming. Even when using fast gigabit network environments, the transfer time was significant. For example, a VM which has a disk size of 20 GB requires about 10 minutes in a 1Gbps LAN environment. This is unacceptable by users because it is inconvenient.

We proposed a fast VM storage migration mechanism using *data deduplication* to reduce the transfer time and to reduce the volume of transferred data. Our fast VM storage migration works as follows: assume that two physical hosts, A and B, are located in different regions. Initially, a VM is migrated from source host A to destination host B. Thereafter, the VM returns to A from B. We can then leverage disk pages which were not updated while in host B to reduce the transfer time and the volume of transferred data. Although the initial transfer cost from A to B is large, VM migration will be faster in the subsequent round. Consequently, we can achieve faster VM storage migration than traditional VM storage migration techniques.

We implemented this idea to a QEMU/KVM (kernel virtual machine) to develop a prototype. QEMU/KVM already has a storage migration mechanism. Thus, we improved the storage migration mechanism to support data deduplication by tracking all disk-writing operations on the disk by the VM, and to identify the disk pages that are reusable. To track the disk pages that are reusable, we developed a *DBT (Dirty Block Tracking)* mechanism and new diff format to store the tracking result.

 ISBN: 978-1-61208-216-5

We examined our prototype on several machines which have difference workloads. After developing our prototype, we compared our storage migration with traditional storage migration to determine the difference in the volume of data transferred and time taken are reduced. Our result shows the effectiveness of our deduplication mechanism for fast storage migration.

This paper is based on our previous work which has been presented at a local symposium in Japan (written in Japanese) [3] In this paper, we have substantially improved our previous work by conducting further experiments and making more detailed discussions which is mentioned in Section V.

II. Related Work

There have been related work that enables VM live-migration with VM disk images. Studies by Luo et al. [4] and Bradford et al. [5] enable VM storage migration in Xen [6] using their special back-end drivers, and they achieve VM live-migration without a file sharing system. However, they did not discuss re-using disk pages, and if there are reusable disk blocks in the destination host, the hypervisor can perform data deduplication to reduce the volume of data and time. Therefore, their research is somewhat different from ours. Hirofuchi et al. [7] implemented a NBD (Network Block Device) protocol server. In this system, a user use `/dev/nbd0` to enable VM storage migration without the file sharing system. When the user wants to boot a VM, he executes the VM with `/dev/nbd0` instead of a normal VM storage file. When VM storage migration occurs, `/dev/nbd0` copies the disk image to the destination host. This is also different from our approach since we focus on the application of data deduplication to reduce transfer data and time while Hirofuchi did not discuss data deduplication for VM storage migration, as is the case with our approach.

VMFlocks Project [8] proposed a data deduplication mechanism for VM storage migration between data centers. This project was implemented as a user-level file system on a host OS. This user-level file system inspects VM disk images without hypervisors, and it deduplicates data disk blocks of VM disk images using fingerprint algorithms such as SHA-1 [9] Our research is different in that we implement our deduplication mechanism by modifying a hypervisor. On the other hand, VMFlocks is implemented as a user-level file system on a host OS. Our approach using the DBT within a hypervisor is beneficial since it can leverage raw level hardware commands such as the ATA TRIM command to optimize disk pages, For example, garbage collection for disk pages.

Sapuntzakis et al. [10] proposed several techniques for speeding up virtual machine migration in various user scenarios. More specifically, they proposed a tree structured based VM disk management system as follows: First, a user creates a root VM storage image in a destination host. Second, the user checks out a VM image from the root VM storage image on a source host. Finally, the approach reduces the amount of data transferred by exploiting similarities between the transferred image on the source host and the root image on the destination host. Our proposal is different since we use a simpler approach in which the DBT records dirty block information into a simple dirty map in order to reduce the amount of transferred data.

Intel Research proposed ISR (Internet Suspend/Resume) techniques [11] [12] ISR is a cold VM migration technique and is explained as follows: First, before a VM transfer takes place, a user suspends the VM. Next, the suspended VM image that includes a VM disk image migrates to a destination host. Finally, after migrating the VM image, the VM resumes in the destination host. Kozuch et al. [11] and Tolia et al. [12] proposed a VM storage migration technique using Coda [13], which is a traditional distribution file system. To enable VM storage migration, they store the entire VM status (including VM disk images) to Coda, and frequently download the VM status on the destination host on demand. In fact, this approach supports VM cold migration. Coda has an excellent caching and reading prediction mechanism. Thus, a user can eventually obtain the entire VM status without having a network connection to a Coda server. However, a constant network connection is required to access the Coda file system until file caching is filled. On the other hand, our approach only requires a temporary network connection while transferring the VM. Also, our approach supports VM live-migration.

VMware's VMware Storage VMotion [14][15] supports VM migration including VM file images. Unlike our approach, they do not leverage re-usable disk pages on a destination host to reduce transferred data or time.

The Shrinker project [16] focuses on reducing the transferred data to minimize transfer time. To reduce the transfer time, they share VM memory pages between a destination host and a source host using a distributed hash table (DHT). They focus on using memory pages to reduce migration time while we focus on disk pages.

III. Motivation

In this paper, we proposed fast VM storage migration using data deduplication. As mentioned above, our idea would be effective if VMs are transferred between specific physical hosts.

In this section, we discuss the kinds of scenarios in which VMs can be transferred between specific physical hosts, and the kinds of situations for which our deduplication mechanism would be suitable.

In today's cloud computing environments, we believe that the transfer of VMs between specific physical hosts is very likely. Two scenarios are as follows:

 ISBN: 978-1-61208-216-5

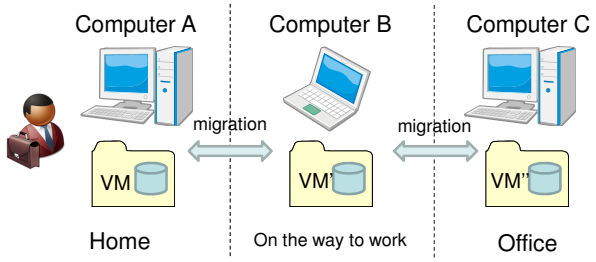


Figure 1. Live-migration for personal use.

A. VM storage migration for personal computer environment

Recently, the use of computing systems that provide VMs instead of physical computers has increased. In this case, users cannot use physical computers directly, but can use virtual machines that are isolated from each other. We refer to such a system as a *personal virtual machine system*. Personal VM systems are more convenient than traditional physical computers and have the following benefits: First, we can easily store entire VMs on portable storage devices such as USB memory, and it is therefore portable. Personal computer environments can therefore be used at any location. Secondly, Personal VM systems can provide highly secure computing, because backups of personal computing environments can easily be made as virtual machine images. If a VM is contaminated by a malicious program, the computing environment can be quickly recovered using a backup VM image. As previously mentioned, personal VM systems have many benefits. Moka Five [17] has developed a personal VM system which is widely used, and it is believed that personal VM system usage will rapidly spread.

We believe that by introducing storage migration technologies to personal VM systems, we can realize more convenient computing environments. Figure 1 shows an image of VM storage migration in a personal VM system. This image was inspired by Shivani Sud et al. [2], whose work we summarized as follows:

> Jane uses her home computer to check her email and reviews a presentation she needs to deliver later that morning. As the day progresses, she seamlessly migrates her work environment from her home PC to her mobile device before leaving home. While traveling she continues reviewing the presentation, adding notes as she rides the subway to work. Soon it is time for her to dial into a teleconference. On reaching her desk, her work environment seamlessly migrates from her mobile device to the office PC, and she can now use the office PC to continue reviewing the presentation, while she continues her teleconference from her mobile device.

However, as mentioned above, VM disk images are large (typically 1-30GB in size). Consequently, VM storage migration is time-consuming. For example, a 20GB disk image, requires about 10 minutes in a gigabit LAN environment. Traditional VM live-migration including storage migration focuses mainly on reducing the down-time in extreme cases. In fact, pre-copy and post-copy live-migration algorithms are designed in an effort to reduce the down time. However, in the assumed personal VM, it is important to minimize the entire migration time as opposed to the down-time. This is because for personal VM live-migration, the most important thing is that when a user wants to migrate a VM to another device, the VM moves immediately to the another device without subsequent network communication. Our fast storage migration using data deduplication can reduce the entire transfer time, and is therefore considered to be effective.

B. Follow the "moon" data center access model

To reduce the electricity bill and cooling cost in data centers, some companies have proposed a strategy to deploy server computers to regions in the world which have nightfall. Using this approach, it is possible to maximize the use of inexpensive off-peak electricity and lower temperatures. By taking this approach, VMs are frequently deployed in the data centers which are located in nighttime regions. In fact, a VM may be migrated to a host to which it has previously been deployed. By introducing our fast storage migration, we can reduce transfer time. Additionally, in today 's multi-tenant cloud computing environments, many customers' VMs are consolidated into a single data centers. If many VMs concurrently migrate to a night-time data center with their large disk images, the bandwidth of the data center may become saturated. However, our fast storage migration approach can reduce the volume of transferred data for VM migrations.

As previously mentioned, in today's cloud computing environments, VM would alternate between particular physical hosts. Thus, our fast storage migration using data deduplication is an effective way of achieving this goal.

IV. System Overview

We implement a simple prototype for VM storage migration mechanisms using data deduplication. More precisely, we implement DBT (Dirty Block Tracking) on a hypervisor. DBT is a tracking module for the writing of a guest OS on a VM. In order to achieve fast storage migration, DBT works as follows : First, a VM disk image is divided into fixed chunks using DBT. Next, DBT tracks all written disk page blocks. DBT leverages a bitmap to manage the disk pages that have been updated by the guest OS. Then, when VM live-migration from a host A to a host B occurs, the hypervisor executes normal slow VM-live migration if there are no available reusable disk page blocks in the destination

 ISBN: 978-1-61208-216-5

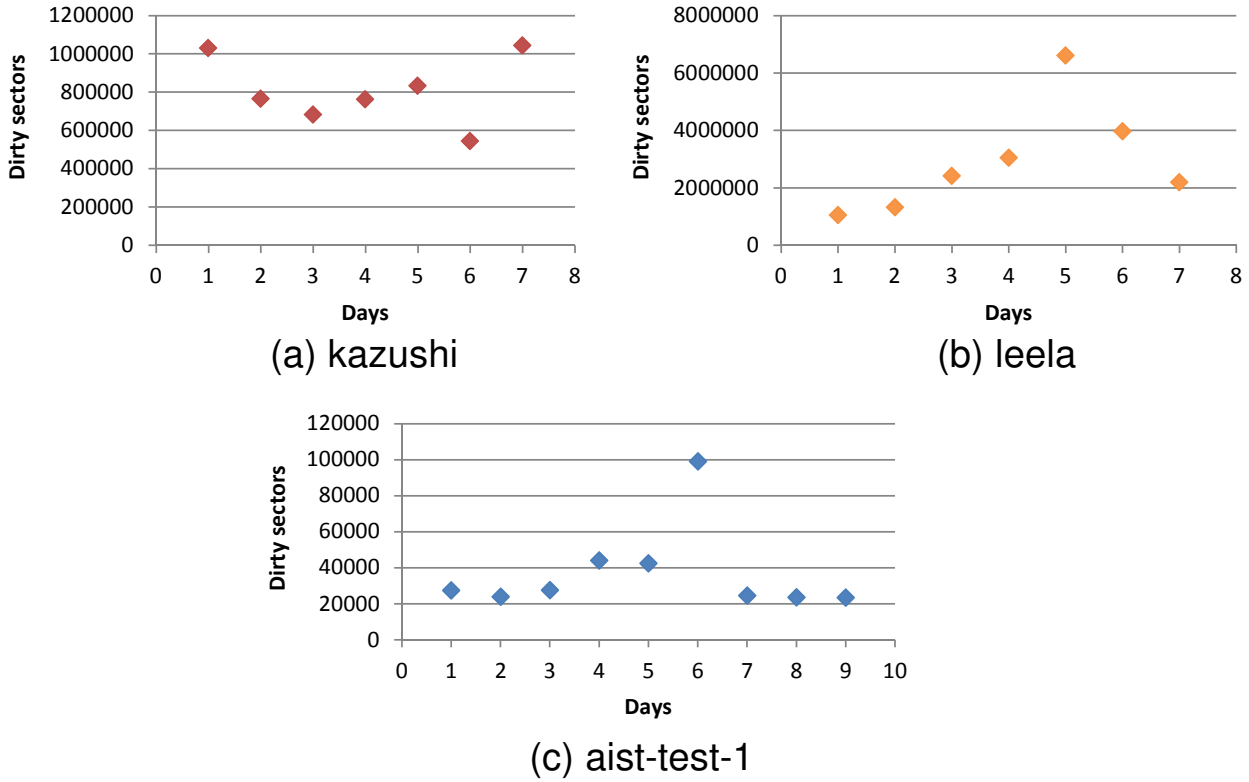


Figure 2. Three machines and three different workloads.

host B. Entire VM disk images are translated. On the other hand, if there are reusable disk pages in the destination host B, only dirty pages which have been updated on the source host A are transferred. With this mechanism, we can achieve our fast VM storage migration using data deduplication.

V. Pre Examination

To examine the efficiency of deduplication transfer for storage migration, we investigate for several days the number of disk pages that are updated in the average daily operation of a computing environment. We examined three computers which have different workloads. Three computer setups are shown in Table I. With the exception of aist-test-1, all of the machines are physical machines. Kazushi consisted of a hard disk drive that is 300GB in size, and which executes Ubuntu 11.04 (32bit) with the ext4 file system format. Leela consisted of a solid state drive that was 160GB in size, and which executes Windows 7 (32bit) with the NTFS format. Aist-test-1 consisted of a hard disk drive that was 60GB in size, and which executes CentOS 6.2 (64 bit) with the ext4 file system format.

Aist-test-1 is a server which works as a web-based groupware. Kazushi is a laptop computer which executed web-browsing operations including the playback of YouTube videos and text editing. Leela is also a laptop computer which executed only web-browsing operations, including the playback of YouTube videos.

Table I. Pre examination environments

Name	Type	Size	OS	FS
(a) kazushi	HDD	300GB	Ubuntu 11.04 (32bit)	ext4
(b) leela	SSD	160GB	Windows 7 (32bit)	NTFS
(c) aist-test-1	HDD	60GB	CentOS 6.2 (64bit)	ext4

Our results are shown in Figure 2. For all of the test machines, we find that changes were made to only a few disk pages in the entire physical disks. First, in the case of kazushi, a maximum of only 0.49 GB disk space was updated, out of 300 GB. This accounts for only 0.16% of the 300 GB disk. A minimum of, only 0.25 GB of the 300 GB disk space was updated. This accounts for only 0.08% of the disk. Secondly, in the case of leela, a minimum of only 0.49 GB of the 160 GB disk space was updated, accounting for only 0.33% of the disk, while a maximum of only 3.14 GB of the 160 GB disk space was updated, accounting for only 2.11% of the disk. Thirdly, for aist-test-1, a minimum of only 0.01 GB of the 60 GB disk space was updated, accounting for only 0.07% of the disk, while a maximum of 0.04 GB of the 60 GB disk space was updated, accounting for only 0.31% of the virtual disk.

 ISBN: 978-1-61208-216-5

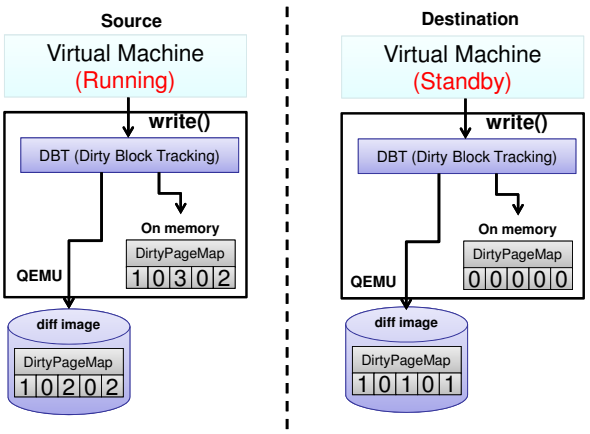


Figure 3. Our prototype system overview.

This examination shows that our deduplication transfer mechanism for virtual machine storage migration is efficient.

VI. Design

In the previous section, we showed the deduplication system for storage migration is efficient. Thus, we designed a system for the deduplication of storage migration. An overview of our system is shown in Figure 3.

1) Dirty Block Tracking: DBT is a mechanism that traces entire disk pages written by a guest OS, and records the tracking result into the dirty map structure. DBT within a hypervisor hooks the writing by a guest OS. DBT simply divides entire disk images at block boundaries, and allocates 8 bits of space for each block. This 8 bit space is updated by DBT with a generation number, which is an identifier for VMs. When a VM moves to another host by storage migration, this value is increased. Consequently, we can identify the disk pages that should be transferred when storage migration takes place.

2) Diff Image Structure: Diff image is a new VM image format that supports deduplication for VM storage migration by managing the structured dirty map with DBT. The diff image structure is shown in Figure 4. The diff image consisted of G, which is the generation number, S, which is the seed number, and the freeze flag, which indicates whether or not the VM image is able to boot, and the dirty page map, which indicates which blocks have been updated by the guest OS. The generation number is increased when the VM migrates to another host, and this number is initially one. The seed number is a unique number which is allocated when the disk image is created.

3) Migration Algorithm: The migration procedures are as follows:

(a) When a diff disk image is created, the diff structure is initialized. The seed number is a unique number, the generation number is 1, the freeze flag is 0, and the dirty page map is all zeros.

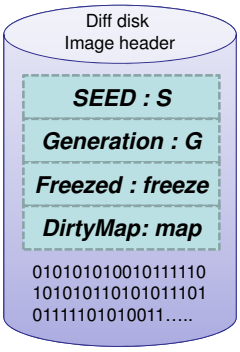


Figure 4. The structure of diff disk image.

(b) A guest OS writes to the disk. Using DBT, all writings by the guest are tracked and recorded. The blocks updated by the guest are tracked and recorded. The recording is conducted by writing the generation number into the dirty page map.

(c) The guest OS migrates to a host A. This is a slow storage migration process because it is an initial transfer. The generation number G is incremented when this migration is completed.

(d) A host B, which is the destination, acquires ownership of the guest OS. Thus, the VM image in the host A is frozen, and it is temporarily not bootable.

(e) The guest OS writes to the disk in the host B. DBT records the disk pages that are updated in the generation number G.

(f) The guest OS migrates the host B to host A. Now, fast deduplication storage migration is possible. DBT compares the generation number in the host A with generation numbers in the dirty map in the host B. If a generation number in the dirty map in host B is greater than the generation number G in host A, the disk block which corresponds to the generation number in the dirty map is transferred.

A. Discussion

We also considered another approach to achieve data deduplication for storage migration. In our prototype, we proposed a simple method to track disk block writing with dirty maps using DBT. On the other hand, other methods using fingerprints such as SHA-1 [9] and Rabin-fingerprint [18] are available. In fact, as in rsync [19], we can use the *Compare-by-hash* [20], [21] method to achieve data deduplication.

According to Jacob et al. [22], hand-optimized SHA-1 implementation, running on a single Intel Core-2 CPU core is able to hash more than 270 MB of data per second, which is almost enough to saturate the full write bandwidth of three SATA disk drives. Thus, although DBT calculates SHA-1 hash when disks are written to by a guest OS, the guest OS

 ISBN: 978-1-61208-216-5

does not incur loss of speed.

However, our goal in this paper is to show that our deduplication VM storage migration method is practical. Thus, in this paper, we adopt a more simple method which uses the dirty page map with generation numbers. As described later, we believe that the use of SHA-1 hash has some side effects. We now plan to develop a deduplication VM transfer mechanism using SHA-1

VII. Implementation

We implement the previously mentioned DBT and the diff format to Linux/KVM (QEMU) [23], [24] We divide entire VM image files into chunks. DBT constructs a dirty page map using an 8-bit space for each of the chunks. Currently, for the bitmap, one chunk is 2,048 sectors, where one sector is 512 bytes. This is a constant value for QEMU. A bitmap that is in 4 Kbytes is generated for 4 GB. VM disk images. Although a VM image that is 20 GB is generated, a bitmap that is about 20 KB is generated. Thus, the bitmaps do not place additional stress on the physical hard disk drive. This bitmap is deployed in memory when the guest OS executes on the VM. When the guest OS exits, the bitmap is updated on the diff image.

Additionally, we implement two APIs to communicate the dirty block information between the disk driver layer and the live migration mechanism layer in QEMU, and to increment the generation number when storage migration is completed. QEMU implementation is a structured design, For example, vmdk, qcow, and qcow2, which are formats of the QEMU's disk images, are updated as device drivers in QEMU. In order to develop new QEMU disk formats, developers implement only the specific handlers in QEMU. Developers, who are desirous of adding new QEMU disk image formats, can implement a new QEMU format by implementing only the specific callback handlers. BlockDriver structure in block_int.h source header file of QEMU defines the callback handlers to implement QEMU disk image in QEMU. We add two callback handlers to QEMU because there are no APIs to increment the generation number and, to communicate the dirty map between the disk driver layer and live-migration implementation layer.

VIII. Evaluation

We evaluate and analyze the impact of our deduplication migration mechanism. First, we examine whether or not writing to the disk has slowed. Secondly, we conduct measurements for the speed and efficiency of the data transfer using several machines which have different workloads.

A. Evaluation for the DBT cost

In this subsection, we show that DBT does not incur a loss of speed by tracking the disk writing operations. Our setup consisted of a ThinkPad X220 laptop computer that was booted up with an Intel Core i5-2430M @ 2.40GHz, and which has 4 GB of memory. This machine is attached to a Seagate 7200rpm HDD with a cache memory of 16 MB.

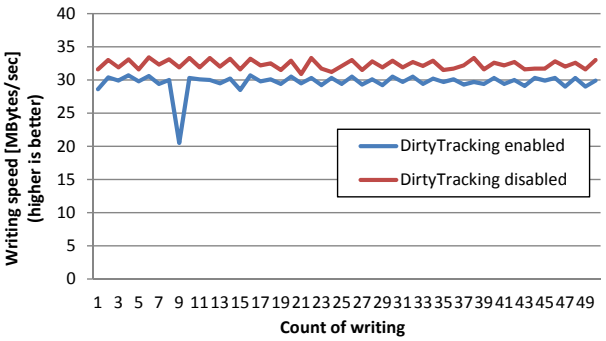


Figure 5. The result of writing benchmark.

To realize the impact of DBT on the performance, we compare two benchmark results: First, the write speed of the diff format with DBT. Secondly, the write speed for the raw-format, which is the primitive VM format for QEMU. We measure only the write cost and not the read cost because DBT only works with guest disk writing. We use UNIX dd commands to measure the disk write cost. Using dd, we write 1 MB blocks 100 and 50 times.

The evaluation result is shown in Figure 5. The x-axis indicates the write speed in Mbytes/sec, and The y-axis indicates the writing count. We find that the hypervisor without DBT achieved 32.304 Mbytes/sec, and the hypervisor with DBT achieved 29.656 Mbytes/sec. Because DBT leads to a decrease in the write speed of only 8%, it is not thought to be significant.

B. Migration speed with different workloads

In this subsection, using a series of benchmarks, we show the speed and the efficiency of data transfer for migration.

We assume practical workloads as follows: First, a user downloads an MS Office power point file, views, edits, and saves the file. Secondly, the user downloads from a Japanese literary website a literary creation "Kokoro" by Souseki Natume, who is a famous Japanese scholar in the field of Literature. The user then views, edits, and saves the information. Next, the user views the video on YouTube with a Firefox web-browser. Fourthly, the user downloads a 3 MB PDF file and views it. We run these practical workloads on both Windows 7 (32bit) and Ubuntu 11.04 (32bit) operating systems. After each user performs these actions for five minutes on both virtual machines, we conduct our deduplication procedure for both VM storage migration and normal VM storage migration. Finally, we compare the times taken for our deduplication storage migration and normal VM storage migration.

The source host consists of a ThinkPad X60 laptop computer booted with an Intel Core Duo CPU T2400 @ 1.83 G Hz with 2 GB of memory. The destination host consists

 ISBN: 978-1-61208-216-5

Table II. Storage migration measurement result on Ubuntu 11.04 desktop (32bit)

	No deduplication	PDF	Presentation	YouTube	Kokoro
Whole Migration Time (sec)	919.358	29.139	30.141	28.720	25.900
Transferred size (MBytes)	—	101	104	111	90

Table III. Storage migration measurement result on Windows 7 Professional (32bit)

	No deduplication	PDF	Presentation	YouTube	Kokoro
Whole Migration Time (sec)	991.044	89.028	68.892	78.450	86.703
Transferred size (MBytes)	—	933	613	927	907

of a DELL LATITUDE D630 laptop computer booted with an Intel Core 2 Duo T7300 @ 2.0 GHz with 2 GB of memory. Both computers are connected in a 1 Gbps LAN environment.

The result for Ubuntu 11.04 (32 bit version) is shown in Table II. The longest migration time was 30.141 seconds for the presentation benchmark. On the other hand, the best migration time was 25.900 seconds for the Kokoro benchmark. We found that our approach was able to reduce the migration time. All of the benchmark results lasted about 10 minutes. However, they lasted only about 10 seconds after introducing our approach. We also found that using our method, the volume of transferred data had been reduced from 20 GB to several hundreds of megabytes.

Next, the result for Windows 7 Professional (32 bit version) is shown in Table III. When compared with the result for Ubuntu, in the case of Windows, the number of dirty disk blocks was larger. and the volume of data that was transferred was also larger. The longest migration time was 89.028 seconds for the PDF benchmark. On the other hand, the best migration time was 68.892 seconds for the presentation benchmark. Windows was shown to generate a greater number of dirty disk blocks than Ubuntu. Additionally, we found that the Presentation benchmark consumed the most migration time in the case of Ubuntu while the PDF benchmark consumed the most migration time in the case of Windows. The best time in the case of Ubuntu was 28.720 seconds for the YouTube benchmarks, while for Windows, the best time was 68.892 seconds for the presentation benchmark. All of the benchmarks results in Windows achieved a migration time of about 1 minute.

It was found that the introduction of the deduplication for storage migration led to greater efficiency. For all of the benchmarks, we were able to achieve faster storage migration than traditional storage migration techniques.

IX. Future Work

As mentioned above, we can use fingerprint algorithms such as SHA-1 and Rabin fingerprint to deduplicate VM disk pages. In fact, we divide the VM image files into chunks, and calculate fingerprints for all chunks. VM images are transferred, we can compare fingerprints in the source VM image with those in the destination VM image to exploit deduplicated VM disk blocks. This approach also provides deduplication on local VM disk storage to reduce the volume of local storage data. Although this approach somewhat complicated, it is better than our bitmap approach. Therefore, we will implement the deduplication system using a fingerprint algorithm.

X. Conclusion

In this paper, we proposed a fast VM storage migration technique using data deduplication. In the pre-examination results, we show that data deduplication for fast VM is an effective approach because only a few disk blocks are usually updated in daily computing operations. Thus, we implement a prototype that realizes fast VM storage migration using data deduplication. For all of the benchmarks, we achieve storage migration that is faster than traditional storage migration.

References

[1] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, and D. Noveck, “Network file system (nfs) version 4 protocol,” 2003.

[2] S. Sud, R. Want, T. Pering, K. Lyons, B. Rosario, and M. X. Gong, “Dynamic migration of computation through virtualization of the mobile platform,” in *MobiCASE*, 2009, pp. 59–71.

[3] K. Takahashi and K. Sasada, “A fast vm transport mechanism that consider generations with disk dirty page tracking,” in *53th Programming Symposium*. IPSJ, January 2011, pp. 37–45.

[4] Y. Luo, B. Zhang, X. Wang, Z. Wang, Y. Sun, and H. Chen, “Live and incremental whole-system migration of virtual machines using block-bitmap.” in *CLUSTER’08*, 2008, pp. 99–106.

[5] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, “Live wide-area migration of virtual machines including local persistent state,” in *Proceedings of the 3rd international conference on Virtual execution environments*, ser. VEE ’07. New York, NY, USA: ACM, 2007, pp. 169–179. [Online]. Available: http://doi.acm.org/10.1145/1254810.1254834

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen and the art of virtualization,” *SIGOPS Oper. Syst. Rev.*, vol. 37, pp. 164–177, Oct. 2003. [Online]. Available: http://doi.acm.org/10.1145/1165389.945462

 ISBN: 978-1-61208-216-5

[7] T. Hirofuchi, H. Ogawa, H. Nakada, S. Itoh, and S. Sekiguchi, "A live storage migration mechanism over wan for relocatable virtual machine services on clouds," *Cluster Computing and the Grid, IEEE International Symposium on*, vol. 0, pp. 460–465, 2009.

[8] S. Al-Kiswany, D. Subhraveti, P. Sarkar, and M. Ripeanu, "Vmflock: virtual machine co-migration for the cloud," in *Proceedings of the 20th international symposium on High performance distributed computing*, ser. HPDC '11. New York, NY, USA: ACM, 2011, pp. 159–170. [Online]. Available: http://doi.acm.org/10.1145/1996130.1996153

[9] National Institute of Standards and Technology, *FIPS PUB 180-1: Secure Hash Standard*, Apr. 1995, supersedes FIPS PUB 180 1993 May 11. [Online]. Available: http://www.itl.nist.gov/fipspubs/fip180-1.htm

[10] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum, "Optimizing the migration of virtual computers," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 377–390, December 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844163

[11] M. Kozuch, M. Satyanarayanan, T. Bressoud, and Y. Ke, "Efficient state transfer for internet suspend/resume," *Intellectual Property*, no. May, 2002.

[12] N. Tolia, N. Tolia, T. Bressoud, T. Bressoud, M. Kozuch, M. Kozuch, and M. Satyanarayanan, "Using content addressing to transfer virtual machine state," Tech. Rep., 2002.

[13] M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel, David, and C. Steere, "Coda: A highly available file system for a distributed workstation environment," *IEEE Transactions on Computers*, vol. 39, pp. 447–459, 1990.

[14] VMware, Inc., "VMware Storage VMotion: Non-disruptive, live migration of virtual machine storage," http://www.vmware.com/products/storage-vmotion/

[15] A. Mashtizadeh, E. Celebi, T. Garfinkel, and M. Cai, "The design and evolution of live storage migration in vmware esx," in *Proceedings of the 2011 USENIX conference on USENIX annual technical conference*, ser. USENIXATC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 14–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002181.2002195

[16] P. Riteau, C. Morin, and T. Priol, "Shrinker: Efficient Wide-Area Live Virtual Machine Migration using Distributed Content-Based Addressing," INRIA, Research Report RR-7198, Feb. 2010. [Online]. Available: http://hal.inria.fr/inria-00454727/en/

[17] MokaFive, "MokaFive Player," http://www.moka5.com/

[18] M. O. Rabin, "Fingerprinting by random polynomials." TR-CSE-03-01, Center for Research in Computing Technology, Harvard University, Tech. Rep., 1981.

[19] A. Tridgell and P. Mackerras, "The rsync algorithm," Australian National University, Department of Computer Science, Technical Report TR-CS-96-05, Jun. 1996, http://rsync.samba.org.

[20] J. Black, "Compare-by-hash: a reasoned analysis," *Proc 2006 USENIX Annual Technical Conference*, pp. 85–90, 2006. [Online]. Available: http://www.usenix.org/event/usenix06/tech/full_papers/black/black.pdf

[21] V. Henson and R. Henderson, "Guidelines for using compare-by-hash," 2005.

[22] J. G. Hansen and E. Jul, "Lithium: virtual machine storage for the cloud," in *Proceedings of the 1st ACM symposium on Cloud computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 15–26. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807134

[23] F. Bellard, "Qemu, a fast and portable dynamic translator," in *ATEC'05: Proceedings of the USENIX Annual Technical Conference 2005 on USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2005, p. 41. [Online]. Available: http://portal.acm.org/citation.cfm?id=1247401

[24] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "KVM: the Linux Virtual Machine Monitor," in *Proceedings of the Linux Symposium*, 2007, pp. 225–230.

 ISBN: 978-1-61208-216-5

Network Performance-Aware Virtual Machine Migration in Data Centers

Jun Chen, Weidong Liu, Jiaxing Song
Department of Computer Science and Technology
Tsinghua University
Beijing, China
chenjun09@mails.tsinghua.edu.cn
{liuwd, jxsong}@tsinghua.edu.cn

***Abstract*—Virtual machine (VM) consolidation and migration technology in data centers greatly improve the utilization of the server resource. While the previous work focuses on how to use VM migration to balance physical host utilization or optimize energy consumption, little attention has been given to network performance factors, such as link traffic load and inter-traffic between VMs in data centers. In this paper, we present MWLAN (Migration With Link And Node load consideration), a novel automatic data center VM migration system that can detect hotspots (e.g., network congestion and physical host over-loaded) and dynamically remap VMs to improve the network performance. The VM migration approach proposed in MWLAN can efficiently balance the network link load and relieve the local data center network congestion as well as considering physical host constraints. Moreover, experimental evaluations indicate that the proposed approach reduces the packet loss by up to 50% and improves the average application TCP transfer rate by up to 24% compared to the other approaches when the data center network overloaded.**

Keywords-virtualization; virtual machine migration; data center; load balancing

I. INTRODUCTION

With the development of technology, virtualization has been widely used in data centers. It allows a single physical host to run multiple isolated virtual machines. When a physical host is overloaded, virtual machine migration can dynamically remap virtual machines onto physical hosts in data centers, which greatly improves physical host resource utilization. At the same time, network scalability is becoming more and more crucial in data center network system. Many new network architectures [2][3] have been proposed for data centers to solve the network problems. As the server virtualization on data centers, the VMs placed in data center physical hosts are applications or application components (multi- tier applications). There are usually high traffic rate and increasing trend towards more communication due to the inherent coupling among VMs (e.g., scientific computing, web search, MapReduce). The VMs arrive/depart dynamically and their location is not fixed. In such environments, VMs with large communication or belonging to the same application tier are very likely to be scattered into different network segments. We call it service fragmentation, which consumes large inter-node bandwidth. The research [15] shows that service fragmentation can heavily affect the data center network performance. Thus, how to schedule and place the VM to improve the data center network performance is a meaningful topic.

However, in recent years, many work focus on using virtual machines (VMs) consolidation and migration to improve the efficiency of physical host or power management in data centers. Little attention has been given to the network performance influence of VM migration in data centers.

In this paper, we present a novel migration system, MWLAN (Migration With Link And Node load consideration), a dynamic migration scheduling system in data centers. MWLAN collects the load information on physical host and switch links, detects and finds hotspots. After that it chooses a VM candidate and a physical host candidate for VM migration, taking the underlying data center network performance factors into count, as well as the physical host constraints. However, the VMs migration problem with resource constraints on physical node and link can be reduced to virtual network embedding/mapping (*VNE*) problem which is proven to be NP-complete [4]. In this paper, a heuristic algorithm is proposed to solve the migration problem efficiently. The ultimate goal is to balance the network traffic load and improve the network resource utilization while satisfying VMs and physical host resource constraints in data centers. Furthermore, the experiment results demonstrate that the proposed approach reduces the packet loss by up to 50% and improves the average application transfer rate up to 24% compared to the other approach when the data center network overloaded according to scheduling 10% VMs.

Our contributions can be summarized as follows:

- We address the problem of network link load dynamic adaption and formulate the cost of network link load in data centers in order to avoid network congestion or overload.
- We propose a novel VM dynamic migration idea by efficiently utilizing network resources as well as considering physical host constraints.
- We evaluate the proposed algorithms by simulators and the results prove that they can significantly

 ISBN: 978-1-61208-216-5

relieve network congestion and improve the traffic rate when network overloaded.

The rest of this paper is organized as follows. Section II provides some background and gives an overview of the migration system MWLAN. Section III presents our core system architecture of MWLAN. In Section IV, we evaluate the proposed methods using simulations. Then Section V discusses the related work. Finally, Section VI presents our conclusion and future work.

II. BACKGROUND

The existing data center VM migration approaches are used to eliminate the overloaded physical host, which move a virtual machine from the overloaded physical host to another underloaded one. This migration policy can balance the utilization of physical host resource. But no one considers using VM migration to balance the data center network link traffic load and prevent network performance degradation. This paper designs a data center virtual machine migration management system MWLAN. MWLAN is used in virtualized data center. Generally, a virtualized data center is composed by network and physical hosts (or server). The interconnected switches formulate the data center network [2][3], while the physical hosts are connected by data center network. One physical host can hold one or more VMs which are allocated some parts of physical host resource, such as CPU, memory. Each VM runs an application or an application component (multi-tier application). All storage is thought to be on a network file system (NFS) or a storage area network (SAN), thus, MWLAN can avoid storage migration.

More specifically, MWLAN has full knowledge of the network topology, network configuration (routing info), the switch link bandwidth, the physical host capacity and the mapping of applications to physical host. By taking a global view of routing and VM traffic demands, MWLAN can identify the load of physical host and the switch link in data centers. If a hotspot occurs (e.g., network congestion and physical host overloaded), MWLAN can use VM migration to balance the overloaded resources (e.g., physical host or links). Figure 1 shows the virtualized data center and MWLAN.

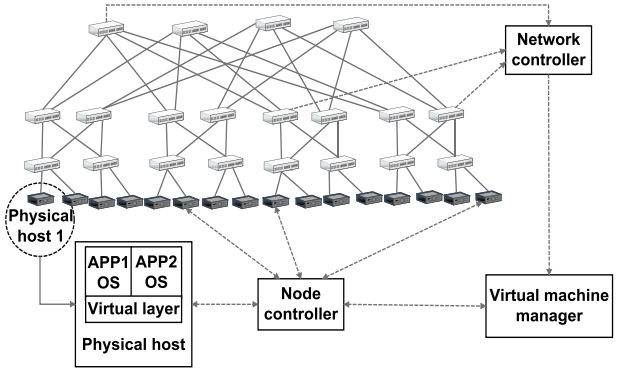


Figure 1. The virtualized data center and MWLAN architecture.

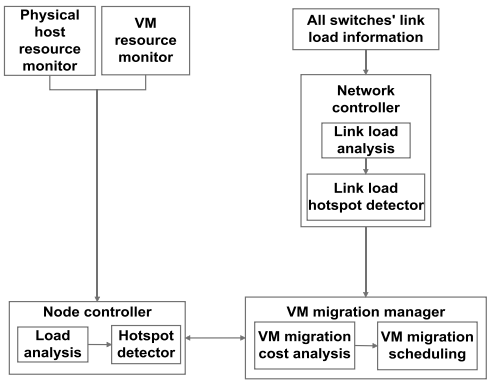


Figure 2. The MWLAN architecture.

MWLAN is consisting of three components: Node Controller, Network Controller and VM Migration Manager. Node Controller is responsible for gathering VM resource usage statistics on each physical host and VMs, doing demand estimation (physical host resource demand and bandwidth demand) and detecting physical host hotspot. Network Controller periodically gathers link bandwidth usage statistics of data center network and the routing info, and then does the link load hotspots detecting process. VM Migration Manager is responsible for choosing the migration VM candidate and the destination physical host candidate. Therefore we propose the MWLAN architecture depicted in Figure 2, the principal components and their interplay are described in more detail in the following architecture section.

III. MWLAN SYSTEM ARCHITECTURE

The below section will discuss the detail function of MWLAN's components which can be divided into four steps. First, host and network resource usage monitoring in date centers, such as physical host usage, VM resource usage and the traffic load at the switches. Next, it describes the hotspot detection. And then, demand estimation and VM migration cost analysis. Finally, VM migration schedule.

A. Monitoring

Monitoring is not only responsible for tracking the resource usage of physical host and VMs, but also gathering the link bandwidth consumed information of switches in data centers. Thus, monitoring is composed by two parts: host monitoring and network monitoring.

Host monitor runs on each physical host and VM. It gathers the host resource usage, such as the CPU usage, the transfer data rate of VMs. As shown in Figure 2, the node controller gathers all hosts' resource usage information from host monitor.

Network monitoring is running on each switch in data centers. It periodically measures the link load of the switch (such as switch logs) and sends the link load information to the network controller.

 ISBN: 978-1-61208-216-5

B. Hotspot Detection

Hotspot detection is used to find out the hotspot on physical host and switch link. As shown on Figure 2, the host controller has a hotspot detector which is responsible for detecting hotspot on physical host. The network controller has a hotspot detector which is responsible for detecting hotspot on the switch link in data centers.

1) Host Hotspot Detection

The physical host load metric contains CPU, memory, network facts. A physical host may be overloaded on one or more facts. So, we use vol_{node} [5] as the quantification of the physical host load. If the physical resources are more overloaded, the vol_{node} will be higher.

$$Vol_{node} = \frac{\omega_1}{1-cpu} * \frac{\omega_2}{1-memory} * \frac{\omega_3}{1-net_{node}} \quad (1)$$

ω_i: the weight of CPU, memory and network load.
cpu: the physical host CPU utilization.
memory: the physical host memory utilization.
net_{node}: the physical host network port utilization.
n: the continues observation times.
k: the threshold of overload times.
λ_{node}: the threshold of vol_{node}.

If there are more than k times $vol_{node} > \lambda_{node}$ in the last n detections, the physical host may be thought to be overloaded [5], a hotspot is detected. Then, it schedules the VM migration manager to do a VM migration to eliminate the hotspot.

2) Network Hotspot Detection

The network resource of a data center is the switches' link bandwidth. Thus, the utilization of the link bandwidth is the load of each switch link traffic load. vol_{net} is used to be the quantification of switch link load. If the utilization of the link is high, the vol_{net} will be high.

$$Vol_{net} = \frac{\alpha_{link}}{1-net_{link}} \quad (2)$$

α_{link} : The weight of switch link, if some of the switch link is much more valuable (such as the bottleneck link of the data center network), the weight net_{link} of this link will be bigger.

net_{link}: The link bandwidth utilization.

Similar to host hotspot detection, a network hotspot is flagged only if vol_{net} exceeds a threshold λ_{net} for a sustained time *k* in the recent *n* time observations.

C. Demand Estimation and VM Migration Cost Analysis

As the VM's current used resource may not reflect the actual demand, MWLAN must estimate the VM's actual resource demand before migration. There are many multi-tier applications models to estimate the multi-tier application resource demand. The queuing model [10] is used as the basic of the VM demand estimation. By using the monitored information of application VMs (Gray-box monitoring [5]) and the model for multi-tier applications, MWLAN can estimate the VM physical resource demand (e.g., CPU demand) and VM's actual bandwidth demand.

VM migration scheduling is responsible for choosing which VM to migrate and which physical host to hold the migration VM. And our ultimate goal is to balance the network traffic load and improve the network resource utilization.

If a VM is moved from one physical host to another host, the flows which related to the migration VM will switch too. So when we schedule VMs, a key challenge is to estimate the migration impact to the traffic loads on links. To solve this problem, the system quantifies the impact of the virtual machine migration on network. It considers the bandwidth consumed and the link load by the flows related to the VM before and after the migration.

The VM_w consumed network resource in data centers can be defined as:

$$Cost = \sum_{\forall VM_p \in \{\forall VM_i | C_E(VM_w, VM_i)!=0\}} \sum_{\forall e \in E(VM_w, VM_p)} C_E(VM_w, VM_p) \quad (3)$$

The variables in the Equation are defined as Table I shown.

TABLE I. THE DEFINITION OF VARIABLES IN THE EQUATION

Variable	Description
$C_E(VM_w,VM_P)$	The transfer data rate between VM_w and VM_p.
$C_N(VM_w)$	The amount of physical resource which is allocated to VM_w on physical host.
$E(VM_w,VM_P)$	The switch link set of transfer data path between VM_w and VM_p.
$E'(VM_w,VM_p)$	The switch link set of transfer data path between VM_w and VM_p if VM_w is migrated to physical host PM_k.
$R_E(e)$	The remaining available bandwidth on link *e*.
$R_N(PM_w)$	Physical host PM_w remaining available resource.
α_e	The weight of switch link *e*.
β	The weight of physical host.
δ	Constant, to make ensure the denominator is bigger than zero.
$\{\forall VM_i \mid C_E(VM_w,VM_i)!=0\}$	The VM set which has network traffic with VM_w.

Since our objective is to balance the link traffic load, the utilization of links should also be taken into account. So if the VM_w is migrated from physical host PM_w, the effect to the network traffic load can be defined as:

$$Revenue(VM_w) = \sum_{\forall VM_p \in \{\forall VM_i | C_E(VM_w, VM_i)!=0\}} \sum_{\forall e \in E\ (VM_w, VM_p)} \frac{\alpha_e}{R_E(e)+\delta} C_E(VM_w, VM_p) \quad (4)$$

Revenue (VM_w) considers both the consumed network resource of VM_w and the utilization of related switch links. If moving VM_w away from physical host PM_w, the traffic load of the switch links used by VM_w will be relieved. So (4) denotes the positive effect to the data center network by moving VM_w away from original physical host.

Similarly, the network cost of placing a VM VM_w on physical host PM_k can be defined as:

$$Cost(VM_w) = \sum_{\forall VM_p \in \{\forall VM_i | C_E(VM_w, VM_i)!=0\}} \sum_{\forall e \in E'(VM_w, VM_p)} \frac{\alpha_e}{R_E(e)+\delta} C'_E(VM_w, VM_p) \quad (5)$$

For each VM_w, if it is moved from the original physical host to a candidate host, we denote the benefit of this schedule by *Benefit(VM_w)*.

$$Benefit(VM_w) = Revenue(VM_w) - Cost(VM_w) \quad (6)$$

And if taking physical host load into consideration, we define the benefit of a VM VM_w migration from physical host PM_w as:

$$Revenue' = \sum_{\forall VM_p \in \{\forall VM_i | C_E(VM_w, VM_i)!=0\}} \sum_{\forall e \in E(VM_w, VM_p)} \frac{\alpha_e}{R_E(e)+\delta} C_E(VM_w, VM_p) + \frac{\beta}{R_N(PM_w)+\delta} C_N(VM_w) \quad (7)$$

Similarly, the cost of a VM VM_w is placed on physical host PM_k can be defined as:

$$Cost' = \sum_{\forall VM_p \in \{\forall VM_i | C_E(VM_w, VM_i)!=0\}} \sum_{\forall e \in E'(VM_w, VM_p)} \frac{\alpha_e}{R_E(e)+\delta} C'_E(VM_w, VM_p) + \frac{\beta}{R_N(PM_k)+\delta} C'_N(VM_w) \quad (8)$$

D. VM Migration Schedule

According to the above migration *cost* and *revenue* equations, the intuitive migration manager policy proceeds as follows: At first, compute the migration revenue of each candidate VM which is located on the overloaded physical host or which traffic flows are forwarded by the overloaded link. After that, referring to the above migration *revenue* (4), we sort the VM migration *revenue* in decreasing order. The policy chooses the candidate VM of the maximum *revenue* as the one to migrate. By considering VMs in *revenue* order, the algorithm attempts to migrate the VM which has the biggest potential to relieve the link load and the bandwidth cost. And then, according to the above migration *cost* (5), the migration manager first computers the candidate VM migration *cost* on each underloaded physical host. And again we sort the VM migration *cost* of the each physical host in increasing order. The policy chooses the minimize *cost* physical host as the destination physical host for the candidate VM, which also aims to minimize the network cost. The main steps of this strategy are listed in Algorithm 1. The complexity of the Algorithm 1 is *O (max (m,n))*, where m denotes the candidate VM number, and n denotes the number of physical host which can hold the migration VM.

Algorithm 1 virtual machine migration (**MWLAN1**)
Require: the overload physical machine(PM) PM_w

```
1:  For each VM_i ∈ PM_w
2:      R(VM_i) = Revenue(VM_i,PM_w)
3:  end for
4:  //Note: Revenue computed by (4)
5:  sort VM_i ∈PM_w in decreasing order Revenue (VM_i))
6:  for each VM_i ∈PM_w in decreasing order Revenue (VM_i))
7:      VM_migration= VM_i, PM_dest = NULL
8:      Min_cost = inf
9:      for each PM_j in a data center
10:         if (!check_pm_constrain (PM_j, VM_migration))
11:             continue // pm can't hold the vm
12:         end if
13:         //Note: Cost computed by (5)
14:         cost(PM_j)= Cost (VM_migration, PM_j)
15:         if (cost(PM_j) <Min_cos)
16:             PM_dest = PM_j
17:             Min_cost = cost(PM_j)
18:         end if
19:     end for
20:     if(PM_dest == NULL)
21:         continue
22:     else
23:         break
24:     end if
25: end for
26: if(PM_dest == NULL)
27:     no physical machine can hold a migration VM
28:     return
29: else
30:     return { VM_migration , PM_dest }
31: end if
```

While the Algorithm 1 takes into account both migration *revenue* and *cost*, it can't make sure that the migration gets to maximum benefit which is defined on (6).

The Algorithm 2 merges the process of VM candidate and destination physical host choosing. As the total migration should both consider the *revenue* and *cost*, the Algorithm 2 chooses the migration VM and destination physical host which maximizes *benefit* (6) among all candidate VMs on overload PM and all candidate physical host in data centers. The complexity of algorithm 2 is *O (mn)*, where m denotes the candidate VM number, and n denotes the number of physical host which can hold the migration VM.

 ISBN: 978-1-61208-216-5

Algorithm 2 virtual machine migration (**MWLAN2**)
Require: the overload physical machine(PM) PM_w

```
1:  Max_benefit =0 ,Total_benefit = 0
2:  Current_benefit = 0, Min_cost = inf
3:  Current_pm = NULL,VM_migration= NULL
4:  PM_dest = NULL
5:  for each VM_i ∈PM_w
6:      //Note: Revenue computed by (4)
7:      R(VM_i) = Revenue (VM_i,PM_w)
8:      Current_pm = NULL
9:      for each PM_j in a data center
10:         If(!check_pm_constrain(PM_j,VM_i))
11:             continue  // pm can't hold the vm
12:         end if
13:         //Note: Cost computed by (5)
14:         Current_cost = Cost (VM_i , PM_j)
15:         If(Current_cost < Min_cost)
16:             Min_cost = Current_cost
17:             Current_pm = PM_j
18:         end if
19:     end for
20:     if(Current_pm = =NULL)
21:         continue
22:     else if( (R(VM_i)- Min_cost) > Max_benefit )
23:         VM_migration= VM_i
24:         PM_dest = PM_j
25:     end if
26: end for
27: if(PM_dest == NULL)
28:     no physical machine can hold a migration VM
29:     return;
30: else
31:     return { VM_migration , PM_dest }
32: end if
```

If we also consider the physical host load balancing as well as link load balancing, we can use (7)(8) as the VM migration revenue and cost to replace (4)(5).

IV. EVALUATION

This section describes the evaluation of MWLAN and other migration schemes on simulated data center. The goal of these tests is to compare data center link load on different migration schemes and analyze the impact of different migration schemes on application's TCP transfer rate. The simulated data center is implemented by using ns-3 simulator. Ns-3 [1][11] is a discrete-event network simulator and used in lots of research work [12][13]. What's more, ns-3 is free software, licensed under the GNU GPLv2 license.

A. Evaluation Setup

We use ns-3 to generate a three-layer tree structure of the data center network. Each leaf node is a physical host. Each non-leaf node is a 10-port switch which is connected with sub-node. This data center network has 1 0-level switch which link bandwidth is 5MB/S, 10 1-level switches which link bandwidth is 1MB/S, 100 physical hosts, so the 0-level switch will be the bottleneck of data center network. In order to compare the efficiency of migration schemes on different data centers' link load, we increase the number of VMs placed in the data center from 0 to 360. All the VMs are 2 tier multi-tier application components. Each VM only transfers data with the other VM which belongs to the same multi-tier application. The default transfer protocol is TCP. The detail simulation parameters are noted in Table II.

TABLE II. PARAMETER FOR SIMULATIONS

Variable	Distribution	Mean	Var
Capacity (PM)	Normal	1.8	0.1
Demand (VM)	Normal	0.2	0.1,0.2
Rate (VM)	Normal	0.2,0.4	0.1
Arrival of VMs	Poisson	20(s)	20(s)
The initial placement of arrived VMs	Random, Same switch, Different switch		
Num of VMs	0-360		
VM migration schedule interval	200(s)		
Data center network topology	Tree		

Because the efficiency of migration schemes may vary with different traffic patterns caused by the initial placement of VMs before migration, we run the compared test on three different VM initial placement patterns. In the first pattern, the initial placement of arrived VM is random (Random Pattern). And the VMs which have traffic are placed in the same 1-level switch in the second pattern (Same Pattern). And in the last pattern, the VMs which have traffic are placed in different 1-level switches (Different Pattern).

The benchmark tests are running as follows: we assume the VM requests arrive in a Poisson process with an average rate of 1 VMs per 20 seconds units. Each VM sends data to another VM using TCP protocol. VM migration occurs periodically every 200 seconds. This configuration can make sure the percentage of the migration VM is about 10%. Considering the VM migration cost, 10% is an appropriate migration proportion. The experiment lasts until the number of VMs larger than 360 in the data center. We implement MWLAN 1 and MWLAN 2 which are presented in Section III. The test compares MWLAN 1 and MWLAN 2 with previous migration scheme Sandpiper [5] which moves the VM from the most overloaded physical host to the least overloaded physical host. All VM migration schemes make sure total load of VMs on a physical host that doesn't larger than its capacity. In this paper, the experiments employ several network performance metrics: the average TCP transfer rate and the total link packet loss in the data center.

 ISBN: 978-1-61208-216-5

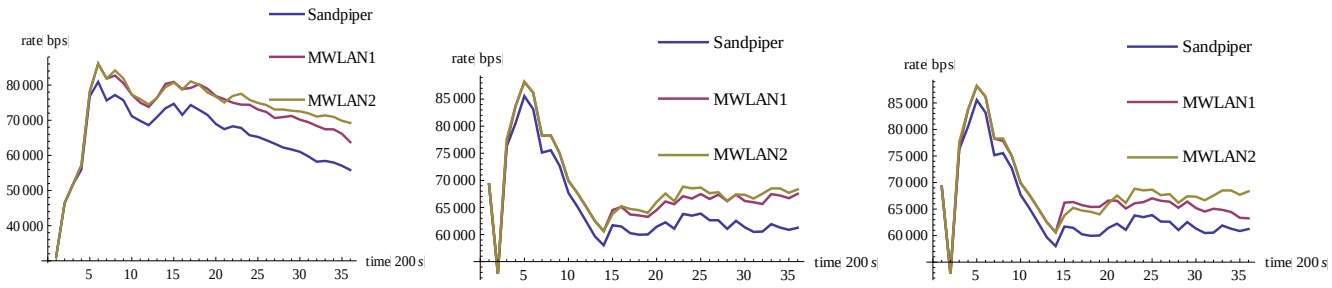


Figure 3(a) Random Pattern Figure 3(b) Same Pattern Figure 3(c) Different Pattern

Figure 3. Average VM transfer rate in three initial VM placement patterns

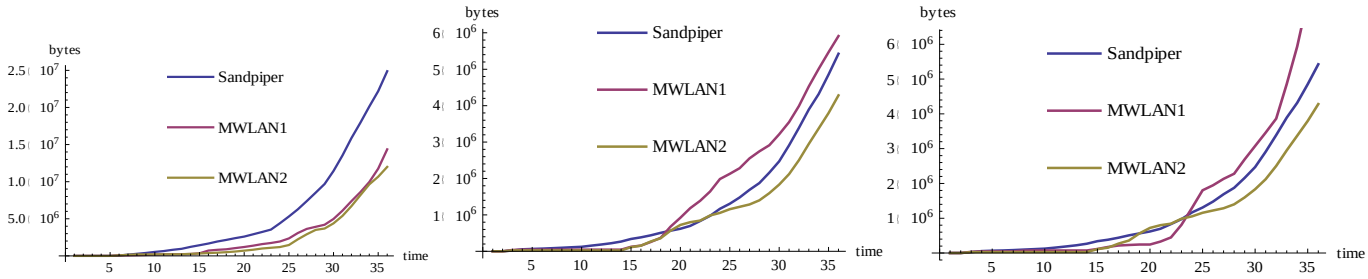


Figure 4(a) Random Pattern Figure 4(b) Same Pattern Figure 4(c) Different Pattern

Figure 4. Total packet loss in three initial VM placement patterns

B. *Evaluation Results and Analysis*

Figure 3 shows the application average TCP transfer rate of Sandpiper and MWLAN as time changes on different VM placement patterns. The result indicates that the application performance of MWLAN2 is better than the other scheme as the VM load increasing. The average improvement of application rate is up to 24% compared to Sandpiper. As shown in Figure 3, the average TCP rate is nearly the same in the beginning. And as the VM load increases, the TCP average rate differs to each other for three VM migration approaches. The traffic rate declines more obviously when using Sandpiper compared to our approaches. The reason is that there is no network congestion when the traffic load is not heavy in data cent network. So the VMs can achieve the demand TCP rate. But as the VM load is increasing, the link traffic load is becoming heavier. When network congestion occurs, the TCP rate decreases, as what we see in Figure 4. And MWLAN1 and MWLAN2 consider the link load cost. So they will move the traffic flows from the loaded links to the underloaded links by using VM migration or move the VMs with heavy traffic near to each other for saving link bandwidth cost. Thus MWLAN1 and MWLAN2 not only eliminate the local traffic congestion but also improve the utilization of network resources. These two factors make MWLAN 1 and MWLAN 2 have better network improvement compared to Sandpiper. The Figure 3 also indicates that MWLAN2 has better network performance improvement than MWLAN1. The reason is that MWLAN1 consider the migration revenue and cost separately, it can't make sure the VM migration achieves the maximum benefit, while MWLAN2 always chooses the VM and destination physical host which can get the maximum benefit.

The experiments also make a comparison on the link load when using different VM migration schemes in the data center. We use the packet loss amount as a comparison object. The link packet loss amount can reflect the load of link traffic in the data center. It can be seen from the Figure 4, MWLAN2 outperforms MWLAN1 and Sandpiper, decreasing total link packet loss up to 50% compared to Sandpiper. It reflects that MWLAN2 policy can be more efficient to avoid network congestion in contract to the other policies. Because MWLAN1 can't get maximum migration benefit, it is not as good as MWLAN2. The MWLAN1 only takes load revenue into account when it chooses candidate VM to migrate, the VM which has high migration revenue may also have high migration cost. As a result, MWLAN1 may burden link load and cause network congestion. On the other hand, MWLAN2 always choose the candidate VM and physical host which can get maximum migration benefit. Thus, MWLAN2 can find the best approach to change link load dynamic to avoid and relieve network congestion.

V. RELATED WORK

As VM migration is transparent to the application [14][16][17], virtual machines consolidation and migrations based on data centers have attracted significant attention in recent years [5][6], many works focus on improving the

efficiency of physical host or power management in data centers.

The work in [6] employs dynamic VM consolidation to reduce the number of working physical host in data centers. Wood et al. implement a system that automates the task of monitoring and detecting hotspots, eliminating physical host hotspots by using VM migration [5]. However, this proposed migration algorithm only considers physical host and virtual machine node-resource load (such as CPU, memory)), which ignores the impact of inter-communication between virtual machines and the data center network factors (link bandwidths, the distance between physical machine). Verma et al. [7] discuss the issue between the physical resource utilization and the data center power consumption. It analyzes the application workload and makes consolidation for power saving. Again, these above approaches do not take the effects on underlying network traffic and link load into account when doing VM consolidation and migration in data centers.

Recent proposals [8][9] for VM placement and migration consider network traffic among virtual machines, But they only consider the total transfer data between virtual machine and the distance between physical machines when doing migration. This network factor is too coarse-grained to effectively use the data center network resources, while our migration system considers not only the traffic among VMs but also link traffic load of data centers.

In contrast to our work, none of the approaches mentioned above addresses the problem of network link load dynamic adaption in order to avoid network congestion or overload.

VI. Conclusion and Future Work

Previous VM consolidation and migration strategy mainly focus on the physical host resource utilization or physical host load balancing, but ignore the factors of data center network and the traffic between VMs. As the network performance is becoming more and more important in data centers, how to use VM migration to improve the data center network traffic load is a meaningful research topic. This paper proposes a novel migration strategy MWLAN. It quantifies the benefit of VM migration and the cost of VM placement to the network link load in data centers. This migration strategy takes the data center network link load and link bandwidth cost factor into account to solve the migration problem efficiently. What's more, the experimental results demonstrate that MWLAN has better network performance compared to the other schemes. MWLAN not only reduces data center network congestion but also improves the application transfer data rate. For future work, we look forward to implementing and evaluating our scheme on different kinds of data center network. Moreover, we plan to coordinate the VM placement and VM migration policy for network load balancing in data centers.

References

[1] T. R. Henderson, M. Lacage, and G. F. Riley. Network Simulations with the ns-3 Simulator. Demo paper at ACM SIGCOMM'08, August 2008.

[2] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, et al. PortLand: a scalable fault-tolerant layer 2 data center network fabric, Proceedings of the ACM SIGCOMM 2009 conference on Data communication, August 16-21, 2009, Barcelona, Spain

[3] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, et al. BCube: a high performance, server-centric network architecture for modular data centers, Proceedings of the ACM SIGCOMM 2009 conference on Data communication, August 16-21, 2009, Barcelona, Spain

[4] D.G. Andersen. Theoretical approaches to node assignment. http://www.cs.cmu.edu/~dga/papers/andersen-assign.ps 2002.

[5] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif. Black-box and gray-box strategies for virtual machine migration. In Proc. of the 4th Symposium on Networked Systems Design and Implementation (NSDI). USENIX, 2007.

[6] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy: a consolidation manager for clusters, Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, March 11-13, 2009, Washington, DC, USA.

[7] Verma, P. Ahuja, and A. Neogi. pMapper: Power and migration cost aware application placement in virtualized systems. Technical report, IBM, 2008.

[8] X. Meng, Y. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. IEEE INFOCOM, 2010.

[9] V. Shrivastava, P. Zerfos, K. won Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee. Application-aware virtual machine migration in data centers. In Proc. of IEEE INFOCOM Mini-conference, Apr. 2011.

[10] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. An Analytical Model for Multi-tier Internet Services and Its Applications. In Proc. of the ACM SIGMETRICS, Banff, Canada, June 2005.

[11] http://www.nsnam.org/

[12] Jahanzeb Farooq and Thierry Turletti. An IEEE 802.16 WiMAX module for the NS-3 simulator, Proceedings of the 2nd International Conference on Simulation Tools and Techniques, March 02-06, 2009, Rome, Italy.

[13] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. ns-3 project goals. Proceeding from the 2006 workshop on ns-2: the IP network simulator, October 10-10, 2006.

[14] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proc. NSDI '05*, May 2005.

[15] Y. Zhang, A. Su and G. Jiang. Evaluating the Impact of Datacenter Network Architectures on Application Performance in Virtualized Environments, Proceedings of 18th IEEE International Workshop on Quality of Service (IWQoS), 2010.

[16] M. Nelson, B. Lim, and G. Hutchins. Fast Transparent Migration for Virtual Machines. In Proc. USENIX 2005.

[17] Sherif Akoush, Ripduman Sohan, Andrew Rice, An-drew W. Moore, and Andy Hopper. Predicting the performance of virtual machine migration. Modeling, Analysis, and Simulation of Computer Systems, International Symposium on, 0:37–46, 2010.

[18] N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping, IEEE INFOCOM, 2009.

 ISBN: 978-1-61208-216-5

Performance Influence of Live Migration on Multi-Tier Workloads in Virtualization Environments

Xiaohong Jiang, Fengxi Yan, Kejiang Ye
College of Computer Science, Zhejiang University
Zheda Road 38#, Hangzhou 310027, China
{jiangxh, yanfengxi, yekejiang}@zju.edu.cn

***Abstract*—Live migration is a widely used technology for load balancing, fault tolerance, and power saving in cloud data centers. Previous research includes significant research work in the performance improvement of live migration. However, little work has been done to investigate the influence of live migration on virtual machine workloads that users care about most. We notice that these workloads can be classified into two categories: single-tier workloads and multi-tier workloads which is a typical type for internet applications. We conduct a series of deliberate experiments to investigate the influence of live migration on multi-tier workloads in a cloud environment and also on traditional physical machines for comparison. Our experimental results show that multi-tier workloads on virtual machines can work as well as those on traditional physical machines. However, in an unstable environment, if virtual machines migrate constantly, live migration will cause a profound performance decrease on multi-tier workloads. Also, it is best to avoid migrating virtual machines that are hosting memory intensive workloads in a virtualization environment due to bad downtime performance. Further, we perform experiments trying to find the turning point of the performance of a virtual machine, which might provide support evidence for future research on live migration policy.**

Keywords-virtualization; live migration; XEN; Muti-tier workload.

I. INTRODUCTION

In a cloud datacenter, virtualization technology is widely preferred because of its impressive advantages in cost savings, easy resource management, high resource utilization, high availability, and good scalability. Live migration [1] is a core technique to implement load balancing, fault tolerance, and power savings in a virtualization environment. Most virtualization systems such as XEN [2], KVM [3], and VMware [4] support the live migration of virtual machines. Many researchers have been attracted to the investigation of live migration performance [5, 6]. However, the influence of live migration on virtual machine workloads, especially complex interactive workloads, hasn't been considered. What type of workloads will be affected most by live migration? Which virtual machine (VM) should be migrated so that the influence on workloads will be as small as possible? These questions are important for data center management as they directly affect the Quality of Service (QoS).

There are many kinds of workloads in a cloud datacenter. We classify them into two categories: single-tier workloads and multi-tier workloads. A single-tier workload runs on one single host and does not exchange data with workloads on other hosts. Most traditional single machine applications belong to this category. Multi-tier workloads are composed of a set of workloads running on different hosts and are constantly interacting with each other through the network. Multi-tier workloads have the following obvious features:

- Group work.
  Multi-tier workloads are not alone. They are a group of workloads running on different hosts connected to each other and work together in a multiple tier architecture.
- Interactive.
  Multi-tier workloads interact with each other. For example, Tier A transfers data to Tier B, Tier B analyzes the data and transfers the result back to Tier A.
- Sensitive of Single-Node Failure.
  If one of the nodes in a multi-tier workload fails, the remaining workloads should be stopped and wait for the failed node to resume again.

A dynamic website is an example of a typical multi-tier workload, which is composed of a frontend web server and a backend database server. Dynamic websites are the main form of websites on the internet as they provide better communication between web users and the website. A dynamic website can capture web users' input, search or retrieve data from the database, return the data to the web server and display the data in the web browser in an easily understandable way. When a web user sends a HTTP request containing some parameters to the web server, the web server will execute scripts based on the parameters, make queries to the database, and format the result into HTML files, which will be transferred back to the client. Figure 1 shows the architecture of a dynamic website. In fact, most internet applications fall into the category of multi-tier workloads.

Some research work has been done to measure the influence of live migration on single-tier workloads [5, 6] instead of multi-tier workloads. However, in a real Cloud data center, multi-tier workload is one of the most commonly used

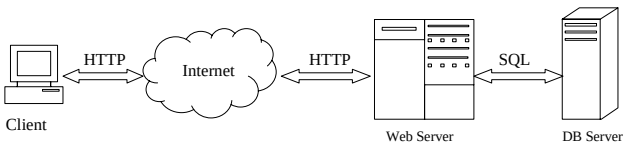


Figure 1. Architecture of Dynamic Website

 ISBN: 978-1-61208-216-5

application types other than the single-tier one. Research about the influence of live migration on multi-tier workloads has profound guiding significance to the choice of live migration policy. This paper is trying to determine how multi-tier workloads will behave when the host virtual machine is migrated to another physical machine in a virtualization environment. We conduct a series of experiments in a XEN virtualization environment with RUBiS [7, 8], a dynamic website benchmark. The experimental results show some useful information on virtual machine management that can be used as support evidence for a live migration policy.

The main contributions of this paper are summarized as follows:

- We study the performance effects of live migration on multi-tier workloads, including both web server and database server. And we analyze the migration overhead from downtime, total migration time, and the workload performance.
- We investigate the migration point issue or turning point issue at which the virtual machines should be migrated to other physical machines to avoid the performance degradation. It is the best migration point to reduce both the migration overhead and workload overhead.
- The experimental results show some meaningful suggestions to real cloud computing environments, and it is also meaningful to the further migration strategy development. For example, the memory-intensive workloads should avoid migrating first.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to the benchmark "RUBiS" and the XEN hypervisor. Section 3 describes our experimental design. Section 4 describes the experimental results and our analysis. Finally, we summarize our conclusion in Section 5.

II. BACKGROUND

A. RUBiS Benchmark

RUBiS (Rice University Bidding System) [7, 8] is a free and open source benchmark of dynamic websites developed by Rice University. Its prototype is eBay and it is designed to evaluate application design patterns and the scalability of application servers using MySQL as its database.

The benchmark implements the main functions of an auction website: browsing, registering, selling, and bidding. There are three kinds of user sessions: visitor, buyer, and seller. A visitor does not need to register and is only permitted to browse. Buyers and sellers need to register. A buyer can bid on items and check the list of his or her current bids as well as any competitive bidding and comments left by other users. A seller can register an item for sale, sometimes with a reserve price, and view the list of his or her selling list.

RUBiS can be accessed by users from a browser, but for convenience, RUBiS implements a client emulator tool, which can emulate common users of this auction site. In fact, the client emulator can create many user sessions randomly. During a user session, RUBiS mass generates URLs for this user based on a pre-defined workload, and sends HTTP requests based on these URLs. With this mechanism, the client emulator behaves just like a real user: browses the homepage, browses items from categories and regions, registers to become a user, bids or buys an item, registers an item for sale and views his or her bidding and selling history.

There are three versions of RUBiS: a PHP version, a Java servlets version and an EJB version, which are for different usage. In our experiment, we use the PHP version for three reasons. First, this version is easy to install, maintain and use, so we can concentrate on our experiments. Second, PHP is one of the most popular languages used in web applications nowadays. Third, the PHP server Apache and its database server MySQL have the typical architecture of a dynamic website.

B. XEN and Live Migration Technique

In our experiment, we use XEN [2, 9] as our virtual machine monitor (or called Hypervisor). XEN is an open source project developed and maintained by Xenoserver research project at Cambridge University. It is a layer of software running directly on computer hardware replacing the operating system, thereby allowing the computer hardware to run multiple guest operating systems concurrently. Because of its support for x86, x86-64, Itanium, Power PC, and ARM processors, XEN hypervisor is able to run on a wide variety of computing devices and supports various operating systems (for example, Linux, NetBSD, FreeBSD, Solaris, Windows, and other common operating systems) as guest operating systems running on the hypervisor.

A virtual machine running on XEN hypervisor can be migrated to another physical machine, using the cold migration or the live migration technique. Cold migration needs the migrated virtual machine to be shut down completely in order to transfer the virtual machine disk image to the destination physical machine. The migrated virtual machine is restarted only after the disk image transfer is completed. This kind of migration takes too much time and the migrated virtual machine is not available during the period of migration. If the migrated virtual machine is undertaking some interactive workloads with other virtual machines, all workloads must stop running because of the disconnection to the migrated virtual machine.

Live migration handles the migration of a virtual machine in three aspects: network, disk, and memory. The network migration is just an IP address redirection, and the disk migration can be solved with storage net-share technology (for example, NFS[10], SAN[11], and NAS[12]). The main problem is the memory migration. It is done in 4 phases:

Phase 1: Pre-migration and Reservation

Assume a virtual machine is about to migrate from host A to host B. The XEN hypervisor first makes sure that host B has enough resources to hold the virtual machine, and

 ISBN: 978-1-61208-216-5

then reserves an empty VM container on host B for the virtual machine to be migrated

Phase 2: Iterative Pre-Copy

Dirty memory is transferred to host B in time intervals called iterations. During the first iteration, all memory pages will be transferred from host A to host B. Subsequent iterations only transfer the dirtied memory generated during the previous iteration.

Phase 3: Stop-and-Copy

This phase comes when the XEN hypervisor thinks that the remaining dirty memory can be transferred in a very short time interval or that there have been too many iterations of pre-copy in the previous phase. The virtual machine on host A will be shut down and its remaining dirty memory and CPU state will be copied to the virtual machine on host B. Now there are 2 copies of the virtual machines, one on A and the other on B.

Phase 4: Commitment and Activation

Host B informs Host A that it is ready to start the new virtual machine, and some post-migration code runs to attach the disk driver and IP address to the new virtual machine. The new virtual machine starts and the migration is complete.

Live migration can proceed seamlessly when the migrated virtual machine is running, and the virtual machine only stops for a very short time to restart. This period of time is called downtime which is so short that users and workloads on the virtual machine would not even be aware of it.

III. EXPERIMENT DESIGN

Our experiments are conducted on 4 physical machines (PM1, PM2, Client Emulator Server and Storage Server), which are connected by an Ethernet with the bandwidth of 1000Mbps. A network with such a high bandwidth will not become a bottleneck in the network transmission in our experiments. Every physical machine has enough memory so that memory will not become a bottleneck, either. Each machine has 8 CPU cores, with a clock rate of 2.27GHz.

We use Apache as our Web Server, MySQL as the Database (DB) Server, Debian Linux as the Operating System (OS). On PM1 and PM2, we deploy XEN hypervisor 4.0 to manage the virtual machines in the experiments. The virtual machines are also installed with Debian Linux as their OSes.

The Storage Server is a SAN server. All the virtual machine images are stored in this SAN server which is shared by PM1 and PM2 with an iSCSI access interface.

Our experiments are conducted in 4 phases:

In phase 1, we measure the performance of RUBiS on physical machines. We turn off 6 CPU cores on PM1 and PM2 respectively so that we have 2 CPU cores left on each of them. The RUBiS Web Server is deployed on PM1 and the RUBiS DB Server on PM2. The Client Emulator Server runs the RUBiS Client Emulator program to emulate common users who would visit the RUBiS website through HTTP connections. The architecture is shown in Figure 2. We call this phase ***PHYSICAL MODE***.

In phase 2, we measure the performance of RUBiS on virtual machines. In order to compare with the previous set of experiments, we allocate two VCPUs for each virtual machine to get an equivalent configuration compared with the ***PHYSICAL MODE***. A virtual machine with 2 VCPUs will be created on PM1, running the RUBiS Web Server, we call this virtual machine VM1; another virtual machine with 2 VCPUs will be created on PM2, running the RUBiS DB Server, we call this virtual machine VM2. Then enough memory is allocated for VM1 and VM2, so that memory will not become a bottleneck. The Client Emulator Server still runs the RUBiS Client Emulator program. The architecture is shown in Figure 3. We call this phase ***VIRTUAL MODE.***

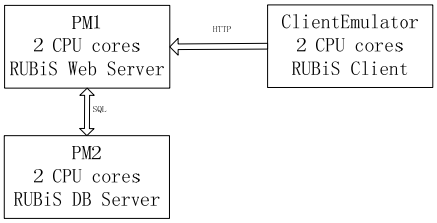


Figure 2. Experiment overlay of ***PHYSICAL MODE***

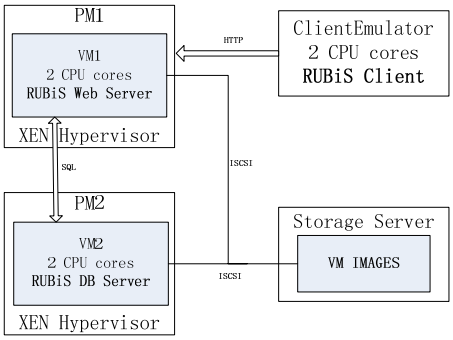


Figure 3. Experiment overlay of ***VIRTUAL MODE***

In phase 3, we measure the performance of RUBiS on virtual machines under live migrations. The experiment overlay is just the same as that in phase 2 (see Figure 3). But in the middle of every experiment, we conduct a live migration for VM2 which holds the RUBiS DB Server. VM2 migrates from PM2 to PM1. After the migration, we collect the migration time and downtime. This phase is named ***MIGRATION-DB MODE***.

Phase 4 is similar to phase 3. The difference is that VM1 is migrated instead of VM2. VM1 is migrated from PM1 to PM2. We collect the migration time and downtime of every single migration. This phase is named ***MIGRATION-WEB MODE.***

We collect the RUBiS throughput (requests per second) and the CPU usage rate in each experiment in all of the above 4 phases. Then we compare the collected data and

 ISBN: 978-1-61208-216-5

make a further analysis of the performance of multi-tier workloads in virtualization environment.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. *Comparison of **PHYSCIAL MODE** and **VIRTUAL MODE***

The throughputs of RUBiS increasing with the number of clients in ***PHYSICAL MODE*** and ***VIRTUAL MODE*** are shown in Figure 4 and Figure 5 separately.

We can figure out from Figure 4 that the throughput of RUBiS benchmark in the ***PHYSICAL MODE*** goes up quickly before the number of clients reach 1400, slows down after reaching the number of 1400, and finally stabilizes after the number 1600.

Compared with Figure 4, it's easy to determine in Figure 5 that throughput in ***VIRTUAL MODE*** goes up almost the same way as that in ***PHYSICAL MODE***. It implies that virtual machines with equivalent configuration of hardware resources can achieve equivalent performance compared with traditional OS instances running on physical machines. In this circumstance, virtualization technology does not cause any obvious performance decrease for the multi-tier workloads.

Throughput reaches its maximum value when the number of clients is 1600, as the concurrent connections with RUBiS Web Server reaches the Apache Server's configured "MaxClients" attribute. In both ***PHYSICAL MODE*** and ***VIRTUAL MODE***, CPUs with 2 cores is powerful enough to run the RUBiS system, so CPU will not be a bottleneck.

The two curves shown in the graph series in Figure 6 and Figure 7 are CPU usage ratios of "Web Server" in the upper side and of "DB Server" in the lower side. From Figure 6 and Figure 7, we can determine that both the RUBiS Web Server and the DB Server use a small fraction of CPU even if throughput reaches its peak value.

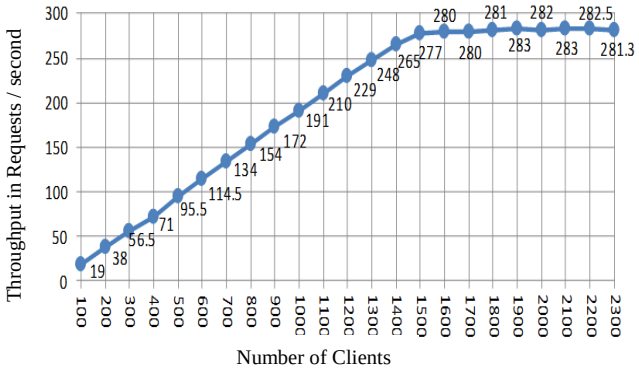


Figure 4. Throughput of RUBiS in ***PHYSICAL MODE***

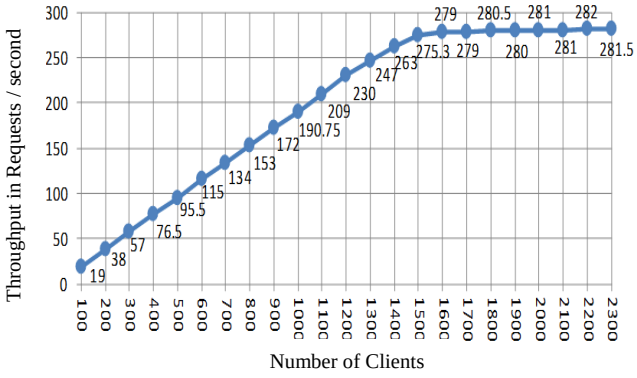


Figure 5. Throughput of RUBiS in ***VIRTUAL MODE***

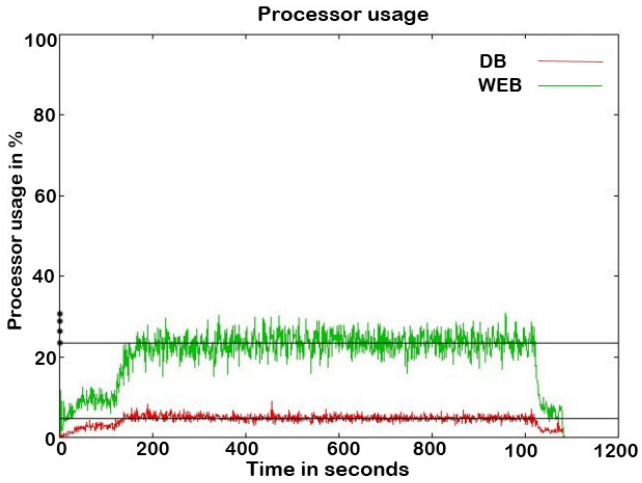


Figure 6(a). CPU usage when clients = 1300

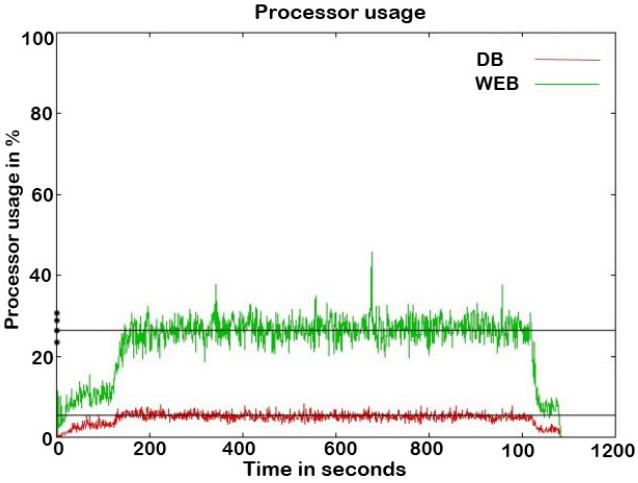


Figure 6(b). CPU usage when clients = 1400

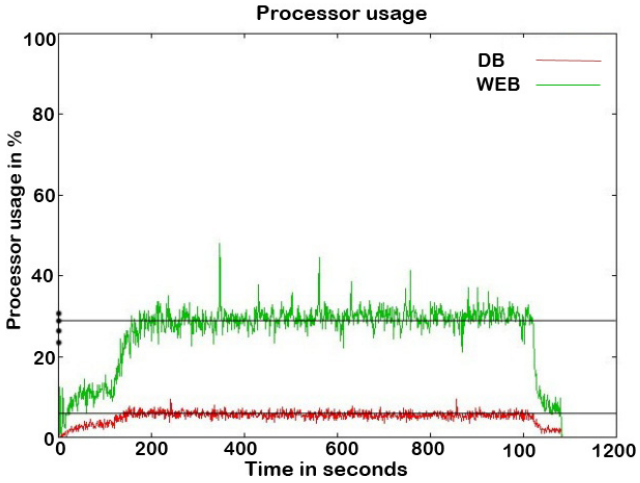


Figure 6(c). CPU usage when clients = 1500

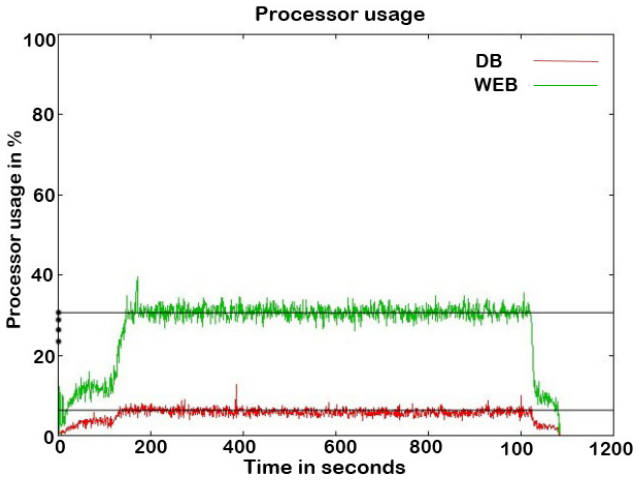


Figure 6(d). CPU usage when clients = 1600

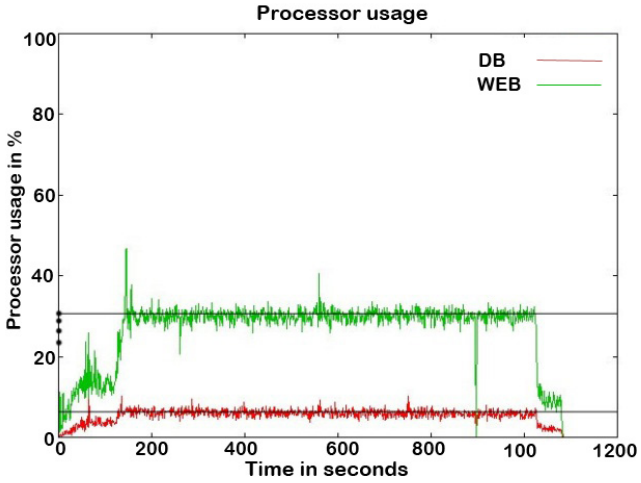


Figure 6(e). CPU usage when clients = 1700

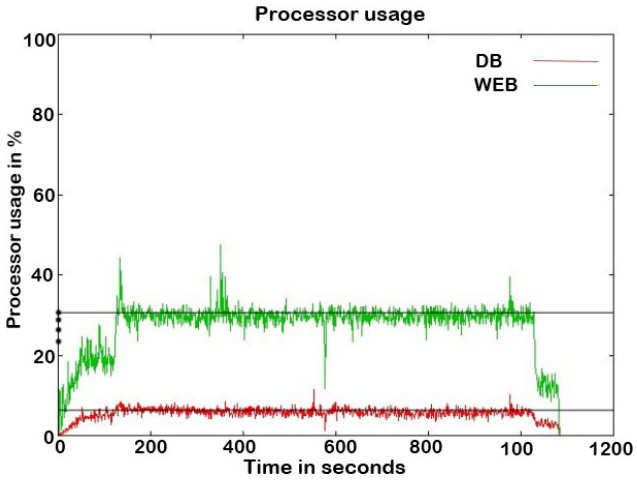


Figure 6(f). CPU usage when clients = 2300

Figure 6. CPU usage ration as a function of time in seconds in ***PHYSICAL MODE***

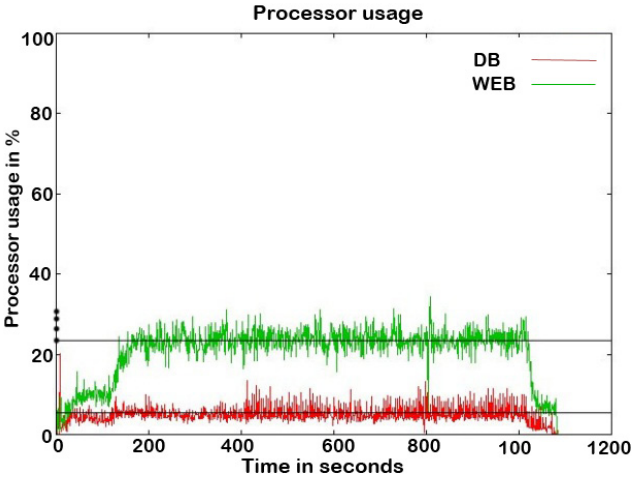


Figure 7(a). CPU usage when clients = 1300

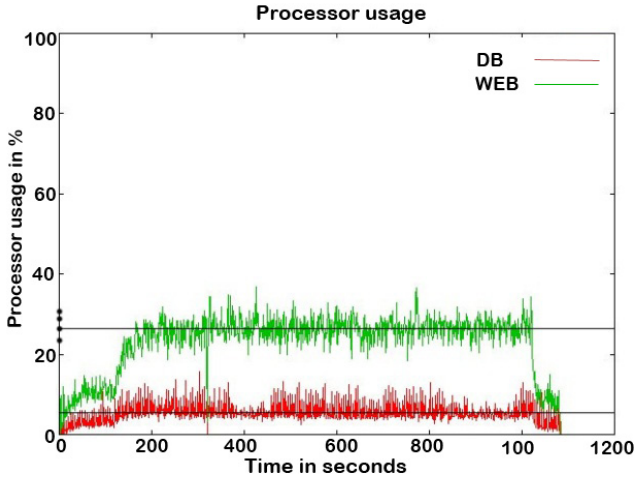


Figure 7(b). CPU usage when clients = 1400

 ISBN: 978-1-61208-216-5

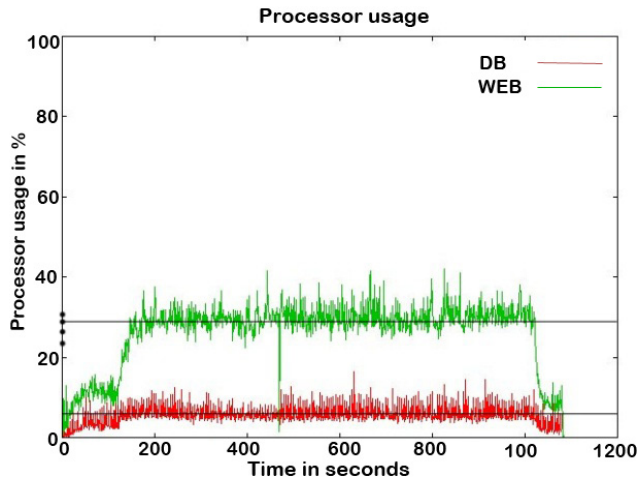


Figure 7(c). CPU usage when clients = 1500

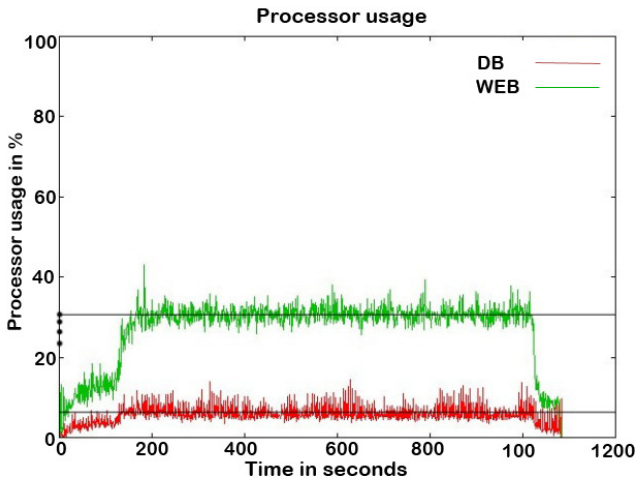


Figure 7(d). CPU usage when clients = 1600

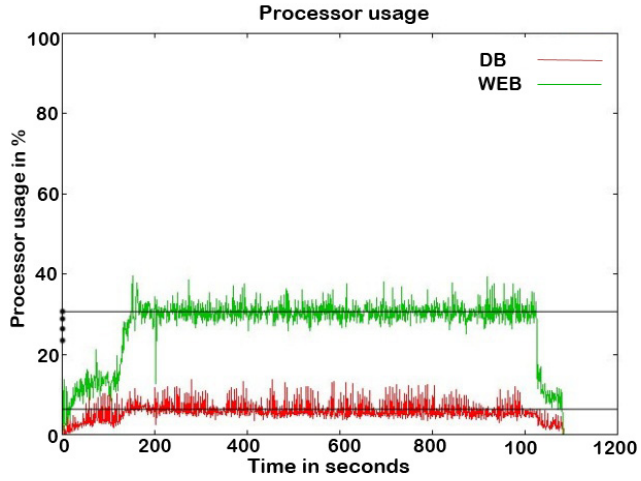


Figure 7(e). CPU usage when clients = 1700

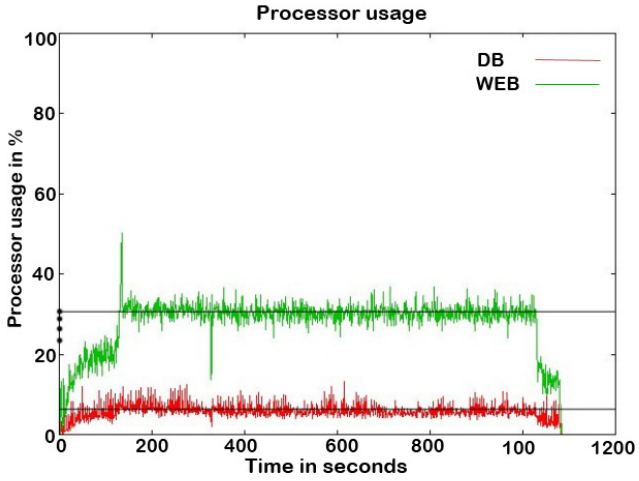


Figure 7(f). CPU usage when clients = 2300

Figure 7. CPU usage ratio as a function of time in seconds in ***VIRTUAL MODE***

Images in Figure 6 and Figure 7 are too small and it is hard to tell the number of CPU usage ratio, so we drew some sub lines to indicate the average CPU usage ratio comparatively. Figure 6 shows that the CPU usage ratio increases with the number of clients and reaches the maximum value when the client number reaches 1600 or more. The peak value of CPU usage is 31% for the RUBiS Web Server and 6% for the DB Server.

Figure 7 shows that in ***VIRTUAL MODE***, the CPU usage ratio is very similar to that in ***PHYSICAL MODE***. It also increases to the peak value when the client number reaches 1600 at about 31% for the RUBiS Web Server and 6% for the RUBiS DB Server.

From the above figures, the virtual machines show demonstration of wonderful performance: with equivalent configuration of hardware, they perform as well as the physical machines and do not consume more CPU resource than physical machines, even when running multi-tier workloads. We conclude that when multi-tier workloads are deployed on virtual machines, they can work as well as that on physical machines, without any extra CPU consumption.

However, we notice that the above conclusion for the ***VIRTUAL MODE*** can be drawn only in a somewhat stable circumstance. What will the result be if workloads run in an unstable circumstance? For example, how will the performance of multi-tier workloads be influenced when the host virtual machine is migrated? Experiments comparing ***MIGRATION-DB MODE*** *and* ***MIGRATION-WEB MODE*** try to answer this question and provide evidence support for a migration policy.

*B. Comparison of **MIGRATION-DB MODE** and **MIGRATION-WEB MODE***

In this subsection, we analyze the migration performance of virtual machines running RUBiS Web

 ISBN: 978-1-61208-216-5

Server and DB Server respectively. These two migration experiments show very different effects in migration time, downtime and throughput when migrating Web Server and DB server.

We first analyze the migration time difference. Figure 8 shows the memory usage in ***VIRTUAL MODE***. We obtain the memory usage when the client number is 1600 which is the turning point of the throughput. From Figure 8, we can see that the RUBiS Web Server (upper curve in Figure 8) consumes more memory than the DB Server (lower curve in Figure 8). So it is clear that the Web Server is more memory intensive than the DB Server. As mentioned in Section II, memory migration is the main task in virtual machine migration compared with network migration and disk migration, and is the decisive factor for migration time, downtime and throughput. So it's easy to jump to the conclusion that migration of virtual machine hosting memory intensive workloads will lead to longer migration time and downtime due to the migration of more dirty memory. However, experiment results turn out to be different from the above imprudent conclusion.

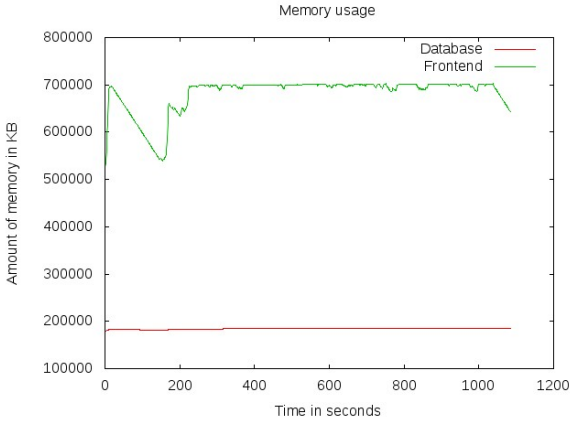


Figure 8. The memory usage graph when client=1600 in ***VIRTUAL MODE***

Figure 9 shows the migration time both in ***MIGRATION-DB MODE*** and ***MIGRATION-WEB MODE***.

The migration time in ***MIGRATION-WEB MODE*** is longer than that in ***MIGRATION-DB MODE*** when the client number is less than 1100, which is in accord with our prior intuitive conclusion. However, the migration time in ***MIGRATION-WEB MODE*** becomes the shorter one when the client number increases larger than 1100. Given the fact that the RUBiS Web Server is more memory intensive and more memory can be dirtied during the migration in the ***Iterative Pre-Copy*** phase in migration, the RUBiS Web Server will spend more time to copy the dirty memory than the DB Server. So it's very easy to understand why ***MIGRATION-WEB MODE*** has longer migration time. But why does it becomes the shorter one when the client number grows larger than 1100? In order to answer this question, we need to analyze the phases occurring in the live migration.

There are in total 4 phases in the live migration: 1) Pre-migration and reservation; 2) Iterative pre-copy; 3) Stop-and-Copy; 4) Commitment and Activation. Especially two conditions in the 2nd phase can trigger the 3rd Stop-and-Copy phase. One is the number of small dirty pages falling below the threshold; usually the dirty pages will become less when the dirty pages migrate by round. The second condition is the restriction of the number of iterations, in which when the number of iterations reaches a threshold, the virtual machine has to stop and copy all the remaining dirty memory. This happens when the dirty memory cannot be diminished as the iterative migration is performed.

In our experiment, as shown in Figure 9, there are very few clients accessing the Web Server at the beginning, so the memory used is very little. But when the client number increases, the dirty memory also increases and finally becomes a very large overhead. Because the RUBiS Web Server is memory intensive, the RUBiS Web Server has much more memory dirtied during the migration than the DB Server. The first condition of Stop-and-Copy that achieves a small dirty memory working set cannot be satisfied because dirty memory is generated faster than the memory has been migrated. So the virtual machine hosting Web Server ends the Iterative Pre-Copy phase in advance and makes the total migration time relatively shorter than the DB virtual machine when the client number increases more than 1100.

It can also be validated in Figure 10, from which we find the downtime in ***MIGRATION-WEB MODE*** is much longer than that in ***MIGRATION-DB MODE*** because the dirty memory working set of the Web server is larger than the DB server. It consumes more time to migrate the last of the dirty memory and incurs longer downtime. On the other hand, the DB server iterates more round cycles and the dirty memory can be relatively less, so the downtime can be short.

Based on the above evidence, we can conclude, contrary to our intuition, that the migration time of a VM hosting Web server is shorter than that of a VM hosting DB server when the client number is larger than a specific size. Nevertheless, it's better not to migrate the virtual machine hosting memory intensive workloads as the Web server in our experiment due to longer downtime.

In realistic situations, to achieve different goals of migration, we should use different methodologies accordingly. If we need to keep a stable performance of the workload involved, we should migrate the VM that is not memory intensive, because this would guarantee shorter downtime. Otherwise, if we want to finish the migration as soon as possible and keep a stable performance from the entire Cloud datacenter's sight, we should migrate the memory intensive ones because shorter migration time will occur.

Figure 11 depicts the throughputs in the last three modes. The throughput in ***MIGRATION-DB MODE*** and ***MIGRATION-WEB MODE*** is much smaller than that in the first two modes. During the downtime, the migrating virtual machine is entirely disconnected, and all clients'

 ISBN: 978-1-61208-216-5

accessing RUBiS will fail. What's more, the throughput in ***MIGRATION-WEB MODE*** is much less than that in the ***MIGRATION-DB MODE*** because of its longer downtime in migration. So it's better to avoid migrating virtual machines running memory intensive workloads (the Web server virtual machine in our experiment).

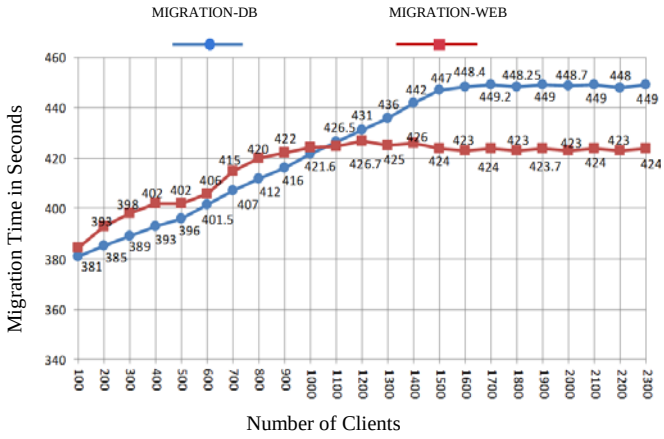


Figure 9. Migration Time as the Client Number Increases

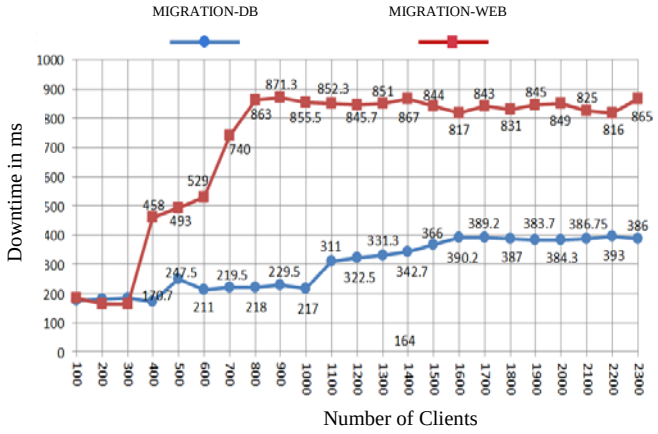


Figure 10. Downtime Time as the Client Number Increases

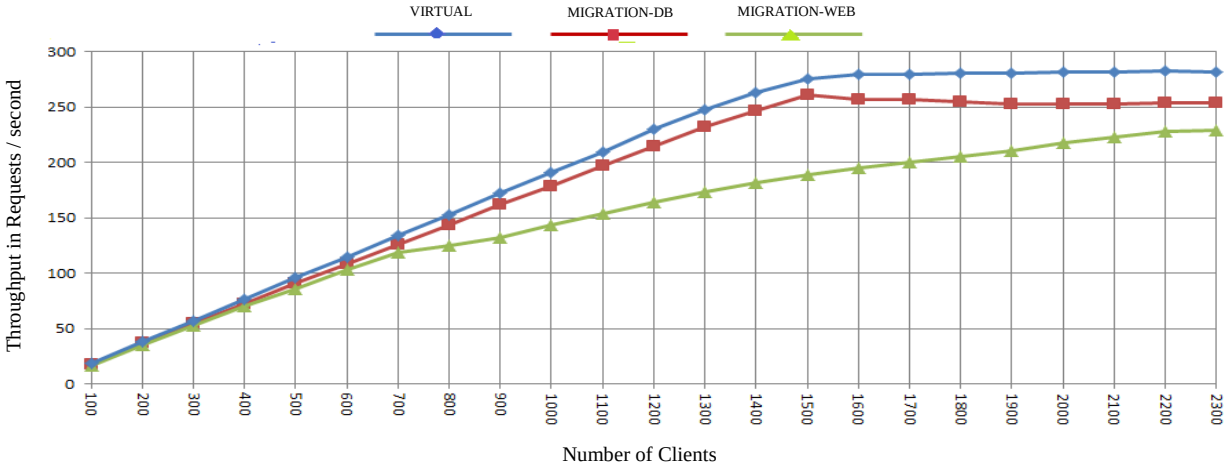


Figure 11. Throughput in ***VIRTUAL MODE, MIGRATION-DB MODE*** and ***MIGRATION-WEB MODE***

Based on the above analysis of ***MIGRATION-DB MODE*** and ***MIGRATION-WEB MODE***, we can determine that live migration indeed has an adverse effect on the running of multi-tier workloads. It's better to avoid the live migration of virtual machines as much as possible. When live migration cannot be avoided on demand of load balancing, fault tolerance, or power saving, it's better to not migrate the virtual machine hosting memory intensive workloads.

C. Analysis of Live Migration Point

From the above experiments, we can conclude that the performance of multi-tier workloads running on a virtual machine can be affected by the live migration process with different degrees. However, live migration of virtual machines indeed happens frequently in cloud computing environments to achieve the goals of dynamic resource management. For example, when the physical machine is nearly exhausted of CPU resource, it is better to migrate some of the virtual machines on this physical machine to other physical machines, because the lack of CPU resource also decreases the workloads' performance and might even lead to application failure. After the virtual machine is migrated to the physical machine rich with CPU resource, the performance of the physical machine will return to the normal level, and eventually avoid server or application failure. In this subsection, we will investigate performance issues in such scenarios.

In order to make the most of the CPU and start migration only when necessary, we should find a performance turning point(we name it **T**) of the application when the physical machine is about to be fully

loaded. The applications running on virtual machines work well before point **T**, and will turn bad after **T.** The turning point **T** might be a proper point to migrate the virtual machine on the nearly fully loaded physical machine. We conduct the following experiment to determine the **T** point specifically in our system.

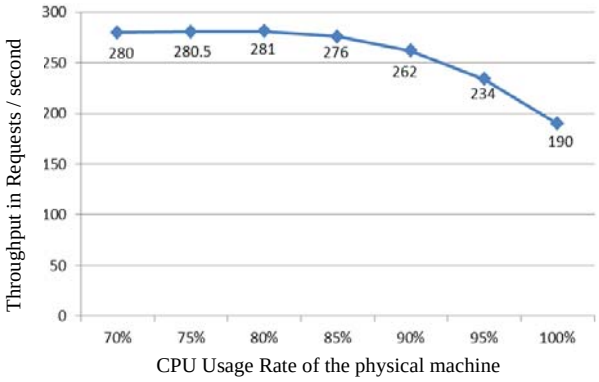


Figure 12. Throughput under different CPU usage rate when client=1600

First, we run VM1 on PM1 and VM2 on PM2 with both PM1 and PM2 having abundant CPU resource. The RUBiS Client emulates 1600 client sessions. And we can get a throughput of 281 requests per second.

Then, we start running a CPU intensive program on PM1, which can use up as much of the CPU resource as we set. We conduct a series of experiments under different CPU usage rates, and get the throughput accordingly. The results are shown in Figure 12. The x axis indicates the CPU usage rate of the physical machine hosting the virtual machines running the benchmark RUBiS. The y axis indicates the throughput of the multi-tier workloads.

From Figure 12, we can determine that the throughput of the virtual machine hosting RUBiS starts to decrease at 80%. So 80% CPU usage rate might be the **T** point. So Xen's migration policy might be improved to start live migration only when necessary at the turning point of 80% CPU usage rate. It is better for a Cloud administrator to find the turning point specifically in his Cloud data center to make the live migration policy more efficient. This experiment provides a basis for further research work in finding a proper T point of live migration in the cloud data center, which might be more precisely defined as the sub-healthy state of the virtual environment.

V. RELATED WORK

The workload performance issue incurred by virtualization technology in cloud computing environments has been widely investigated. Researchers have studied the performance overheads from both a single virtual machine perspective [2, 13, and 14] and a multiple virtual machine perspective [15, 16]. However, there is relatively little work referring to multi-tier workloads with interactive characterization.

A Multi-tier application is a typical kind of internet workload and has specific characterization. Urgaonkar et al. presented an analytical model for this application by using network of queues [17]. Bi et al. employed a hybrid queuing model to understand the performance of virtualized multi-tier applications and determine the number of virtual machines at each tier in a virtualized application [18]. However, they didn't consider the factors of live migration.

Live migration of virtual machines is used widely in today's cloud data center to achieve the goal of load balancing, fault tolerance, and saving energy. Xen and VMware primarily use the pre-copy technology to implement the live migration of virtual machines [1, 19]. After that many efforts have been made to improve the performance of live migration. Hines et al. presented a post–copy technique to implement the live migration of virtual machines which is different with the pre-copy technique [20]. Jin et al. proposed an adaptive memory compression method to reduce the overhead of memory transfers and improve the migration performance [21]. Liu et al. used the technique of full system trace and replay to optimize the migration efficiency [22]. Luo et al. solved the problem of whole-system migration in which both the memory and disk states were migrated [23]. Ye et al. investigated the issue of multiple virtual machine migration and proposed a method based on resource reservation to optimize the overall migration efficiency [24]. In order to evaluate the performance of different live migration techniques, Huang et al. designed a benchmark for live migration [25]. However, all the above work has not considered the characteristics of the multi-tier virtual machine workloads.

VI. CONCLUSION

We have made a deliberate analysis about multi-tier workloads and found that very little work has been done to measure the influence of virtualization technology especially live migration on multi-tier workloads running on VMs. Because multi-tier workloads comprise most of the workloads in a real Cloud data center, determining the influence detail is significant to the choice of live migration policy.

To achieve this goal, we have conducted a comprehensive performance analysis of multi-tier workloads in a virtualization environment, especially the performance characterization under the live migration. Based on the experimental analysis, we find that virtualization technology, especially the live migration technique, has some hidden influences on multi-tier workloads. The experimental results tell us that virtual machines can achieve nearly equivalent performance with the same system configuration compared with traditional OS instances running on physical machines. That is to say that multi-tier workloads can work well in a virtual machine environment. However, the live migration of virtual machines can cause some performance decrease

 ISBN: 978-1-61208-216-5

due to migration overhead and the downtime during which the migrating virtual machine should be shutdown. This decrease is especially obvious to those virtual machines running memory intensive multi-tier workloads. It is necessary to balance the migration benefits and overheads. In order to answer the question when the virtual machines should be migrated, we designed an experiment to find the proper migration point under different hardware resource configuration (for example, CPU utilizations in the experiment). Experimental results show that at the turning point of 80% CPU usage rate, the migration can benefit the workloads' performance. A Cloud administrator should determine the turning point of their Cloud data center specifically and adjust the live migration policy.

Future work will include developing adaptive migration framework for cloud computing and designing intelligent live migration strategies to improve the overall workloads' performance.

VII. ACKNOWLEDGMENT

This work is funded by the National High Technology Research 863 Major Program of China (No.2011AA01A207) and MOE-Intel Information Technology Foundation (No.MOE-INTEL-11-06). We would like to thank the reviewers for their insightful comments. We would also like to thank Mr. Jeff Wood for his careful revision of our paper.

REFERENCES

[1] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines", in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2 (NSDI'05), Vol. 2. USENIX Association, Berkeley, CA, USA, pp. 273-286, 2005.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," ACM SIGOPS Operating Systems Review, vol. 37, no. 5, pp. 164–177, 2003.

[3] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "KVM: the Linux virtual machine monitor", in Proceedings of the Linux Symposium, vol. 1, pp. 225-230, 2005.

[4] C. Waldspurger, "Memory resource management in VMware ESX server", ACM SIGOPS Operating Systems Review, vol. 36 (SI), pp. 194, 2002.

[5] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation", in Proceedings of the international conference on Cloud Computing (CloudCom), pp. 254-265, 2009.

[6] S. Akoush, R. Sohan, A. Rice, A.W. Moore, and A. Hopper, "Predicting the performance of virtual machine migration", 2010 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 37-46, 2010.

[7] RUBiS home page. http://rubis.ow2.org/, [retrieved: March, 2012]

[8] C. Amza, A. Chanda, A.L. Cox, S. Elnikety, R. Gil, K. Rajamani, W. Zwaenepoel, E. Cecchet, and J. Marguerite, "Specification and implementation of dynamic Web site benchmarks," 2002 IEEE International Workshop on Workload Characterization, pp. 3-13, 2002.

[9] XEN community. http://www.XEN.org/, [retrieved: March, 2012]

[10] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network Filesystem", USENIX, 1985.

[11] J. Tate, F. Lucchese, and R. Moore, "Introduction to Storage Area Networks- Exhaustive Introduction into SAN", IBM redbook. www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf

[12] G. A. Gibson and R. V. Meter, "Network Attached Storage", Communications of the ACM, vol. 43, no. 11, pp. 37-45, 2000.

[13] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. Matthews, "Xen and the art of repeated research," USENIX annual Technical Conference, pp. 135-144, 2004.

[14] A. Menon, J. Santos, Y. Turner, G. Janakiraman, and W. Zwaenepoel, "Diagnosing performance overheads in the xen virtual machine environment," in VEE: Proceedings of the 1st ACM Conference on Virtual Execution Environments, pp. 13-23, 2005.

[15] M.F. Mergen, V. Uhlig, O. Krieger, and J. Xenidis, "Virtualization for high-performance computing", ACM SIGOPS Operating Systems Review, vol. 40(2), pp. 8-11, 2006.

[16] K. Ye, X. Jiang, S. Chen, D. Huang, and B. Wang, "Analyzing and modeling the performance in xen-based virtual cluster environment", 12th IEEE International Conference on High Performance Computing and Communications (HPCC), pp. 273-280, 2010.

[17] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications", ACM SIGMETRICS Performance Evaluation Review, vol. 33(1), pp. 291-302, 2005.

[18] J. Bi, Z. Zhu, R. Tian, and Q. Wang, "Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center", in Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 370-377, 2010.

[19] M. Nelson, B. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in Proceedings of the annual conference on USENIX Annual Technical Conference, p. 25, 2005.

[20] M. Hines and K. Gopalan, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning," in Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp. 51-60, 2009.

[21] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive memory compression," in Proceedings of the IEEE International Conference on Cluster Computing, pp. 1-10, 2009.

[22] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," in Proceedings of the 18th ACM international symposium on High performance distributed computing, pp. 101-110, 2009.

[23] Y. Luo, B. Zhang, X. Wang, Z. Wang, Y. Sun, and H. Chen, "Live and incremental whole-system migration of virtual machines using block-bitmap," in Proceedings of the IEEE International Conference on Cluster Computing, pp. 99-106, 2008.

[24] K. Ye, X. Jiang, D. Huang, J. Chen, and B. Wang, "Live migration of multiple virtual machines with resource reservation in cloud computing environments", in Proceedings of the 2011 IEEE International Conference on Cloud Computing (CLOUD), pp. 267-274, 2011.

[25] D. Huang, D. Ye, Q. He, J. Chen, and K. Ye, "Virt-LM: a benchmark for live migration of virtual machine", in Proceeding of the second joint WOSP/SIPEW international conference on Performance engineering, pp. 307-316, 2011.

 ISBN: 978-1-61208-216-5

About the flexible Migration of Workflow Tasks to Clouds

Combining on- and off-premise Executions of Applications

Michael Gerhards, Volker Sander
Faculty of Medical Engineering & Technomathematics
FH Aachen, University of Applied Sciences
Jülich, Germany
{M.Gerhards|V.Sander}@fh-aachen.de

Adam Belloum
Institute of Informatics
University of Amsterdam
Amsterdam, Netherlands
A.S.Z.Belloum@uva.nl

***Abstract* - An increasing number of applications target their executions on specific hardware like general purpose Graphics Processing Units. Some Cloud Computing providers offer this specific hardware so that organizations can rent such resources. However, outsourcing the whole application to the Cloud causes avoidable costs if only some parts of the application benefit from the specific expensive hardware. A partial execution of applications in the Cloud is a tradeoff between costs and efficiency. This paper addresses the demand for a consistent framework that allows for a mixture of on- and off-premise calculations by migrating only specific parts to a Cloud. It uses the concept of workflows to present how individual workflow tasks can be migrated to the Cloud whereas the remaining tasks are executed on-premise.**

Keywords - Cloud Computing; Cloud Service Broker; Grid Computing; Workflow; Workflow Orchestration

I. INTRODUCTION

An increasing number of applications target their execution on specific hardware. Field Programmable Gate Arrays (FPGAs) and free programmable general purpose Graphics Processing Units (GPUs) are existing approaches to use cost-effective high performance computational power in specific applications. Image processing and image guided interventions are well-known examples for use cases in which both platforms compete with each other [1].

However, not all parts of those applications are equally suitable for the usage of this hardware. Of course, related applications follow an approach in which only specific parts of a program were optimized for the specialized computation resources that are therefore only used during specific time slots. As a consequence, there is the risk that these resources are otherwise idling so that an own purchase might not be cost-effective. Therefore, for many scenarios it appears to be opportune to outsource computation intensive parts off-premise with easy-scale and dynamic provisioning whereas the other parts are executed on-premise on local available general-purpose computational resources.

Grid and Cloud Computing are potential infrastructures that support this scenario since both provide special resources for suitable application parts, whereas the remaining application parts can be executed on general resources. This concept can be extended to software in deploying software with expensive licenses on only some computers on Grids and Clouds. These computers were used to execute the application parts that require the deployed software, whereas the remaining parts might be executed elsewhere to make the computers available for other applications that rely on the related software.

But, not every organization has access to a Grid or does want to use it because it requires joining a related virtual organization [2]. Cloud Computing offers a promising alternative infrastructure for using scalable on demand resources with specific hardware. Providers such as Amazon allow users to allocate virtualized general purpose GPU-resources. Of course, those providers allow for porting the full application including the parts that rely on specific hardware to their premises. However, as described above, this might not be the most cost-effective solution. This paper addresses the demand for a consistent framework that allows for a mixture of on- and off-premise calculations. The proposed solution is based on workflows. The motivation scenario can therefore be viewed as an example for a concept that applies to a much broader application domain.

Modeling a complex application as workflow supports its division into simpler individual parts that are executed as interacting tasks by a workflow management system. These tasks are reusable for other workflows in the same way that software libraries are reusable in applications. Workflows are frequently used in e-Science for “climate modeling, earthquake modeling, weather forecast, astrophysics and high energy physics” [3] but also in the e-Business domain for Business Process Management (BPM).

The remaining of this paper is organized as follows: Section II introduces workflows with related definitions. It also provides an example in which parts of the workflow rely on specific hardware resources. Further on, it briefly describes the differences between Grids and Clouds according to workflow integration. Since the support of workflows in Cloud infrastructures is surprisingly rather limited, Section III introduces a novel approach to handle workflows in the Cloud computing domain. It provides technical descriptions, discusses possible alternatives, and provides more complex extensions. Section IV describes the related work and delimits the suggested architecture from an existing approach. Finally, the last section concludes the results and describes future work.

 ISBN: 978-1-61208-216-5

II. WORKFLOWS IN GRIDS AND CLOUDS

Complex processes are often modeled as workflows described using a specific workflow modeling language. A workflow is composed of several tasks, which could depend on each other. Therefore, a workflow can be illustrated as directed graph composed of tasks as nodes and task dependencies as directed edges. Directed edges connect the predecessor task with its successor task. A task can only start its execution if its predecessor has finished its own execution.

The example workflow illustrated in Figure 1 was designed for the Shape Retrieval Contest 2010 (SHREC'10) aiming to classify a set of proteins based on their 3D structure [4]. It consists of five tasks, illustrated as rectangles. The arrows illustrate the dependencies of the tasks. In this workflow data are only fed in at the beginning of the two task pipelines and are then handed over from task to task.

The tasks *APURVA* and *Sort* are computation intensive and well parallelizable. Therefore, they are candidates for a migration to off-premise computation resources like the Cloud, potentially by using specific High Performance Computing (HPC) hardware such as general purpose GPUs or FPGAs. In the following such tasks are called *Cloud Tasks*. The pre-processing of the *PP* tasks and the item duplication of the *X 1000* task should stay for execution on on-premise computation resources to reduce data movements and avoid costs. In the following such tasks are called *Local Tasks.*

A so-modeled workflow is called a workflow template that describes the behavior of a process; thus, it can be referred to as a general workflow definition. It is comparable with a program's source code. Such templates are deployed, instantiated, and executed on a workflow management system [5] that takes care of the individual tasks' progress and dependencies. Workflow instances follow the behavior of their assigned workflow template for a particular incident. It is comparable to a program's execution.

A particular challenge arises when workflows are mapped to resources at different organizations, each providing a heterogeneous system with non-uniform interfaces to access these resources. Thus, the submission of workflow jobs is more difficult due to the fact that different administrative domains have different accounting mechanisms.

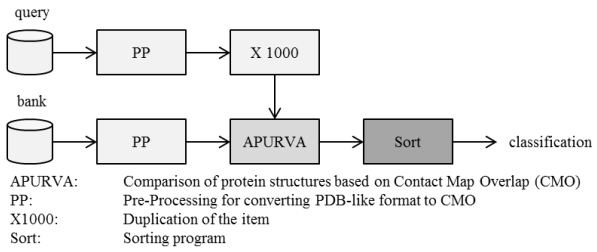


Figure 1. Example workflow with the two computation intensive tasks *APURVA* and *Sort.*

Grid middleware platforms support the execution of workflows in virtual organizations, where the distributed resources are owned by multiple organizations. Abstract Grid workflows are described independently of specific resources because new resources can be established or existing ones can be omitted or blocked. The binding of workflow tasks to Grid resources is done at runtime.

The Grid concept of considering only physical resources is gone in the Cloud vision of infinite resources that just have to be activated. The allocation of resources is different than in Grids. Any number of Cloud resources can be instanced on demand. "With the emerging of the latest Cloud Computing paradigm, the trend for distributed workflow systems is shifting to Cloud Computing based workflow systems [6]."

Cloud resources are not automatically part of a virtual organization and therefore not integrated into a trusted domain. The resource allocation mechanism differs from provider to provider. To execute a workflow task in a Cloud, the software must be deployed on a Cloud instance and be accessible from the workflow management system via a remote procedure call (RPC) mechanism like a web service. Cloud Computing per se does not impose any specific limitations with respect to the usage API while Grid Computing needs a middleware using a particular API that complies to the rules of the virtual organization.

The National Institute of Standards and Technology (NIST) [7] distinguishes the three Cloud service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). SaaS providers often focus on standard applications like text processing or customer relation management and will not cover the whole variety of possible tasks. The current existing PaaS offerings only provide standard hardware for general purpose. IaaS is currently the only service model which enables executing programs on specific hardware in the Cloud. Therefore, the rest of the paper will only consider IaaS resources. This should not limit the generality since suitable SaaS or PaaS offerings can be used instead.

NIST [7] also distinguishes four different deployment models: Private Cloud, Community Cloud, Public Cloud, and Hybrid Cloud. Since the example scenario assumes that the specific hardware is not used frequently, a Private Cloud providing such hardware is not feasible. However, the Private Cloud can be used to provide general on-premise resources for the execution of *Local Tasks*. Sharing the specific hardware of a Community Cloud is only possible if such a community exists but this cannot be assumed. Since the paper focuses on outsourcing calculations, renting Public Cloud special resources fulfills all hardware requirements for off-premise calculations. A Hybrid Cloud as combination of a Private Cloud for general on-premise resources with a Public Cloud for special off-premise resources is the required environment for the combination of on- and off-premise calculations.

The rest of paper will only focus on the integration of *Cloud Tasks* that should be executed on IaaS in a Public Cloud.

 ISBN: 978-1-61208-216-5

III. WORKFLOWS WITH CLOUD TASKS

A simple approach to migrate a workflow task to the Cloud is the usage of a service-oriented approach by deploying the task software as web service on the Cloud instance and binding the workflow task to this web service. Web services provide standardized uniform interfaces which supports interoperability of heterogeneous systems. The data to be processed are typically passed as parameter from the workflow task to the assigned web service. An alternative approach for passing larger sets of data is that the web service loads the requested data itself using a onetime access ticket granted by the workflow management system. Independent of the data transfer mechanism, the data should not be stored permanently on the computing Cloud instance because the data are not automatically saved persistently on Cloud images so that a reboot of the resource will result in data loss. On-premise databases or storage Clouds provide permanent, secure, and persistent data storage for the results of the calculation.

Since IaaS resources are frequently provided following a pay-per-time billing structure, any Cloud instance should be terminated after each use to avoid unnecessary costs while the resource is idling. The consequence is that the Cloud instance has to be started again before a re-use is possible. The task execution idles during the bootup of the Cloud instance. Preconfigured machine images contain only the required software to speed up the instantiation. Each abstract Cloud Task could use its own machine image or a basic machine image including all necessary basic systems could be loaded and setup with the task software dynamically on bootup. The required task software is identified using the workflow template. The installation of the software can be done automatically using Secure Shell (SSH).

For a just in time start and termination of the Cloud instance, an automatic mechanism must be available. Otherwise the workflow task idles till the Cloud instance service is available or the Cloud instance service is still available after the workflow task's execution. The Cloud instance start and termination can be included into the workflow template by adding the administrative tasks *Create* and *Destroy* which start and terminate the Cloud instances using a Cloud unification layer or a Cloud agnostic Application Programming Interface (API) like the Open Cloud Computing Interface (OCCI) [8]. The *Cloud Task* is bounded fix to the Cloud instance web service that is only available in the time span between the *Create* and *Destroy* tasks. The usage of automatic template modifications has been already validated in [9].

The concept of the workflow template extension has the benefit of being interoperable with other workflow management systems without individual source code modifications. This makes it even usable for proprietary systems. The same template extension application can be used by different workflow management systems if the same modeling language is supported. Standard workflow modeling languages like XPDL [10] and WS-BPEL [11] benefit most of this approach.

The main disadvantage is that the modeling of workflows becomes more complex because the execution semantic is integrated. Workflows must consider administrative tasks instead of focusing on worker tasks.

Therefore, it is much more comfortable to the user when the administrative tasks are integrated automatically into the template during the workflow instantiation. Because the deployment environment cannot decide where a task should be executed, the usage of task annotations in the template specifies where the task has to be executed. This is similar to MAUI [12] where developers annotate which methods of an application can be offloaded for remote execution.

Figure 2 shows the extended example workflow of Figure 1. The two *Cloud Tasks APURVA* and *Sort* now have administrative predecessor and successor tasks. The so modified workflow is executed instead of the original one. The end user will not notice the difference.

Many users instantiate workflows but not each of them should be able to start arbitrary Cloud resources. Otherwise it would not be possible to map caused costs to individual Cloud usages and an abuse of resources would be possible. Therefore, an authentication service is required on workflow side. This service maps the authentication mechanism of the organization to the authentication mechanism of the Cloud Service Provider. The user privileges can be assigned considering many strategies, e.g., a user could have access only a limited time to a Cloud or she/he could have access only to specific Clouds or for specific workflows. SAML [13] assertions can be used for this. The granularity of user privileges is not in focus of this paper. A standard based security system like WS-Trust [14], Simple Authentication and Security Layer (SASL) [RFC 4422], oAuth [RFC 5849], or OpenID can be integrated into the workflow management system.

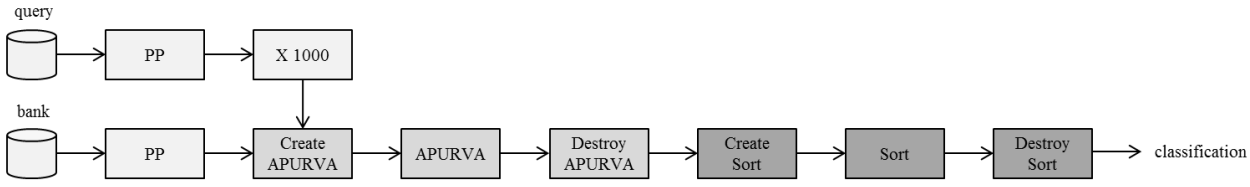


Figure 2. Example workflow extended with administrative Create and Destroy tasks for the two computation intensive tasks *APURVA* and *Sort*.

 ISBN: 978-1-61208-216-5

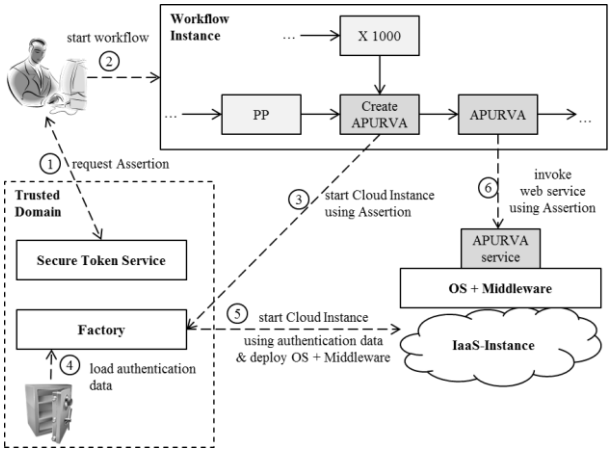


Figure 3. Relationship between workflow instance, Cloud instance, and authentification center.

The process of executing a workflow with *Cloud Tasks* is summarized in the following with reference to Figure 3 where the numbers in circles indicate the order. First the user requests an *assertion token* (1) with only limited use at the *secure token service* by providing her/his own identification together with the identification of all *Cloud Tasks* she/he wants to use. The *secure token service* evaluates the request and decides if the *assertion* can be granted. If the result is positive, the user instantiates the workflow (2). The *Create Task* uses this *assertion* at the factory (3) to proof its eligibility. The *factory* then loads the Cloud account *authentication data* from a secure storage (4) and starts the *Cloud instance* (5) with the deployed *web service*. The *assertion* is now invalidated. The *APURVA Cloud Task* invokes the *web service* (6) that is running on the *Cloud instance*. The *web service* processes the data on the high performance Cloud hardware. After the web service returns its results, the *Destroy Task* shuts down the Cloud instance.

A. Reuse of Web Services

In scenarios like parameter studies, the same workflow task is executed frequently. Other examples of reusing the same task are loops, multiple workflow instances, and different workflows instances using the same *Cloud Task*. The simple approach introduced above starts a new Cloud instance for each Cloud Task instance and terminates the Cloud instance after the web service's execution. The Cloud instance starting overhead slows down the workflow's execution but can be reduced for future invocations by keeping alive the Cloud instance for reusability. A single Cloud web service is then used multiple times by different *Cloud Task* instances of the same abstract *Cloud Task* like APURVA in Figure 4.

The implementation is described in the following: The *Destroy Task* only notifies the *Factory* that the web service is no longer needed by the *Cloud Task*. The integrated scheduler keeps alive the Cloud instance if it expects future web service invocations. Otherwise, the scheduler shuts down the Cloud instance as usual. The prediction is possible by evaluating the assertion requests at the *Secure Token Service*.

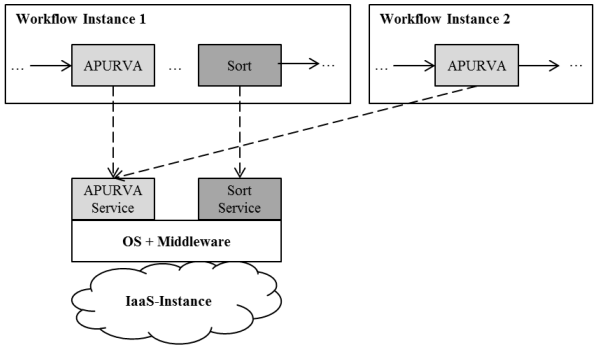


Figure 4. Reuse of Cloud web services and sharing of an IaaS instance.

Listing 1. Shell script to install the web service

```
#/bin/bash
scp -B ~/program.jar user@instance:~/program.jar
ssh user@instance java -jar program.jar parameter
```

B. Multiple Web Services on the same Cloud Instance

To reduce Cloud instance staring overhead and to avoid costs, additional web services can be deployed on the same Cloud instance if they are suitable for the hardware. Figure 4 depicts the *IaaS Instance* that hosts both: *APURVA Service* and *Sort Service*. This optimization is most suitable for workflows with different *Cloud Tasks* that can then be executed in a pipeline on the same Cloud instance. Using this optimization, static machine images cannot be instantiated because additional software must be installed during the uptime of the Cloud instance. The installation can be done using SSH in a shell script like in Listing 1. The first line copies the program via secure copy *scp*. The second line uses *ssh* to start the remote program that will publish its web service as an own endpoint on the Cloud instance by considering the parameter. The password prompt is suppressed using public/private key based authentication.

C. Dynamic Assignment of Tasks to Cloud Resources

The idea of outsourcing only single parts of an application to the Cloud can be extended with a dynamic assignment of the *Cloud Task* to the most suitable Cloud resource at runtime that is illustrated as an example in Figure 5. The selection process is similar to the three-phase cross-cloud federation model described in [15]. In the *discovery* phase, the Cloud Service Broker creates a table in a database which provides information about *Assured Properties* offered by the Cloud Service Providers like in the first three columns of TABLE I. Possible properties are special hardware like general purpose GPUs, best performance, lowest price, performance/price ratio, available volume resources of non-pay-as-you-go contracts, and location of the Cloud for liable reasons or for data nearness as well as data sensitiveness. This table must always be kept up to date. In the workflow template each abstract *Cloud Task* specifies its *Required Properties*. In the example in Figure 5, *APURVA* has the properties *a* and *b* whereas *Sort* has the property *c*. These *Required Properties* are sent to the Cloud Service Broker before the assignment of the *Cloud Task* to its Cloud

 ISBN: 978-1-61208-216-5

resource. Now in the *match-making* phase, the Cloud Service Broker compares the *Cloud Task*'s *Required Properties* with the Cloud Service Providers' *Assured Properties*. The Cloud Service Providers that assure all *Required Properties* of the requesting task are *potential task owners*. The last two columns of TABLE I indicate which resources are the potential owner of which *Cloud Task*. In Figure 5, these potential owners are encircled. In the *authentication* phase, the Cloud Service Broker selects the cheapest *potential owner* as the *current owner* for each *Cloud Task*: Resource 2 for *APURVA* and Resource 3 for *Sort*.

D. Provenance

The importance of validating and reproducing the outcome of computational processes is fundamental to many application domains. It is exposed that there is a need to capture extra information in a process documentation that describes what actually occurred. The automated tracking and storing of provenance information during workflow execution could satisfy this requirement [16]. The amount and the kind of data to be stored are always user and implementation dependent. Provenance traces enable the users to see what has happened during the execution of the workflow. This also enables failure analysis and future optimization. Provenance becomes even more important in distributed environments because workflow tasks are loosely bound to computational resources. Using provenance in the Cloud-workflow domain enables the identification of Task to Cloud assignments so that it is visible where the *Cloud Task* has been executed and where its data have been stored.

TABLE I. ASSURED PROPERTIES OF CLOUD RESOURCE

Cloud Resource	**Assured Properties**	**Price**	**Potential Owner of**	
			APURVA	*Sort*
Resource 1	a	3	×	×
Resource 2	a, b	4	✓	×
Resource 3	a, b, c	6	✓	✓
Resource 4	b, c	7	×	✓

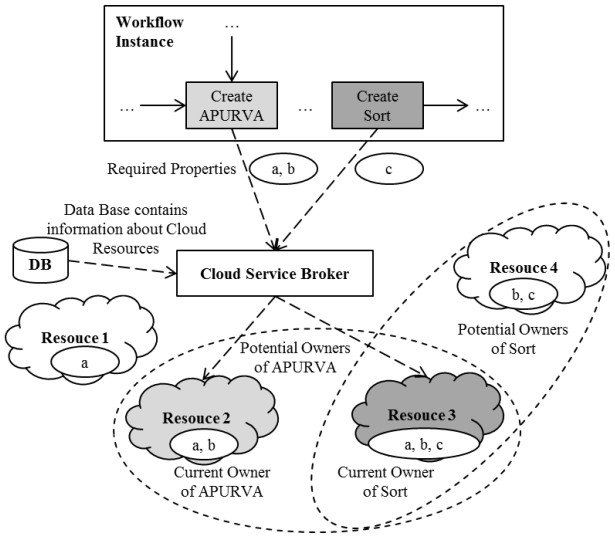


Figure 5. Dynamic assignment of Cloud Tasks to Cloud resources.

Provenance also shows at which time the Cloud instance was running and therefore causing costs. Based on provenance traces, statistics can be created showing which workflows cause which costs, which users cause which costs, which Clouds cause which costs, which users instantiate which workflows, which Clouds execute which *Cloud Task*, etc.. Also the runtime of *Cloud Tasks* can be examined in the provenance trace to optimize future *Cloud Task* to Cloud resource assignments.

A provenance model describes how the gathered provenance data are interpreted and stored in the provenance trace. Several provenance models exist and two of them are described briefly in the following. A detailed comparison is done in [17]. The Open Provenance Model (OPM) [18] is very prominent in the e-Science domain. It provides a comprehensive set of concepts to capture how things came out to be in a given state and is designed to achieve inter-operability between various provenance systems. Another provenance model is the so-called History-tracing XML (HisT) [9]. It was developed within the HiX4AGWS project [19] and provides provenance following an approach that directly maps the workflow graph to a layered structure within an XML document. The *Create* and *Destroy* workflow tasks can be used to identify and transmit the provenance data according to the Cloud instances. HisT directly supports the integration of digital signatures and is therefore optimized for the e-Business and cross-organizational domain where responsibility and liability play an important role.

IV. RELATED WORK

Cloud Computing is the greatest IT hype of the last ten years. Therefore, many publications deal with Cloud Computing. Surprisingly the combination of Cloud Computing with workflows is little addressed. The integration of single off-premise *Cloud Tasks* into on-premise workflows is not supported yet. In comparison to the mobile smartphone domain, approaches like CloneCloud [20] already exists to dynamically partition applications between weak devices and Clouds. Some workflow management systems claim to be ready for the Cloud but they are mostly ported from the Grid domain and only support running in the Cloud as extension to running in the Grid. The flexible selection and interaction with Cloud resources is not implemented in the workflow management systems considering the requirements identified in section III. One approach is presented in the following and then delimited to the approach presented in this paper.

The Generic Workflow Execution Service (GWES) [21] is an open source workflow management system and was developed by Frauenhofer-Gesellschaft for the management and the automation of complex workflows in heterogeneous environments. The service orchestration goes through five abstraction levels: *User Request*, *Abstract Workflow*, *Service Candidates*, *Service Instances*, and *Resources*. The formal described *User Request* represents an abstract operation and is automatically composed into an infrastructure independent non-executable *Abstract Workflow*. This *Abstract Workflow* is mapped at runtime down to available *Resources*. During

 ISBN: 978-1-61208-216-5

this process *Service Candidates* web services are searched and optimally selected to become *Service Instances*. GWES was originally developed basing on Grid technologies like Globus Toolkit as Grid Workflow Execution Service (also GWES) and was then adjusted to the Cloud domain.

The proposed approach of this paper differs from the basic GWES concept. GWES is a specific workflow management system with an own workflow description language. In contrast the interoperable approach of this paper bases on an extension for existing modeling languages of arbitrary workflow management systems by the integration of the Cloud administrative tasks *Create* and *Destroy* which connect the workflow instance with the Cloud Service Broker to select, start, and stop the Cloud instance. By choosing a workflow management system independent approach the usage of the already known system is given for the end-user. The approach is the migration of only individual workflow tasks to the Cloud whereas the remaining tasks stay in the local environment for execution.

V. CONCLUSION AND FUTURE WORK

This paper presented a general concept for the hybrid execution of workflows by allowing the off-premise execution of specific tasks in the Cloud whereat the remaining tasks stay on-premise to avoid unnecessary costs. The proposed architecture has the advantage that it is neither depending to a particular workflow engine nor to a particular workflow description language. It follows the approach of automatically modifying workflow templates to incorporate the steps for assigning the appropriate off-premise resource in a flexible manner. This approach has been already validated in the domain of provenance [9]. The Cloud Service Broker automatically selects the most suitable Cloud resource to guaranty the fulfillment of all task requirements. The end users' interfaces are not changed so that workflows can be used the same way as before.

Next steps of work will be the implementation of the introduced Cloud Service Broker including an analysis of an according selection metric. The occurred costs of a partial off-premise execution will be compared with the costs of a full off-premise execution to calculate a costs reduction ratio. The time overhead for migrating tasks across Cloud and organizational boundaries has to be measured and set it into relation with the avoided costs to figure out if the costs reduction is worth the time overhead. Even data movement strategies have to be implemented.

The security of the whole architecture plays an important role which is minor addressed in this paper. The Secure Token Service and the Factory are together the single point of access. Unauthorized Cloud resource instantiations and unauthorized Cloud web service invocations must be protected against requests without permission to avoid a misuse.

ACKNOWLEDGMENT

This work was carried out in the context of HiX4AGWS [19]. HiX4AGWS is supported of the Federal Ministry of Education and Research in Germany. Grant No.: 17N3409.

References

[1] S. Asano, T. Maruyama, and Y. Yamaguchi; "Performance comparison of FPGA, GPU and CPU in image processing", International Conference on Field Programmable Logic and Applications, 2009. FPL 2009, pp. 126-131.

[2] W. H. Davidow and M. S. Malone; "The virtual Corporation". New York: HarperBusiness, 1992.

[3] E. Deelman, D. Gannon, M. Shields, and I. Taylor: „Workflows and e-science: an overview of workflow system features and capabilities", Future Gener. Comput. Syst., 25 (2009), pp. 528–540.

[4] BIOWIC, Bioinformatics Workflow for Intensive Computation, http://biowic.inria.fr/workflows/shrec.html 05.05.2012.

[5] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing", Journal of Grid Computing, Vol. 3, No. 3-4, pp. 171-200, 2005.2. Oxford: Clarendon, 1892, pp.68–73.

[6] X. Liu, D. Yuan, G. Zhang, W. Li, D. Cao, Q. He, J. Chen, and Y. Yang; "The Design of Cloud Workflow Systems", SpringerBriefs in Computer Science.

[7] P. Mell and T. Grance, National Institute of Standards and Technology (NIST), "The NIST Definition of Cloud Computing", Special Publication 800-145, September 2011.

[8] Open Grif Forum (OFG), Open Cloud Computing Interface (OCCI), June 2011.

[9] M. Gerhards, A. Belloum, F. Berretz, V. Sander, and S. Skorupa: "A History-tracing XML-based Provenance Framework for Workflows", The 5th Workshop on Workflows in Support of Large-Scale Science (WORKS), November 2010.

[10] R. M. Shapiro, Workfl ow Management Coalition Working Group One, "XPDL 2.1 Integrating Process Interchange & BPMN", January 2008.

[11] D. Jordan and J. Evdemon, "Web Services Business Process Execution Language Version 2.0 (BPEL)", OASIS Standard, April 2007.

[12] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches", Wireless Communications and Mobile Computing.

[13] S. Cantor, J. Kemp, R. Philpott, and E. Maler, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, 15 March 2005.

[14] K. Lawrence and C. Kaler, "WS-Trust 1.3 OASIS standard", March 2007.

[15] A. Celesti, F. Tusa, M. Villari, A. Puliafito: "How to Enhance Cloud Architectures to Enable Cross-Federation", 3rd International Conference on Cloud Computing (CLOUD), 2010, pp. 337-345.

[16] Y. L. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance in e-Science", SIGMOD RECORD, vol. 34, 2005.

[17] M. Gerhards, V. Sander, T. Matzerath, A. Belloum, D. Vasunin, A. Benabdelkader: "Provenance Opportunities for WS-VLAM: An Exploration of an e-Science and an e-Business Approach", The 6th Workshop on Workflows in Support of Large-Scale Science (WORKS), November 2011, pp. 57-66.

[18] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. Van den Bussche, "The Open Provenance Model Core Specification (v1.1)," Future Generation Computer Systems, vol.27(6) pp.743-756, June 2011.

[19] History-tracing XML for an Actor-driven Grid-enabled Workflow System, http://www.fh-aachen.de/en/research/projekt-hixforagws/ 05.05.2012

[20] B. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti: "CloneCloud: Elastic Execution between Mobile Device and Cloud", Proceedings of the sixth conference on Computer systems (EuroSys '11), 2011, 301-314.

[21] Generic Workflow Execution Service (GWES) http://www.gridworkflow.org/kwfgrid/gwes/docs/ 05.05.2012.

 ISBN: 978-1-61208-216-5

HPCCA: Is efficient in Mobile Cloud Environment (MCE)?

Dr. Khalid Mohiuddin, Asharul Islam, Ashiqee Rasool Mohammad
Department of Information System
College of Computer Science, King Khalid University
Abha, Saudi Arabia
(drkhalidmk70, ashar.islam, ashique.rasool)@gmail.com

Aftab Alam
Department of Computer Science
College of Computer Science, King Khalid University
Abha, Saudi Arabia
aftabjh@gmail.com

***Abstract*—Integrating cloud infrastructure and services into the Mobile Communication Environment (MCE) is an intensive research area nowadays. Mobile cloud computing provides interesting research opportunities to resolve the boundaries between mobile and cloud computing. Studies show that mobile devices are limited in resources: memory, network bandwidth, availability and specifically processing power. It does not meet the demand of high performance applications for mobile users. One obvious solution to this requirement is to get the processing power as a service from a resource-rich environment. Cloud computing is a service-based approach which provides the required computing resources to its subscribed users: on-demand, scaled elastically, and economically feasible in response to user's requirement. The cloud services which facilitate mobile environment describes as Mobile Cloud Computing (MC2). Existing advanced mobile devices can perform various multimedia applications (e.g., M-Commerce, Health Care, Games, Rich Media, etc.) and provide a number of utilities; they are not efficient for executing intensive computing applications such as advanced 3D Games, scientific calculations, result optimizer, high definition weather forecasting and many more. These applications require high processing power, intensive memory mapping and sufficient network support for efficient execution. Currently some IT giants like IBM, Rackspace, Penguin Computing, Sara, PureWeb and Sabalcore are providing High Performance Cloud Computing Applications (HPCCAs) over the cloud infrastructure. The HPCCAs shall be potential services for existing smart mobile device users. In this paper our focus is to analyze and present a comprehensive study to observe: Is utilization of HPCCA efficient in mobile environment? In our analysis, we considered different aspects and consequences of High Performance Computing (HPC) in mobile cloud environment. For example, High bandwidth, Signal quality, Mobility, Service availability and Security concerns. Through this analysis, we found HPCA is efficient in mobile cloud environment.**

Keywords- Mobile Cloud Computing; High Performance Cloud Computing Applications; High Performance Computing; Smartphones.

I. INTRODUCTION

More than decades of research on computational performance in traditional Information Technology (IT), the focus is now shifted towards the computation and communication resources as a service on-demand, over the internet, pay-on-usage. Availability is the vital metric for these resources; near 100% availability is becoming mandatory for both intensive users and service providers. Computational needs of users (desktop and mobile) increasing to the alarming stage. They need strong support of technology and its providers to meet their needs; particularly for High Performance Computing Applications (HPCAs). The conventional computing technology does not have enough potential in mobile environment with resource limited devices. Technology evolves and new integrated service-centric technologies emerged to offer high quality services specifically for HPCA in mobile environment.

Cloud computing has been emerged as a new service-centric technology. Offers service on-demand, elastic provisioning, reliability, security and pay-per-uses economic model. Cloud computing exists if tasks and data are kept on the internet rather than on individual devices, providing on-demand access. Data is provided and managed by the service providers. Applications run on a remote server and then sent to the user [1]. According to NIST cloud offered services in the form of Software as service (SaaS), Platform as a service (PaaS) and Infrastructure as Service (IaaS). Cloud users may access the server resources using a computer, netbook, tablet, pad computer, smart phone, or other device. In cloud computing, applications are provided and managed by the cloud server and data is also stored remotely in the cloud configuration [2]. Cloud subscriber need not to invest on high configuration hardware and expensive licensed software. Processing and storage maintained by the cloud service provider with the integration of local service provider on economically feasible model. Cloud extends its service domain with the integration of mobile computing technology. This integrated technology refers as Mobile

Mobile Cloud Computing (MC2), becomes a convenient alternative to personal computers by integrating mobility, communication, software functionality, and entertainment [3]. It offers mobile users great opportunities and turns from resource limited device into a resource-rich environment. It enables mobile cloud users to execute high computing application in potential and flexible environment such as 3D games, scientific calculations, result optimizer, high definition weather forecasting, and many more. With the influential growth of intensive mobile applications,

 ISBN: 978-1-61208-216-5

developers are shifting from desktop computing to mobile cloud computing environment.

The remaining structure of this paper as follows: In Section II, we explain existing infrastructure of HPCAs, in both fixed/static and mobile cloud environment. Section III discusses the importance of offloading for HPCCAs in desired environment. Section IV describes the required service-architecture of HPCCA in mobile environment. Section V presents HPCAs offerings by different service providers. In Section VI, we also discuss HPCCA economics. In Section VII, we discuss the much needed security and challenges issues and best suited solutions; result analysis. Finally, we conclude with our findings and future work.

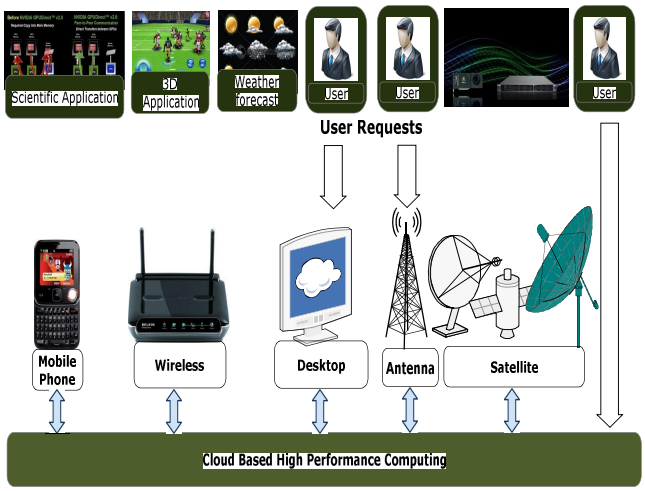


Figure 1. A software system that integrates mobile and cloud computing services.

II. EXISTING INFRASTRUCTURE FOR HPCA

HPCA basically needs intensive computing, accelerations, efficient parallel computing algorithms, and bridge between software and hardware; includes a primary HPC and a backup HPC connected by a robust communications and secured IT infrastructure. In the early age of computing, the high performance computing is done by supper computers. It requires a large amount of electrical power for its infrastructure and need significant cost to maintain. In general, scientific and commercial organizations handle the HPCAs by the operating system in powerful computers or by the cluster-servers. High-performance analytics enable organizations to quickly and confidently seize new opportunities in order to make better choices ahead of competitors and create new value from big data. It enables organizations to handle their most difficult challenges, quickly generate high-impact insights and transform their operations [4]. With the computing technology evolution, the infrastructure keeps on changing. Large organizations like IBM releases technically configured powerful systems to meet the increasing demand of HPC. The world increasingly global and highly interconnected planet; needs communication and computation technology on move, anywhere, any time. In the present information age, acceleration of data processing is growing dynamically and influenced markets to deploy HPC for their applications.

The HPC architecture requires intensive applications to rum on multiple processors, rather than on single, to achieve the desired performance. Virtualization, parallel and vector processing, multi and co-processing are the fundamentals of HPC. Applications such as, 3D- imaging processing, financial commerce, medical imaging, data compression, seismic data interpretation, search, security, and many more have been efficient in desktop environment. Shall be the same in mobile environment? Today, many service providers offer HPCA as a service, available for enterprise users, community users, and individual subscribers in mobile cloud environment.

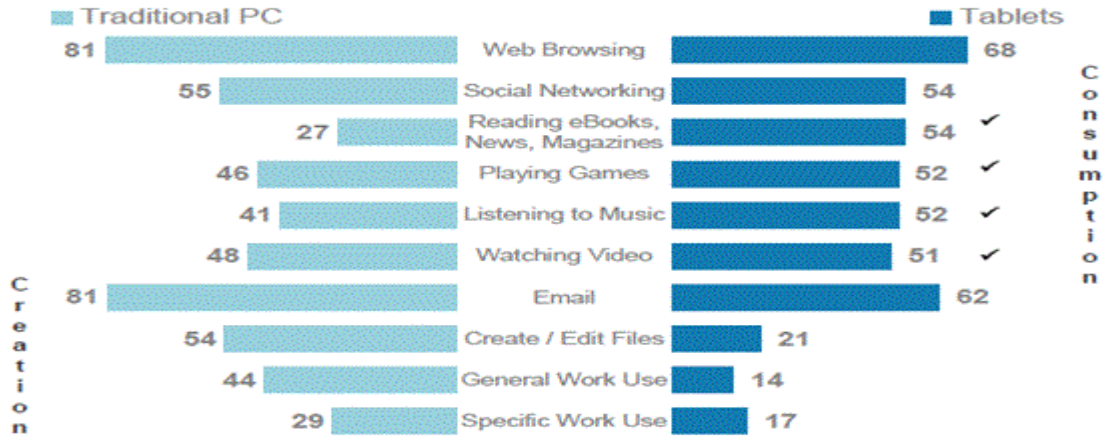


Figure 2. Percentage of users who uses wearable devices for each activity and shows content consumption [5].

 ISBN: 978-1-61208-216-5

A. HPCAs in Mobile Cloud Environment

In present scenario, the usage of mobile phone applications is potentially increased, due to the integration of mobile technologies with the cloud. Cloud delivers services to its remote users over the IP network often through a web browser without referring to the boundaries. Mobile computing technology whereas connects its users in mobile or non-static environment across the network(s). This is accomplished by connecting mobile computing activities wirelessly through the internet or a private network. This connection ties the mobile device to centrally located information and application software by using battery powered, portable, and wireless computing and communication devices. This encourages manufactures, vendors, and service provides to develop an efficient mobile environment for intensive computations and quality-communications. Present mobile phones are heavily used for executing high performance applications such as 3D applications and scientific calculations. However the limitations exist in resources are the basic obstacles to execute these applications efficiently. These limitations can be removed or minimized with the integration of resource-rich, reliable, service-centric cloud technology.

The integrated (mobile and cloud) technology describes as MC2 deploys in heterogeneous radio access environment such as WiFi, 3G, WLAN, WiMax, GPRS. It is implemented through wireless connectivity. The prime features are access 24X7, on-demand, energy efficient, and economically feasible even for low data rate cloud controlling signals. Mobile applications can be launched on the device or cloud, and can be migrated between them according to dynamic changes of the computing context or user preferences [6]. Smartphone and tablets are quickly becoming the information worker's most valuable tools. Young workers and their strong affinity for go-anywhere technology is changing the shape of the enterprise right here, right now!! Enterprise Mobility is becoming more and more Anytime, Anywhere service in a true sense!! Smart-phones [7].

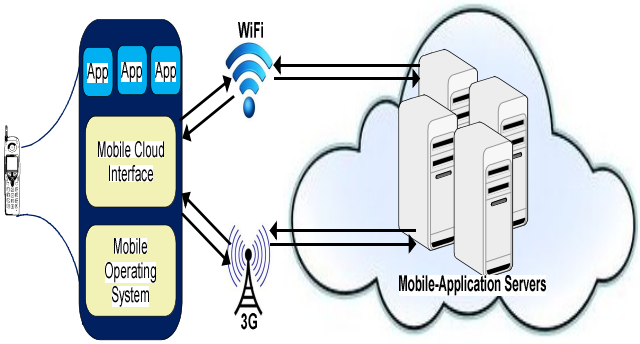


Figure 3. Block Diagram for HPCCA Services.

The most frequent and intensive users of HPC systems are researchers from academic and research oriented business establishments, investigating and specific government agencies; whereas the common users heavily use HPCAs for communication, entertainment and such many more activities. All need effective, improved, convincing, reliable and efficient performance in both, fixed/static and mobile cloud-based environment.

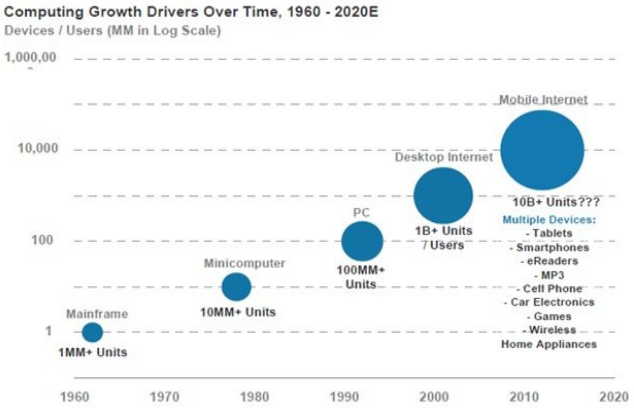


Figure 4. Expected growth of usage of mobile internet [8].

B. Performance Parameters for efficient Mobile Cloud Environment

Advancing the efficient HPC architecture is a big challenge, particularly in mobile cloud environment. It includes processing, managing, using existing architecture, mobility, and offering services anywhere anytime from individual user to enterprise subscription locally and globally. The following features are most desirable for efficient HPCCA:

Availability: HPC system needs to be clock-driven; highly available (99.9%), network, data centers, and at much lower cost.

Scalability: It needs to have an infrastructure that provides expandable resources to accommodate heavier load, high throughput, scalable storage, and reliable communication.

Intensive-scale computing: It is the ability to run massively parallel code of instructions with the simulation of data.

Life cycle management: The efforts to maximize the efficiency of the transition operations throughout the processing cycle.

Software configuration and management: Updates and adopt the software standards that accommodate sharing of code internally and externally with other partners.

According to Lawrence Berkeley National Laboratory, the following features are vital in mobile cloud environment:

- A global shared memory abstraction
- Support dynamic updates
- A high-bandwidth, low-latency network
- Ability to exploit fine-grained parallelism
- Support for light-weight synchronization
- Massively Multithreaded architectures

Symmetric multiprocessors

III. OFFLOADING ESSENTIAL FOR HPCCA IN MOBILE ENVIRONMENT

Offloading is the process of using complementary devices which are resource-rich, for accelerating the processing originally targeted for resource-limited mobile devices. Offloading technology must be an essential tool in the desired environment. Offloading of a mobile computing

 ISBN: 978-1-61208-216-5

task is a tradeoff between the energy used for local processing and the energy required for offloading the task, uploading its data, and downloading the result, if necessary. One can express the offloading energy trade-off with the formula $E_{trade} = E_{local} - E_{delegate} > 0$, where E_{local} is the energy used for complete local execution, and $E_{delegate}$ is the energy used from the perspective of the mobile device if the task is offloaded. If E_{trade} is greater than zero, then there is an energy benefit for delegating the task to the cloud [9].

A. Factors affecting offloading

Network traffic: High bandwidth is required for fast connection to the cloud through internet.
Security Aspects: Secured service access mechanism should be ensured.
Business Models: Business models need to be modified to adopt the cloud computing standards.
Accessibility: Robust Infrastructure needs to be placed for providing high accessibility to the services.

B. Parameters for offloading decision

Power Consumption: Power consumption can be a vital parameter for offloading decision as battery life time is the major concern for mobile devices.
Processing Requirement: Processing requirement is an important parameter for offloading decision. The applications requiring intensive processing shall be offloaded.
Storage and memory requirement: Memory and storage requirement is a major parameter. Applications requiring huge memory and storage cannot be executed on mobile devices as they are generally poor in the resources; these applications need to be offloaded.

Latency and bandwidth: Latency also plays an important role in offloading decision. The interactive applications cannot support high latency. Offloading decision shall be taken on the available bandwidth.
As shown in figure 5, percentage of total mobile data traffic from handsets and tablets, mobile offload will be 31 percent (3.1 Exabyte/month) in 2016.

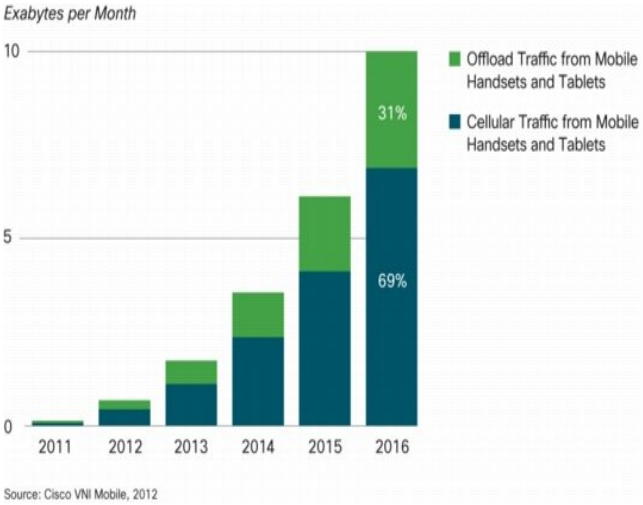


Figure 5. Offloading trend in mobile cloud environment [10].

IV. HPCCA Service Architecture

The service architecture shown in Figure 5 describes how the cloud resources can be utilized by mobile devices for executing high performance applications.

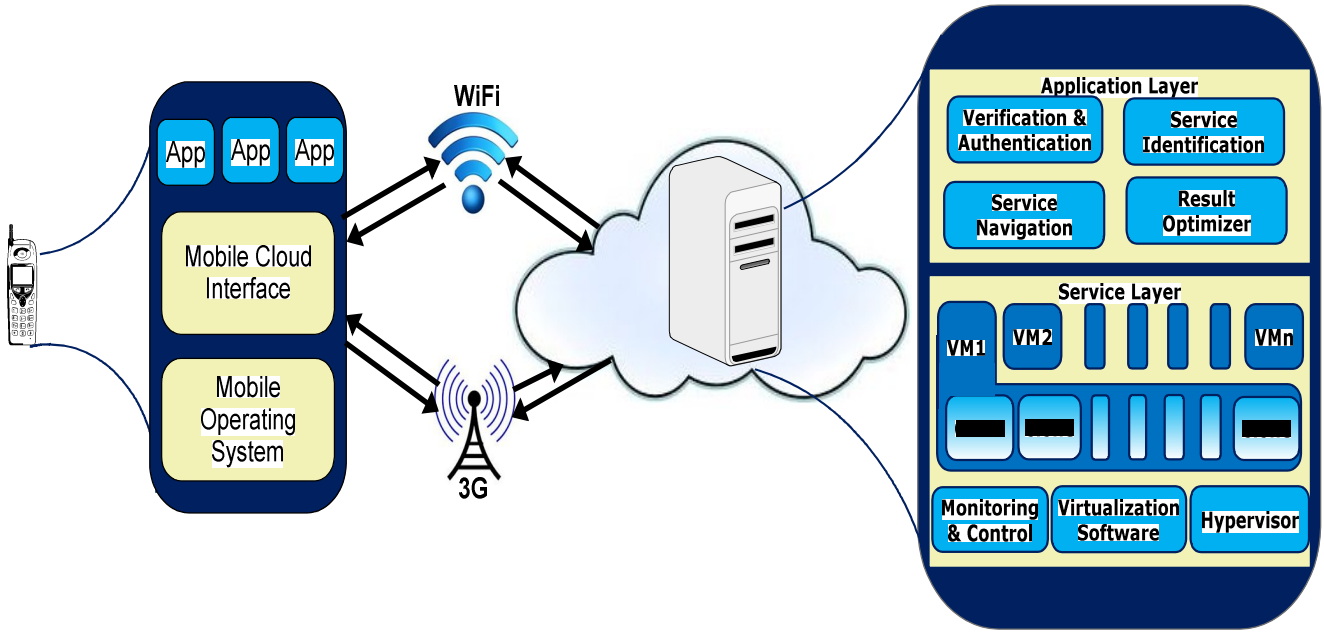


Figure 6. Service Architecture in Mobile Cloud Environment.

 ISBN: 978-1-61208-216-5

The mobile device sends request to the cloud through mobile cloud interface available in the device itself. The request is first verified and authenticated by the verification and authentication module then it passes to the service identification module. This module categorizes the request as per the nature of processing, if the request requires parallel processing then the parallel segments shall be assigned to individual virtual machines else it will be assigned to any of the available clone. Once the process is done, the individual results shall be forwarded to the result optimizer. It combines the individual results, formulates it, and sends back to the mobile client through the service navigation module. The virtual machines are created with the help of hypervisor and virtualization software; it is controlled and monitored by the monitoring and control unit. The clones can be created using any of the existing cloning technology shown in Figure 6.

V. HPCA SERVICES OFFERED IN MOBILE CLOUD ENVIRONMENT

In the technically advanced competent environment, many cloud service providers offer huge benefits for their users in response to their growing needs of HPCAs in mobile environment. Focus is gradually shifting, services offer on-demand, over the internet, through web-browser, pay-as-you-consume, at very low economic subscription. All most all the providers emphasis on HPC, software, infrastructures, data storage, networking, and special attention to privacy, integrity, and security of data, shown in the table1. In the table, column1 shows the services offered by the respective providers. A substantial number of providers have contributed to the development of this technology and many in the adoption process shown in Table 1.

TABLE 1. SERVICES OFFER BY THE RESPECTIVE PROVIDERS IN CLOUD-BASED MOBILE ENVIRONMENT

Services	**Service Providers-HPCCAs in Mobile Environment**					
	Sara	**PureWeb**	**Rackspace**	**Sabalcore**	**BlueCoat**	**Penguin**
Computation	HPC, Grid	HPC	HPC	HPC	HPC	HPC
Software	ADF, BLAS, HDF, BLACS, DMF, FFTW,	Software Transformation KITs	Software as a Service for Business	Open source software	Bespoke Service	Scyld ClusterWare
Infrastructures	Grid, CPU cluster, Lisa Huygens, System	Cyber Infrastructure	High Performance configuration	High Performance configuration	Proxy AB 1400-2400, AND, MACH5	POD, Hybrid, Private and dedicated Cloud
Data Storage	Grid Permanent, Data Services	Data Storage	Hosting Storage	Ample Permanent Storage	-	HPC Datacenter, POD
Networking	High Performance Networking	High Performance Networking	High Performance Networking	High Performance Networking	High Bandwidth 300X	High Performance Networking
Security	Data Security	Mobile Data Security	Data Security	-	ProxySG, web filtering solutions	Strong Security

A. *Efficiency Parameters of HPCCAs in Mobile Environment*

As we discussed earlier, the limitations of executing HPCAs in mobile environment shall be minimize significantly by implementing cloud service-model in turn maximize efficiency. The nature of cloud services is best suited for efficient processing of HPCAs in mobile cloud environment. The following are the most promising features need to be considered for an efficient mobile environment:

- High Data Rate
- Quality of Service (QoS)
- Scalability
- Availability
- Mobility
- Security
- Network latency

B. *Service characteristics of HPCCAs in Mobile Environment*

1) HPC Ubiquitously: By the subscription of HPCCA services, user can access intensive HPAs on their smart mobile devices in heterogeneous environment, without considering processing burden. During the process mobile devices act as interface, send the instructions, and request processing is done on cloud infrastructure. Users have full advantages of access these services from resource limited device to a resourceful environment conveniently.

2) Right to use – anyplace, anytime: Cloud computing with the integration of mobile computing provides services anywhere anytime with the application of adequate Service

Level Agreement (SLA). It includes flexible mechanism for delivering IT services at each level of the computing stack: from hardware level to application level [11]. Initially, HPCCA software was supporting desktop environment and users are free to work anywhere through the internet. With the amount of mobile applications increased, the developers extend its reach for mobile access with the devices like laptops, notepad, tablets and smartphones. In the present, era any software application can be execute by mobile cloud users anywhere anytime.

3) Platform Support: Another promising feature of cloud is HPCAs can execute irrespective of platform dependencies. PureWeb integrates directly into your existing Microsoft Foundation Class, C#, C++ or Java code, bringing the web to your application rather than your application to the web. Furthermore, adding support for the latest mobile touch-based devices such as an Apple iPad, iPhone or Google Android devices is seamless [12].

4) Data Security and Compliance: Major concern of mobile cloud users is security; data is mostly secured in static environment and remains uncertain in mobile environment. Service providers do not comprise on any less secure system for securing and handling data.

VI. ECONOMICS FOR HPCCA

According to Microsoft, the overall cost of IT is determined not just by the cost of capacity, but also by the degree to which the capacity is efficiently utilized. It is needed to assess the impact that demand aggregation will have on costs of actually utilized resources (CPU, network, and storage) [13]. Many organizations realized the impact of low cost offering by the deployment of cloud services. Cloud Computing has been emerged as an economic service-centric technology; combines the best economic properties of mainframe and client/server computing, and shifting the economics of traditional IT. The architecture of cloud facilitates elastic consumption and pay-as-you-consume pricing model. Resource-intensive computing is offloaded to the cloud to leverage the cost advantages of massive data centers [14]. Efficient multi-tenancy is a major factor, increases number of tenants, maximize application processing, minimize the applications management and server cost. Recently, Microsoft joined Google and Amazon Web Services in cutting the cost of cloud services. Microsoft dropped the price on its Azure Storage Pay-as-you-Go service and lowered the price of its six-month storage plan. The cost to use Azure Extra Small Compute has dropped in half [15]. In cloud paradigm, resource-intensive computing is offloaded to the cloud to leverage the cost advantages of massive data centers. Researchers are working on different techniques that minimize the cost of using cloud resources, provide efficient and seamless environment while maintaining user satisfaction. According to PureWeb, the following activities slash down the cost significantly:

- HPCC will slash web migration cost
- The risk & expense of traditional migration
- Fast & straightforward with HPCC
- No need for expensive & risky rewrites
- No licensing fees or proprietary downloads
- Significant hardware saving
- High speed for any high performing application

VII. SECURITY CONCERNS

Security, privacy, and integrity, of data and applications are major concerns in mobile cloud environment. It is quit known fact, data is more secure in static rather than in mobility. To design an efficient HPCCA-service system in mobile environment, various challenges such as high computation, scalability, availability, mobility, and cost restrictions need to be addressed. Cloud computing fits well as an enabling technology in this scenario as it presents a flexible stack of computing, storage and software services at low cost [16]. These challenges can be tackled by leveraging various cloud services in HPCCA-service system. The major constraints, HPCAs require significant computing power, need to process from a limited energy source mobile device. It is essential to outsource intensive computing applications to cloud. Offloading seems to be simple solution; it is non-trivial, since wireless network bandwidth and latency are also big challenge need to be address. According to Alcatel-Lucent and Techzine, four key strategies should be considered to overcome the challenges of mobile cloud computing are 1) Processing time at the data center 2) Processing time on the device 3) Network latency 4) Data transport time.

VIII. EFFICIENCY RESULT ANALYSIS OF MOBILE VS FIXED DEVICES

Connectivity is almost guaranteed in fixed/static networks and potentially rich in resources; whereas mobile environment intensely depend on network bandwidth and latency. In processing of HPCAs, both mobile and fixed devices require intensive computation, significant amount of energy; and consume heavy resources. Efficiency exists in fixed/desktop environment and shall be improve for mobile environment also with the integration of mobile technology with cloud. Although the current generation mobile devices have significantly improved in technology and support Service Oriented Architecture (SOA), need for efficient performance in mobile cloud environment. Market updates, eying the users need and requirements, especially for HPCAs. A comparative study shown by Kyung Mun, of a Dell Inspiron 580 desktop with the iPhone 4 and iPad, for example, reveals the tradeoff cost of mobility. As compared to a fixed device, mobile devices in general have:

- 3 times less processing power
- 8 times less memory
- 5 times less storage capacity
- 10 times less network bandwidth

While mobile device performance will continue to improve in absolute terms (Figure 7), the disparity between the resource constraints of mobile and fixed devices will remain and must be accounted for in the types of application selected for mobile cloud computing [17].

 ISBN: 978-1-61208-216-5

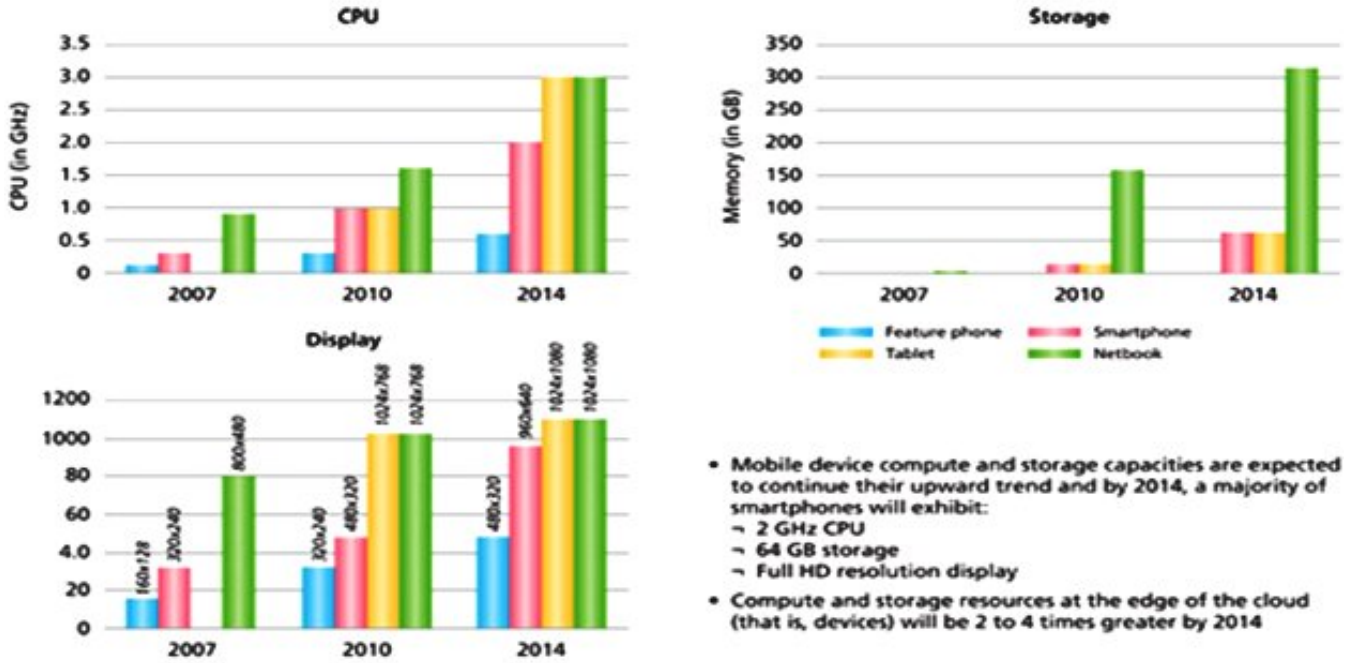


Figure 7. Mobile device computing storage and display trends [18].

CONCLUSION AND FUTURE FOCUS

In this work, our interest was to measure the processing performance of HPCAs on existing infrastructure in fixed/static and mobile environment. Compare the efficiency and investigate: shall HPCAs be potential for existing mobile device users in cloud-based environment. Accordingly, we presented HPCA architecture, resources consumption on both traditional PC and mobile platform. The efficiency shall be achieved on resource-limited devices for HPCAs with the integration of service-centric cloud technology. We presented service architecture for high performance computing applications in mobile cloud environment. Further we analyzed the HPCCA services offered by cloud providers and the characteristics in mobile environment. One of the common interests of HPCCAs mobile device users and providers is the business-economic, discussed in detail. We compared the efficiency performance in our result analysis and concluded with the concerns of security and challenges; common for both users and cloud service providers.

REFERENCES

[1] http://www.smartdevelopments.org/?p=84, [retrieved: May, 2012]
[2] http://www.smartdevelopments.org/?p=84, [retrieved: May, 2012]
[3] Hyun Jung La and Soo Dong Kim, “A Conceptual Framework for Provisioning Context-aware Mobile Cloud Services”, 978-0-7695-4130-3/10 $26.00 © 2010 IEEE DOI 10.1109/CLOUD.2010.78.
[4] High-Performance Analytics from SAS, http://www.sas.com/software/high-performance-analytics/index.html, [retrieved: May, 2012]
[5] Morgan Stanley blue paper, February 14, 2011. http://www.morganstanley.com/views/perspectives/tablets_demand.pdf, [retrieved: May, 2012]
[6] Le Guan, Xu Ke, Meina Song, and Junde Song, “A Survey of Mobile Cloud Computing”, 2011 10th IEEEACIS International Conference on Computer and Information Science (2011).
[7] http://www.smartcloudinfotech.com/enterprise_mobility.php, [retrieved: May, 2012].
[8] Morgan Stanley blue paper, February 14, 2011. http://www.morganstanley.com/views/perspectives/tablets_demand.pdf
[9] Eemil Lagerspetz, and Sasu Tarkoma “Mobile Search and the Cloud: The Benefits of Offloading”, http://www.cs.helsinki.fi/u/lagerspe/publications/mobile-cloud.pdf, [retrieved: May, 2012]
[10] CISCO White paper, http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html, [retrieved: May, 2012]
[11] Christian Vecchiola, Suraj Pandey, and Rajkumar Buyya, “High-Performance Cloud Computing: A View of Scientific Applications”, 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, 978-0-7695-3908-9/09 $26.00 © 2009 IEEE DOI 10.1109/I-SPAN.2009.150.
[12] “Calgary Scientific Revolutionizes Application Sharing and Advanced Collaboration with PureWeb® 3.0” http://webcache.googleusercontent.com/search?q=cache:2ZF0D9mzEfwJ:www.getpureweb.com/files/Infotables/pureweb-[retrieved: June, 2012], df+&cd=1&hl=en&ct=clnk&gl=sa, [retrieved: June, 2012]
[13] “The economics of the cloud by Microsoft, November 2010.” http://www.microsoft.com/en-us/news/presskits/cloud/docs/The-Economics-of-the-Cloud.pdf, [retrieved: June, 2012]
[14] Kyung Mun, “Mobile Cloud Computing Challenges” Corporate Technology Strategist, Alcatel-Lucent- SEP 21 2010.
[15] Nancy Gohring, “Cloud economics improving for users in wake of price cuts, March 09, 2012” http://www.infoworld.com/d/cloud-computing/cloud-economics-improving-users-in-wake-of-price-cuts-188386.
[16] Suraj Pandeya, William Voorsluys, Sheng Niua, Ahsan Khandokerb, and Rajkumar Buyya, “An autonomic cloud environment for hosting ECG data analysis services”, 15-05-2011 Elsevier B.V.
[17] Kyung Mun, “Mobile Cloud Computing Challenges” Corporate Technology Strategist, Alcatel-Lucent- SEP 21 2010.
[18] Kyung Mun, “Mobile Cloud Computing Challenges” Corporate Technology Strategist, Alcatel-Lucent- SEP 21 2010.

 ISBN: 978-1-61208-216-5

Intercloud Object Storage Service: Colony

Shigetoshi Yokoyama, Nobukazu Yoshioka

GRACE Center, National Institute of Informatics, Tokyo, Japan
{yoko, nobukazu} @nii.ac.jp

Motonobu Ichimura

NTT DATA Intellilink, Tokyo, Japan
ichimuram@intellilink.co.jp

Abstract— **Intercloud object storage services are crucial for inter-organization research collaborations that need huge amounts of remotely stored data and machine image. This study introduces a prototype implementation of wide-area distributed object storage services, called colony, and describes a trial of its cloud storage architecture and intercloud storage services for academic clouds.**

Keywords-Cloud computing; Object storage service; OpenStack; Intercloud; Cloud federation.

I. INTRODUCTION

Cloud computing has the potential to dramatically change software engineering. It allows us to manage and use large-scale computing resources efficiently and easily. Moreover, it makes it possible to develop new software by using these resources for scalability and lowering costs.

For example, users can prepare machine images of standard education environments on Infrastructure as a Service to manage the environments efficiently. We have developed edubase Cloud [1], a cloud platform based on open-source software and using a multi-cloud architecture.

We are now developing a research cloud based in part on our experience in managing the edubase Cloud service during the disaster recovery efforts after the Tohoku earthquake and tsunami in March, 2011. Intercloud object storage services that can store machine images and research data remotely are crucial for such a development. Furthermore, if academic clouds are independently deployed and managed, there would be no way for users to continue working within clouds affected by disasters or other outages. By using intercloud object storage services, users can utilize machine images in other clouds operating normally.

We have developed an intercloud storage service architecture and a working prototype called colony [2]. This paper describes this development. Section 2 describes user scenarios on how to use intercloud object storage services. Section 3 presents a comparison with other storage services. We discuss the design and prototype of the intercloud object storage architecture in section 4 and 5, and conclude in Section 6.

II. USER SCENARIOS

The following are academic–cloud-user scenarios for intercloud storage services. In the scene depicted in Figure 1, there are two academic clouds, A and B, providing the intercloud storage service. The users of these clouds can store objects in local storage, i.e., storage-A or storage-B, or in the remote object storage, storage-I. Users just have to change the container attribute from local to remote or vice versa.

Storage-I should be geographically distributed for the sake of availability.

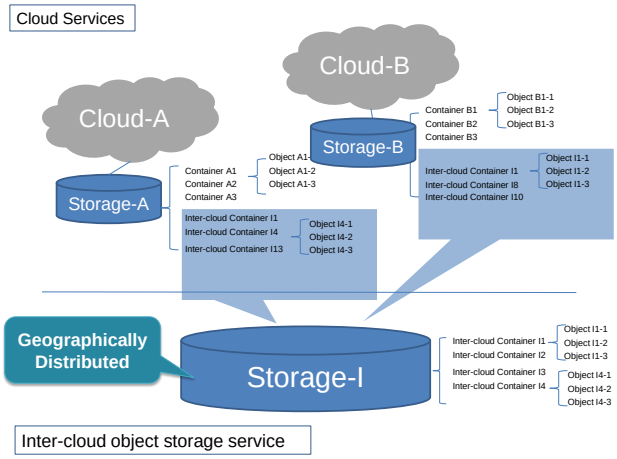


Figure 1. Intercloud object storage service.

A. Access one's own objects from remote clouds

Academic cloud users can access their own containers and objects from clouds that are remote from the one they usually use. The machine images stored as objects in storage-I can be used to launch virtual machines in these remote clouds. Machine image conversion might be needed before the launch, depending on the heterogeneity of the source and destination clouds.

B. Access objects of other users

Academic cloud users can share containers and objects with other users who may access them from remote clouds. The objects could be, for example, machine images or research data.

C. Single sign-on to object storage services

Each object storage service manages its own users but if each manages its users independently, users would have to login to a service every time they want to receive it. To deal with this problem, we support single sign-on among services by using a standardized identity management service such as shibboleth [2].

 ISBN: 978-1-61208-216-5

III. RELATED WORK

We thought that we should not start developing our intercloud storage service from scratch and that it would be better to utilize existing open source object storage service software. Figure 2 compares the various candidates that we examined in focusing on AWS S3 type Web API base object storage open source projects. S3 is a de-facto standard among object storage services, and there is a software eco system around it.

.

	baltic-avenue	board walk	fs3	Rado sgw	sinatra -s3	swift	Walrus
Redundancy mechanism	△	△	-	△	-	×	-
Max data size	1M	∞	∞	∞	∞	5G	5G
Max number of data	1000	∞	∞	∞	∞	∞	finite
Error correcting	×	×	-	×	-	×	-
ACL	×	-	×	×	×	×	×
Cache mechanism	-	-	-	-	-	-	-

×: OK, - : NG, △: redundacy mechanism with S3

Figure 2. Object storage service projects comparison.

Baltic-avenue [3], boardwalk [4], fs3 [5], sinatra-s3 [6] are effectively development test beds for S3, because they are not designed to have redundancy mechanisms. Because of this limitation, they cannot support huge intercloud object storage services.

Radosgw [7] is a web API front-end of the ceph distributed file system [8]. Walrus is a component of Eucalyptus [9], and although it is compatible with S3, it does not have a redundancy mechanism either.

Swift [10] supports large object storage services in commercial public clouds.

The above considerations led us to study OpenStack swift and modify it for our intercloud object storage service.

IV. DESIGN

A. OpenStack swift

OpenStack Object Storage (code-named Swift) is open source software for creating redundant, scalable data storage using clusters of standardized servers to store peta-bytes of accessible data. It is not a file system or real-time data system, but rather a long-term storage system for large amounts of static data that can be retrieved, leveraged, and updated. Object Storage uses a distributed architecture with no central point of control, providing greater scalability, redundancy and permanence.

Objects are written to multiple hardware devices, with the OpenStack software responsible for ensuring data replication and integrity across the cluster. Storage clusters scale horizontally by adding new nodes. Should a node fail, OpenStack works to replicate its content from other active nodes. Because OpenStack uses software logic to ensure data replication and distribution across different devices, inexpensive commodity hard drives and servers can be used in lieu of more expensive equipment.

Swift has proxy nodes and auth nodes acting as the front-end and storage nodes acting as the back-end for accounts, containers, and object storage.

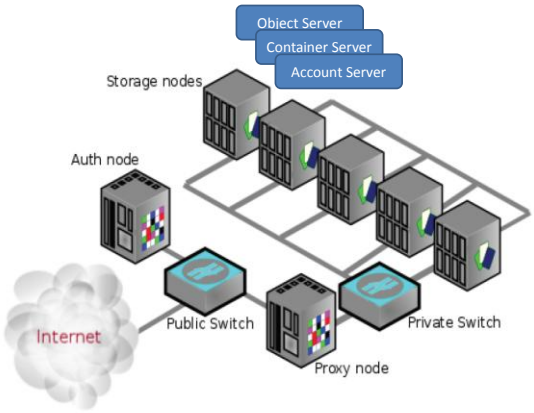


Figure 3. OpenStack swift.

B. Intercloud object storage architecture

Let us begin by discussing the intercloud object storage service architecture by categorizing how to allocate swift components such as proxy nodes, auth nodes, and storage nodes. The proxy nodes and auth nodes categorized as front-end. The storage nodes are categorized as back-end. We examined the suitability of the following architectures.

1. All-in-one architecture
   The front-end and back-end nodes are all on one site.
2. Fan architecture
   One front-end node is on the central site, and the back-end nodes are on each site.
3. Peer-to-peer architecture
   Each site has its own front-end nodes and back-end nodes. The front-end nodes communicate to synchronize the swift rings.
4. Zone architecture
   The front-end nodes have a hierarchical structure similar to the DNS hierarchy and use it to locate storage nodes.
5. Dispatcher add-on architecture
   Dispatchers that can recognize the destination front-end nodes are deployed as an add-on to the front-end.

All-in one, fan, and zone architectures have a single point of failure. The dispatcher add-on architecture is better than a peer –to-peer one because it require fewer servers at each site. Some sites only need to have the dispatcher. These considerations led us to choose the dispatcher add-on architecture.

This architecture has the following advantages:

- Easy to modify swift codes with it
- Easy to extend to more than two swift federations

 ISBN: 978-1-61208-216-5

V. PROTOTYPING

We are now prototyping intercloud object storage service and make the code public as the colony project in github [11] by using the dispatcher add-on architecture which is described in the previous section.

Figure 4 shows an overview of the colony architecture. New components such as swift dispatcher, VM info converter, and caching module were developed by analyzing this prototype. The dispatcher calls the local swift or intercloud swift depending on the container attributes. The VM info converter is used to convert the virtual machine image metadata for one cloud to metadata for another cloud in order to launch the machine image in the other cloud. The content cache helps to make the data transfer efficient.

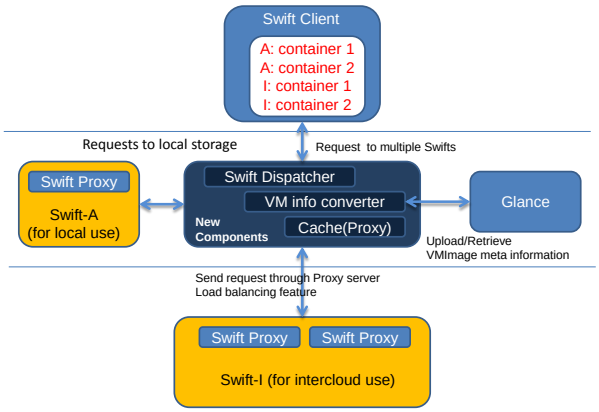


Figure 4. Colony overview.

The swift client can send requests to swift-A and swift-I through the swift dispatcher. In the prototype, the dispatcher can find the destination swift by looking at the prefix string in the container names. In the example in Figure 5, the prefix 'A:' indicates that the container resides in the local cloud, which is 'cloud-A.' The prefix 'I:' specifies that the containers having this prefix are located in the intercloud, which is 'cloud-I.'

When swift sends responses to the client, it merges the response from each swift, as described in Figure 5.

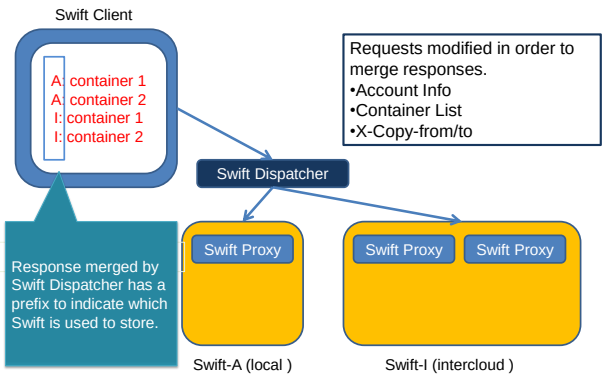


Figure 5. Swift dispatcher.

Swift dispatcher can use a cache proxy per swift proxy to retrieve objects from remote swifts (Fig. 6). In the prototype, the cache is implemented using a squid content proxy cache mechanism [12]. This sort of simple caching mechanism works because the swift proxies in the swift-I are located remotely from the swift client.

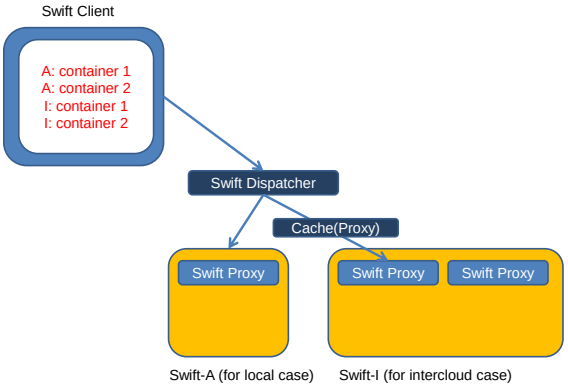
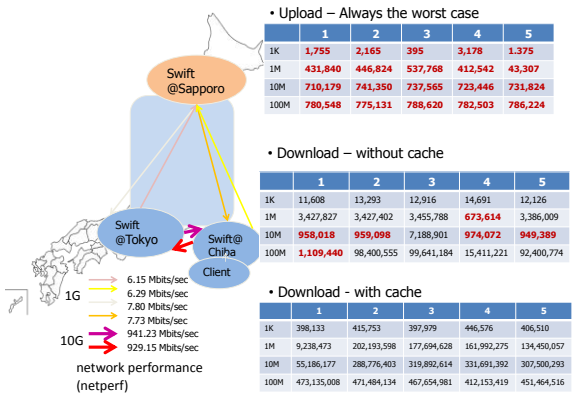


Figure 6. Colony cache.

We implemented a prototype of our intercloud storage service using colony and have started evaluating the performance and usability in three geographically distributed sites. So far, we can say that the colony load balancing seems to contribute to the performance of the intercloud object storage service. We located inter-region swift between three regions, i.e., Tokyo, Chiba, and Hokkaido, and investigated its performance in relation uploading/downloading objects. Throughputs between Tokyo and Chiba were about 1 Gbits/s while throughputs between Hokkaido and Tokyo/Chiba were about 7 Mbits/s.

In this case, uploading of objects is always the worst case because swift proxy puts objects in three zones, sets replication to default, and waits until all objects are uploaded. In contrast, the worst case of downloading objects is one-third of all transactions because the swift proxy randomly chooses one of three object servers. When downloading objects through web cache proxy l, the first download will likely be the worst case, but the results nonetheless show the cache proxy is effective (see Fig. 7).



	1	2	3	4	5
1K	1,755	2,165	395	3,178	1.375
1M	431,840	446,824	537,768	412,542	43,307
10M	710,179	741,350	737,565	723,446	731,824
100M	780,548	775,131	788,620	782,503	786,224

	1	2	3	4	5
1K	11,608	13,293	12,916	14,691	12,126
1M	3,427,827	3,427,402	3,455,788	673,614	3,386,009
10M	958,018	959,098	7,188,901	974,072	949,389
100M	1,109,440	98,400,555	99,641,184	15,411,221	92,400,774

	1	2	3	4	5
1K	398,133	415,753	397,979	446,576	406,510
1M	9,238,473	202,193,598	177,694,628	161,992,275	134,450,057
10M	55,186,177	288,776,403	319,892,614	331,691,392	307,500,293
100M	473,135,008	471,484,134	467,654,981	412,153,419	451,464,516

Figure 7. Uploading and downloading objects performance.

 ISBN: 978-1-61208-216-5

Swift should be zone-aware for geographically distributed use. For example, swift dispatcher can choose the best swift proxy to transfer a request to if it knows the network latency (see Fig. 8).

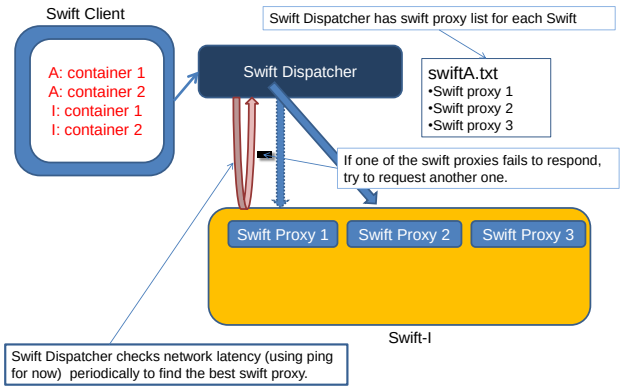


Figure 8. Colony load balancing.

The swift code of the prototype was modified as follows:

- Uploading

Calculate the number of unfinished tasks in the send queue for each area and when one area has much more than the others stop uploading jobs to it.

- Downloading

Check the connection performance of the object servers and try to retrieve an object from the fastest one. Uploading performance improves by utilizing zone awareness (Fig. 9).

object size	1	2	3	4	5
1K	11,356	13,157	13,074	12,758	12,680
1M	9,824,750	11,205,249	7,599,312	10,931,206	11,199,982
10M	52,294,403	51,437,092	51,050,686	52,641,471	52,300,141
100M	97,937,987	101,847,002	102,385,002	102,413,801	101,462,855

Figure 9. Uploading performance with zone awareness.

The VM info converter can be used to share virtual machine image metafiles and is implemented as a swift dispatcher filter (Fig. 10). This implementation enables the shared machine images stored in intercloud storage service to be launched in user specified cloud compute services.

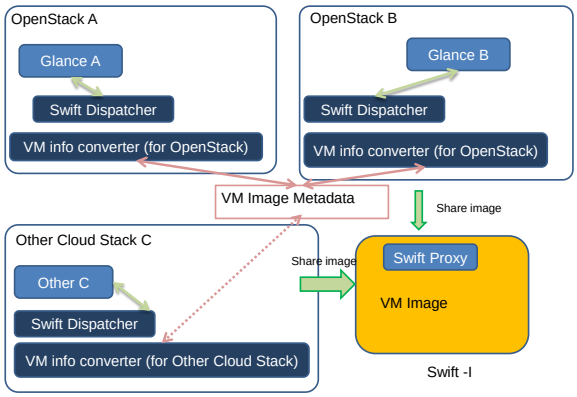


Figure.10. Colony virtual machine image metadata converter.

VI. CONCLUSION

We described an intercloud storage service architecture and prototype using code of the project called colony. The architecture looks feasible, and we will continue to evaluate it in a real environment and enhance the code for better performance.

We already know that there are points in the intercloud object storage service we could tune to get better performance. These points and their evaluations will be reported in the future.

REFERENCES

[1] Nobukazu Yoshioka, Shigetoshi Yokoyama, Yoshionori Tanabe, and Shinichi Honiden , “edubase Cloud: An Open-source Cloud Platform for Cloud Engineers,” SECLOUD '11 Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing, 2011.

[2] Shibboleth: http://www.shibboleth.net/ [retrieved: June, 2012]

[3] Baltic-avenue : http://code.google.com/p/baltic-avenue/[retrieved: June, 2012]

[4] Boardwalk :https://github.com/razerbeans/boardwalk [retrieved: June, 2012]

[5] Fs3: http://fs3.sourceforge.net/ [retrieved: June, 2012]

[6] Sinatra-s3: https://github.com/nricciar/sinatra-s3 [retrieved: June, 2012]

[7] Radosgw: http://ceph.newdream.net/wiki/RADOS_Gateway

[8] Ceph: http://ceph.newdream.net/ [retrieved: June, 2012]

[9] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov, “The Eucalyptus Open-source Cloud-computing System,” 2009 J. Phys.: Conf. Ser. 180 012051.

[10] OpenStack Swift: http://openstack.org/downloads/openstack-object-storage-datasheet.pdf and http://docs.openstack.org/cactus/openstack-object-storage/admin/os-objectstorage-adminguide-cactus.pdf [retrieved: June, 2012]

[11] Colony: https://github.com/nii-cloud/colony [retrieved: June, 2012]

[12] Squid:http://www.squid-cache.org/ [retrieved: June, 2012]

 ISBN: 978-1-61208-216-5

Mobile Cloud Computing Environment as a Support for Mobile Learning

Stojan Kitanov, Danco Davcev
University for Information Science and Technology (UIST) "St. Paul the Apostle"
Ohrid, Republic of Macedonia
stojan.kitanov@uist.edu.mk, dancho.davchev@uist.edu.mk

***Abstract*—This paper presents a new model of mobile distance learning system (MDL) in an extended mobile cloud computing environment (MCC) by using High Performance Computing (HPC) Cluster Infrastructure, as well as some existing videoconferencing technologies enriched with mobile and wireless devices. This MCC model can be applied everywhere where there is need of fast and intensive computing and analysis of huge amount of data, such as modeling of 3D graphics visualization and animation in ecology, global climate solutions, financial risks, healthcare and medical learning, decoding genome projects, etc. After the MCC model presentation, the experimental system architecture will be provided, as well as its possibilities, with particular reference to mobile learning environment and its potential issues. In this architecture the mobile device may optionally use the open source e-learning course management system platform Moodle, to access the learning material and the relevant data that needs to be transferred to the HPC Cluster Infrastructure for further computing. In order to provide higher quality of presenting the learning material, the Cisco WebEx application will be used to test the distance learning in both fixed and mobile environment. Then, a Quality of Experience (QoE) evaluation of such mobile distance learning system will be provided. Finally, it will be concluded that this MCC model that incorporates HPC Cluster Infrastructure can be applied anywhere where there is need of fast and intensive computing and analysis of huge amount of data which cannot be performed by a conventional PC, Laptop or Mobile Device.**

Keywords-Cloud Computing (CC); Distance Learning (DL); Mobile Cloud Computing (MCC); Mobile Distance Learning (MDL); High Performance Computing (HPC) Cluster.

I. INTRODUCTION

Together with the explosive and rapid growth of Internet, mobile networks, mobile applications, and cloud computing, mobile cloud computing is introduced as a potential technology for mobile devices. As mobile network infrastructures continuously improve, their data transmission becomes increasingly available and affordable, and thus they are becoming popular clients to consume any internet web-based applications. Cloud computing provides delivery of services, software and processing capacity over internet, reducing cost, increasing, automating systems, decoupling of service delivery from underlying technology, and providing flexibility and mobility of information. Mobile Cloud Computing (MCC) integrates the cloud computing into the mobile environment and overcomes the obstacles related to the performance (battery life, storage, and bandwidth), environment (heterogeneity, scalability and availability), and security (reliability and privacy) [1]. One future potential application of MCC is the Mobile Distance Learning (MDL), where the students can get the knowledge from centralized shared resources at any place and any time [1], [2].

This paper presents a new Model of Distance Learning System in Mobile Cloud Computing Environment, by using High Performance Computing (HPC) Cluster Infrastructure [3], [4] as well as some existing videoconferencing technologies enriched with mobile and wireless devices. This MCC model can be applied everywhere where there is need of fast and intensive computing analysis of huge amount of data, such as modeling of 3D graphics visualization and animation in ecology, global climate solutions, financial risks, healthcare and medical learning, decoding genome projects, etc. Then, the experimental system architecture of Mobile Distance Learning (MDL) system in Mobile Cloud Computing (MCC) environment will be presented. In this architecture the mobile device may optionally use the open source e-learning course management system platform Moodle [5], [6] to access the learning material and the relevant data that needs to be transferred to the HPC cluster infrastructure for further computing. In order to provide higher quality of presenting the learning material, this architecture uses Cisco WebEx application [7], as well as some existing videoconferencing technologies enriched with mobile and wireless devices such as smart phones, or tablets. The main contribution of the paper is the Quality of Experience (QoE) evaluation of such MDL system in MCC environment.

The paper is organized as follows. Section II summarizes the related work. Section III presents the new model of distance learning system in mobile cloud computing environment. Section IV provides the system architecture of MDL in MCC environment. Section V gives an overview of the Quality of Experience (QoE) aspects of MDL in MCC environment. Section VI presents the QoE evaluation scenarios, while Section VII gives the comparison QoE evaluation results for MDL in MCC environment with respect to the Distance Learning (DL) in the conventional CC Environment. Finally, Section VIII concludes the paper and provides information about future work.

II. RELATED WORK

Cloud computing in mobile platforms has invoked a new wave of evolution in the rapidly developing mobile world. Many mobile devices such as smart phones, PDAs, tablets, pockets PC have been added to the Mobile Cloud Computing (MCC) Environment. Today these mobile cloud applications

 ISBN: 978-1-61208-216-5

(like Google's Map, Gmail for iPhone, and Cisco's WebEx on iPad and iPhone, etc.) are already available [8].

The Mobile Cloud Computing Forum defines MCC as follows [1], [9]:

"Mobile Cloud Computing at its simplest refers to an infrastructure where both the data storage and the data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and mobile computing to not just smartphone users but a much broader range of mobile subscribers".

Mobile Cloud Computing will provide many benefits for cloud computing, mobile network operators, such as increased reach, reduced costs, and reduced reliance on hardware and software equipment. Mobile cloud computing has many advantages among the few listed below:

- Sharing information and applications without the need of complex and costly hardware and software since computations are run in the cloud [10];
- Enhanced features and functionalities of mobile devices through new cloud applications [10];
- Ease of access and development since the access point to mobile cloud computing is through a browser and not a mobile operating system [10];
- Cheaper for cloud computing vendors to build mobile cloud applications because of access to all mobile devices, i.e. one application can be shared and accessed by many mobile device users [10];
- Broader reach, since mobile cloud applications can be accessed through a browser, the cloud computing applications can be reached by all mobile device users, as long as the mobile device has an internet access [10];
- Extending battery lifetime for mobile devices [1], [11], [12], [13];
- Improved data storage capacity and processing power since MCC enable mobile users to store/access the large data on the cloud through wireless networks [1], [14], [15], [16], [17]; and
- Improved reliability since data and computer applications are stored and backed up on a number of computers [1], [18], [19].

However, there are still many obstacles for MCC, including service availability, mobility management, security, privacy, energy efficiency, etc. These problems must be carefully addressed before MCC could become completely operational.

Mobile Distance Learning is seen as one of the potential future applications of MCC [1], [2]. Mobile Learning (m-learning) is one of the applications that can be supported by MCC. Traditional m-learning applications have limitations in terms of high cost of devices and network, low network transmission rate, and limited educational resources [20], [21], [22]. Cloud-based mobile learning (m-learning) applications are introduced to solve these limitations. For example, utilizing a cloud with the large storage capacity and powerful processing ability, the applications provide learners with much richer services in terms of data (information) size, faster processing speed, and longer battery life.

One MCC model that is made up of complex network and relationships of and in between Infrastructure Providers, Application/Services Providers, End-Users and Developers all producing and/or consuming applications and/or services on internet is given in [23]. Of a particular interest in this model are the developers that offer their applications and services on the web via Software as a Service (SaaS) models running on other's hardware (HW) and software (SW) infrastructure providers. However since MCC can be applied in many areas, this model is too general and does not specify any details about MCC implementation.

Microsoft has proposed an HPC Server and Cloud Platform [24]. This platform uses Windows HPC Server 2008 Service Pack 1 that enables service oriented, HPC jobs to be executed as a service using Windows Azure datacenter. High Performance Computing (HPC) gives analysts, engineers, and scientists the computation resources they need to make better decisions, fuel product innovation, speed research and development, and accelerate time to market. Some examples of HPC usage include: decoding genomes, animating movies, analyzing financial risks, streamlining crash test simulations, modeling global climate solutions, computational fluid dynamics (CFD) and other highly complex problems. However this platform is specified only for Conventional Cloud Computing Environment.

Therefore, we propose a new Mobile Cloud Computing Model for Mobile Distance Learning that uses HPC cluster infrastructure. The advantage of the presence HPC cluster infrastructure in the MCC model is that it can be used in situations where the necessary computing cannot be performed by a mobile device, or a conventional PC, or laptop. This model is described in the next section.

III. MOBILE CLOUD COMPUTING MODEL

Our proposed Mobile Cloud Computing Model for Mobile Distance Learning is given on Fig. 1. This Model incorporates High Performance Computing (HPC) Cluster Infrastructure. The communication between the end-user devices (terminals) and the HPC Center is in a cloud computing environment due to the various service requests.

The terminals can be connected to the HPC Cluster Infrastructure inside the University Local Area Network (LAN), or they can be connected on external network (internet). The University Moodle Platform Server (Moodle Course Management System) [5], [6] hosts educational resources and it is connected on the University LAN. The user may access the Moodle platform directly from the University LAN or through the Internet in order to collect the necessary data that needs to be computed by the HPC center. Alternatively, the data that needs to be computed can be collected by the HPC cluster infrastructure throughout the University LAN if the data is too large and cannot be collected by the mobile terminal. When the user wants some data to be computed by the HPC cluster infrastructure it sends request to the HPC center. When the HPC acknowledges the request it receives the data directly from the user terminal or from the University Moodle Platform.

 ISBN: 978-1-61208-216-5

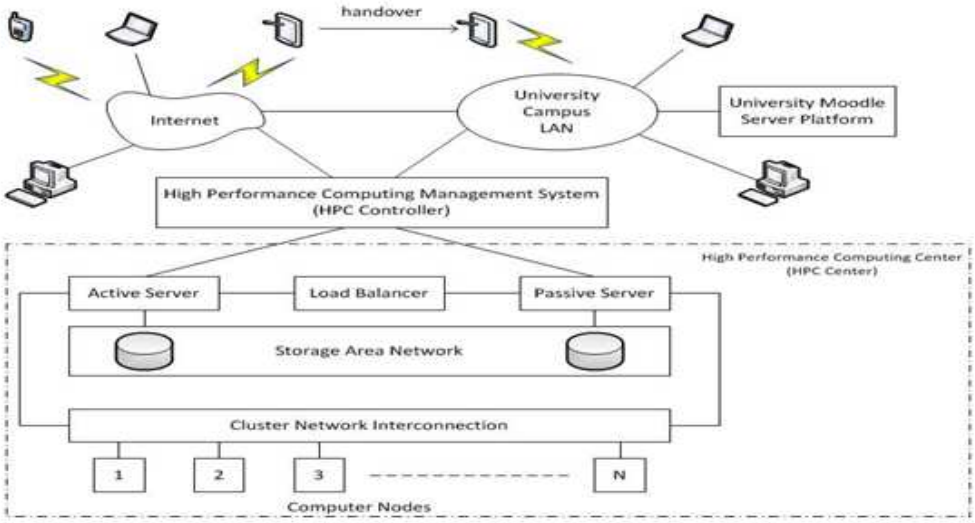


Figure 1. Mobile Cloud Computing (MCC) Model with High Performance Computing (HPC) for Mobile Distance Learning (MDL)

The user can access the HPC center either from University LAN, or directly from internet, through the HPC Management System (HPC Controller). The HPC Controller manages the authorized access to the HPC Center, and it is directly connected on both passive and active server. Like that a redundancy is provided in case the active server goes Out Of Service (OOS). The passive and the active servers are connected to the Load Balancer, which determines which server is active. The Load Balancer also determines which server needs to manage the load (either the active, or both), i.e. the incoming service request from the user.

Both active and passive servers are connected to the storage area network and the cluster network infrastructure. The server takes additional data from the storage area network that needs to be processed (computed), and then it forwards all the necessary data to the cluster network infrastructure for further computing. The cluster network infrastructure consists of N interconnected computer nodes. One of these nodes is the main node, or master node, and it determines which nodes should perform the computing of data. Like that parallel processing is enabled. Once the data computation is completed, the final information is sent back to the user. If the master node fails to operate normally, then another node becomes master node. Like that a redundancy among the nodes is achieved.

The advantage of this model is that it offers new services on mobile devices, as a special benefit from using the HPC center within the mobile cloud environment. In our case, we have provided many statistical calculations connected to the MDL by using HPC center. HPC Cluster infrastructure is useful in situations where the necessary computing cannot be performed by a mobile device, or a conventional PC, or laptop. Another advantage of this model is that it provides service continuity, or seamless mobility as the user handovers from external network to the University Local Area Network.

The next section will present the Experimental System Architecture for m-learning that supports Mobile Cloud Computing. In order to provide higher quality of presenting the learning material, the Cisco WebEx application will be used as an end-user application on the mobile devices.

IV. Experimental System Architecture

The experimental system architecture for Mobile Distance Learning (MDL) that supports Mobile Cloud Computing (MCC) with High Performance Computing is given on Fig. 2. According to this architecture the University classroom is connected to the University Moodle Server Platform, internet and HPC Platform. The University Classroom usually should have the following equipment: PC, or laptop, microphone, speakers, tablet, webcam, projector, and a monitor, or screen. At the University Classroom the instructor presents and delivers the content of the learning material to the students in a classical manner, or via internet to the students that are at home, at work, or simply they are mobile (on the road). The students that are at home, or at work connect to the course by using their PC, or laptop using the high speed internet from their home, or their office. On the other hand, the mobile students (students on the road) use their mobile devices (mobile smartphones, or tablets) to connect to the course via their mobile networks (such as GPRS, UMTS, HSPA, WiFi, WiMAX or LTE).

The University Moodle Server Platform provides possibility to host the digital educational resources, which can be accessed by the instructor and all students either locally, or throughout internet connection. Additionally all students, as well as the instructor over the internet can access the University Moodle Server Platform to collect, or download the data that needs to be computed and to forward it to the HPC platform, for huge amount of data processing in a cloud computing environment. Alternatively, the HPC cluster infrastructure may download the necessary data from the University Moodle Platform, when it receives a request from the authorized user. In the HPC platform, the HPC controllers process the users' requests. The HPC Data center provides the hardware and software facility, as well as the infrastructure for cloud computing service providers. At the

 ISBN: 978-1-61208-216-5

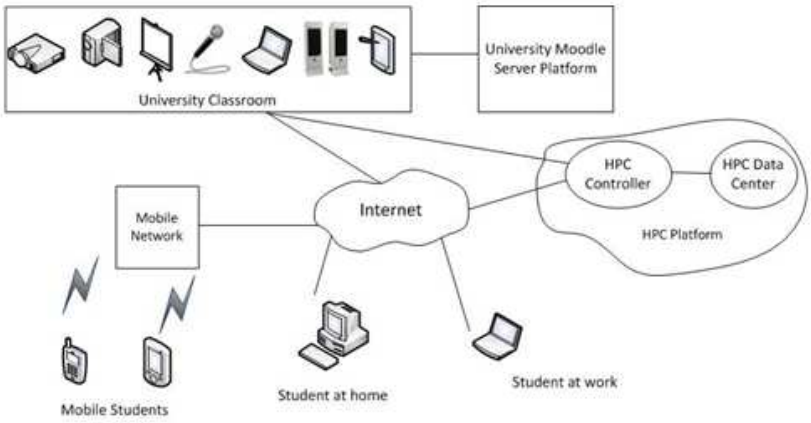


Figure 2. Mobile Cloud Computing (MCC) System Architecture with High Performance Computing (HPC) for Mobile Distance Learning (MDL)

HPC Data centers, several servers are linked with high speed networks to provide services requested by users.

Particularly the overall theoretic performance of the HPC cluster in Macedonia is 9 TFlops, and achieved peak LINPACK performance is 7.776 TFlops, that is 86% efficiency. It consists of 84 computational blade servers with 2 Six core L5640 CPUs and 24 GB RAM. The 6 management servers have also 2 Six core L5640 CPUs and 24 GB RAM, four of which act as storage servers and are connected in a failover configuration to a Serial Attached Small Computer System Interface (SCSI) storage with 60x600 GB Dual channel Serial Attached SCSI (SAS) disks. The HPC cluster provides possibility for deployment of any needed library, or software pack for any research community.

One potential application that delivers the information (learning content) from the course lecturer to the distance student and vice versa with a very high presenting quality is the Cisco WebEx application. WebEx suite, compared to other tools, offers a broad range of Web conferencing, and content sharing [25]. No software download is required for participants, and WebEx will run on any Internet server, or mobile devices such as smart phones, or tablets. A summary of WebEx Key features is given in [26].

WebEx can be used for different educational scenarios. For example the WebEx Whiteboard is a suitable tool for teachers in distance learning sessions. Also there is a possibility of annotations of the browser's application while sharing a map. WebEx is also a suitable tool for sharing and highlighting medical images in Telemedicine. WebEx offers possibility for sharing a presentation, where either can be used the WebEx annotation tools, or better the Power point annotation tools that are available in the presenter mode.

V. QUALITY OF EXPERIENCE (QOE) ASPECTS OF MDL IN MCC ENVIRONMENT

Mobile Distance Learning in a Mobile Cloud Computing Environment can be tested on both QoS and QoE aspects. Below is provided a short description for each of these aspects.

QoS refers to the technical aspects. It is defined as the ability of the network to provide a service at an assured service level. QoS encompasses all functions, mechanisms and procedures in the network and the terminal that ensure the provision of the negotiated service quality between the User Equipment (UE) and the Core Network (CN). QoS is measured, expressed and understood in terms of networks and network elements, which usually has little meaning to a user. The reliability in service concerns throughput, delay, jitter and loss in data during transmission of data; service availability, security in terms of authentication as well as authorization, coverage area, and service setup time of the related bearer service; service retain ability, in general characterizes connection losses [27].

QoE refers to the perception of the user about the quality of a particular service, or network, i.e. it depends on customer satisfaction in terms of usability, accessibility, retain ability and integrity of the service. QoE means overall acceptability of an application, or service, as perceived subjectively by the end-user. Quality of Experience includes the complete end-to-end system effects (client, terminal, network, services infrastructure, multimedia learning content, etc.). Overall acceptability may be influenced by user expectations and context.

However, the overall QoE (user perception) is influenced by both technical performance of the network (QoS aspects) and the non-technical aspects of the service. QoE refers to the personal feelings of the customer about the quality of a service, and it expresses using perceptive words like 'good', 'excellent', 'poor' [28].

Since High Speed reliable and secured internet access is used at the University Campus Network it can be assumed that the network has excellent technical performances, .i.e. no QoS technical issues are present. Therefore the main focus in this paper is directed towards the non-technical aspects of QoE evaluation of the mobile distance learning system in MCC environment, and its comparison to the conventional distance learning system in CC environment.

The QoE will be evaluated through answering the survey questions by the participants after the completion of the distance learning course. The survey consists of the following questions:

- What is the user's satisfaction in using the system from quality of presentation of learning documents?
- Is it easy to understand the presented concept?
- Did the user focus very easy to the presentation?

 ISBN: 978-1-61208-216-5

- How interactive is the system for communication with the presenter, asking the questions, etc.?
- Did the user find the Human Computer Interaction (HCI) friendly for himself/herself?
- How available is the learning system to the user?
- Did you find the usage of High Performance Computing (HPC) Center useful?

VI. QoE Evaluation Scenarios

In order to provide the QoE evaluation results (section VII), the system has been tested in the following two environments: the Distance Learning (DL) system in the conventional Cloud Computing (CC) environment and MDL system in MCC environment. These two scenarios are described in subsections A and B.

A. QoE Evaluation of DL System in a Convetional CC Envrionment

The test of WebEx Communication System for the DL system in the Conventional CC Environment was performed in the following two distance learning conference scenarios: Local Conference and International Conference. This corresponds to the scenario Student at Home, Student at the Office, or Student in a Distant Classroom, described in Fig. 2. The test was performed in the following two conference scenarios: the local scenario and the international scenario.

The local distance learning conference was performed locally within the University in Ohrid, in order to verify whether the WebEx can be used for distance learning, as well as to discover the possibilities and features of WebEx.

The international distance learning conference was performed between the University for Information Science and Technology from Ohrid, Macedonia and the Norwich University from Vermont, USA. A screenshot from this test is given on Fig. 3. It can be noticed that during this test there was a course presentation about Network Security prepared by the University of Norwich. Students were able easily to follow the presentation, to ask questions, or to exchange some ideas using the WebEx features. Several professors and students participated from both Universities.

Both scenarios were several times performed and were successfully completed.

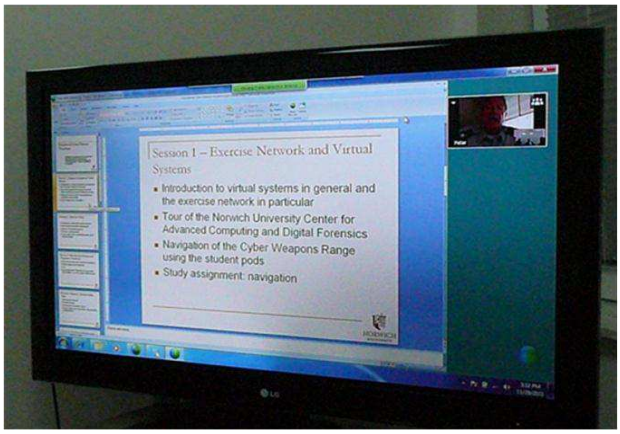


Figure 5. A Screenshot from the testing of WebEx on International Distance Learning Conference

B. QoE Evaluation of MDL System in MCC Environment

After the successful tests in subsection A, the WebEx Application was tested in a MDL System. This scenario corresponds to the mobile students' category (a situation when the students are on the road), described in Fig. 2. In order to perform the tests of this scenario one user used the following mobile devices: Motorola Milestone, HTC Sensation, iPhone 4 and iPad. The tests were successfully performed locally within the University, or regionally between the cities Ohrid and Skopje, at a distance of around 175 km. Screenshots from these tests are given on Fig. 4, 5, and 6.

Figure 6. HTC Sensation as a Part of the Learning System

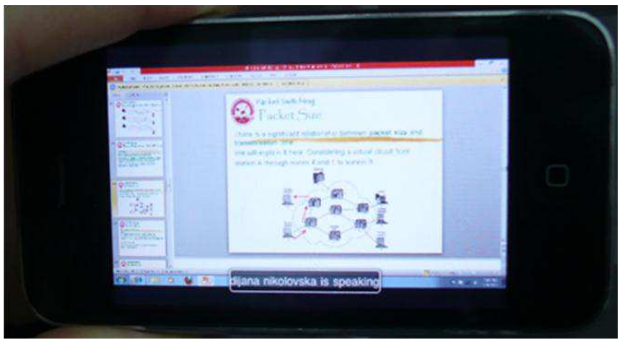


Figure 3. iPhone 4 as a Part of the Learning System

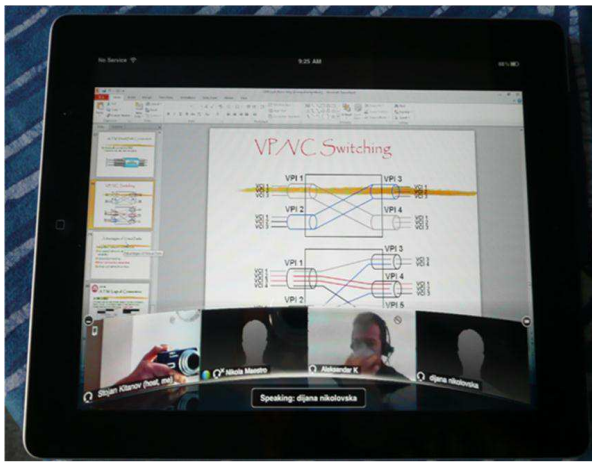


Figure 4. iPad as a Part of the Learning System

VII. COMPARISON OF QOE EVALUATION RESULTS

The survey questions were answered by 30 students that participated in the distance learning sessions of both CC and MCC environment. They answered the questions after their participation in the distance learning course. For simplicity we made the answers to have two options: 'good' or 'bad', i.e. 'yes', or 'no'. Each student's vote for each question has a weight of 10/3 by 30 participants. A summary of the QoE evaluation results is given in Fig. 7. The following things can be concluded.

The mobile devices provide higher and easier availability of the MDL system in MCC environment, since the conventional DL system in CC environment cannot provide the learning content for the mobile students. Additionally the usage of High Performance Computing (HPC) Center is more useful for MDL system in MCC environment, rather than DL in CC environment. HPC Cluster infrastructure is useful in situations where the necessary computing cannot be performed by a mobile device, or a conventional PC, or laptop. This MCC model can be applied everywhere where there is need of fast and intensive computing and analysis of huge amount of data, such as modeling of 3D graphics visualization and animation in ecology, global climate solutions, financial risks, healthcare and medical learning, decoding genome projects, etc.

Additionally, the following was concluded. For the DL system in CC environment was noticed a perfect communication, without any delay, or noise interference, since a high speed secured reliable internet access was used.

For the MDL system in MCC environment the network may not have good performances if the user uses the network on a high speed train. Additionally, the mobile devices have limited capabilities compared to conventional Laptop, or PC. Laptop, or PC can provide audio and video conversation, chat, and data sharing option. The tablet (iPad) supports audio and video conversation, and chat. The mobile phone supports audio conversation and chat. Currently data sharing (content sharing) from the mobile devices is not supported. These constraints are due to the capabilities of the mobile devices as well as the features that are supported by the current WebEx version. However, the smart phone and the tablet (iPad) can only view the data (content) that is shared from a PC or Laptop. This is sufficient for mobile students (students on the road) to listen, to view and follow the lecture, since it not expected from them to make any presentation.

VIII. Conclusion and Future Work

This paper provided a new Model of Distance Learning System in Mobile Cloud Computing environment, by using High Performance Computing (HPC) Cluster infrastructure, as well as some existing videoconferencing technologies enriched with mobile and wireless devices. After the introduction and the related work, the new MCC model was presented. Then new system architecture was proposed for the Mobile Distance Learning System in Mobile Cloud Computing Environment that uses the Internet Access. Then some QoE aspects of such distance learning system were addressed. Finally QoE evaluation was performed by comparing the MDL system in MCC environment with respect to the DL system in CC environment. It was concluded that mobile devices provide higher and easier availability of MDL system in MCC environment, since the conventional DL system with CC environment cannot provide the learning content for the mobile students. The smart phone and the tablet (iPad) can only view the data (content) that is shared from a PC, or Laptop, which is sufficient for the mobile students (students on the road). They have to listen, to view and to follow the lecture, since it is not expected from them to make any presentation. Additionally, the usage of High Performance Computing

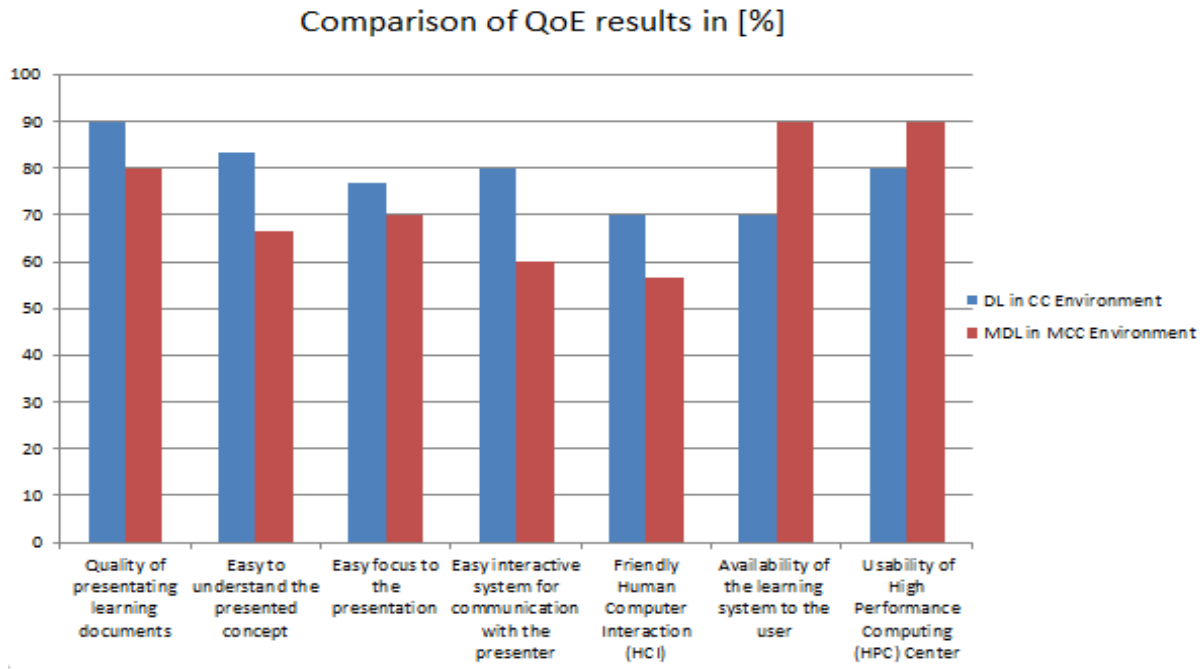


Figure 7. Comparison of QoE Evaluation Results in %

 ISBN: 978-1-61208-216-5

(HPC) Center is more useful for the MDL system in MCC environment, rather than for the DL system in CC environment. HPC Cluster Infrastructure is useful in situations where the necessary computing cannot be performed by a mobile device, or a conventional PC, or laptop. This MCC model can be applied everywhere where there is need of fast and intensive computing and analysis of huge amount of data, such as modeling of 3D graphics visualization and animation in ecology, global climate solutions, financial risks, healthcare and medical learning, decoding genome projects, etc.

In future, we plan to address some additional issues for the MCC, such as, Low Bandwidth, that could be solved with 4G (5G) and/or Femtocells, Network Access Management, QoS (from technical point of view such as network delay by using cloudlets, clonecloud, etc.), billing and standardization of the interface. However our main interest is to provide more services on a Software as a Service (SaaS) basis for mobile learners and/or more efficient MDL by using HPC center. We plan to include services based on simulation, or experiments performed by the HPC center on behalf of mobile users, particularly in healthcare and medical education and learning, where extremely is necessary to perform quick data analysis of 3D medical images.

ACKNOWLEDGMENT

Many special thanks to the respected Dr. Frank Vaneck and Dr. Phil Susmann from Norwich University, Vermont, USA for their participation in this international distance learning educational project. Also many special thanks to Aleksandar Karadimce, Dijana Nikolovska and Daniela Boshnakoska, and all UIST students for their support during this project.

REFERENCES

[1] T. H. Dihn, C. Lee, D. Niyato, and P. Wang, "A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches," Wireless Communications and Mobile Computing – Wiley, DOI:10.1002 WCM/1203, pp. 1 – 38, 11 October 2011. http://www.eecis.udel.edu/~cshen/859/papers/survey_MCC.pdf [retrieved: 05, 2012]

[2] N. M. Rao, C. Sasidhar, and V. S. Kumar, "Cloud Computing Through Mobile Learning," (IJACSA) Internationa Journal of Advanced Computer Science and Applications, Vol 1, No. 6, pp. 42 – 46, December 2010.

[3] High Performance Computing, Coraid solutions. http://www.coraid.com/solutions/high_performance_computing [retrieved: 05, 2012]

[4] Maryland CPU-GPU Cluster Infrastructure. http://www.umiacs.umd.edu/research/GPU/facilities.html [retrieved: 05, 2012]

[5] Moodle, Course Management System. www.moodle.org [retrieved: 05, 2012]

[6] Course Management Site of the University for Information Science and Technology, Ohrid, Macedonia. http://uistmoodle.servehttp.com/ [retrieved: 05, 2012]

[7] Cisco WebEx website. http://www.webex.com [retrieved: 05, 2012]

[8] S. S. Qureshi, T. Ahmad, K. Rafique, and S. U. Islam, "Mobile Computing as Future for Mobile Applications – Implementation Methods and Challenging issues," Proceedings of IEEE CCIS, pp. 467 – 471, November 2011.

[9] Mobile Cloud Computing Forum. http://www.mobilecloudcomputingforum.com/ [retrieved: 05, 2012]

[10] M. Tantow, "Cloud Computing and Smartphones," article in Cloud Times, 01 March 2011. http://cloudtimes.org/cloud-computing-and-smartphones/ [retrieved: 05, 2012]

[11] A. Rudenko, P. Reiher, G.J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," Journal of ACM SIGMOBILE on Mobile Computing and Communications Review, Vol. 2, No. 1, pp. 19 - 26, January 1998.

[12] U. Kremer, J. Hicks, and J. Rehg, "A Compilation Framework for Power and Energy Management on Mobile Computers," in Proceedings of the 14th International Conference on Languages and Compilers for Parallel Computing, pp. 115 – 131, August, 2001.

[13] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making Smartphones Last Longer with Code offload," in Proceedings of the 8th International Conference on Mobile Systems, applications, and services, pp. 49-62, June 2010.

[14] http://aws.amazon.com/s3/ [retrieved: 05, 2012]

[15] http://www.flickr.com/ [retrieved: 05, 2012]

[16] http://www.shozu.com/portal/index.do [retrieved: 05, 2012]

[17] http://www.facebook.com [retrieved: 05, 2012]

[18] P. Zou, C. Wang, Z. Liu, and D. Bao, "Phosphor: A Cloud Based DRM Scheme with Sim Card," in Proceedings of the 12th International Asia-Pacific on Web Conference (APWEB), pp. 459, June 2010.

[19] J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian, "Virtualized in-cloud Security Services for Mobile Devices," in Proceedings of the 1st Workshop on Virtualization in Mobile Computing (MobiVirt), pp. 31 – 35, June 2008.

[20] X. Chen, J. Liu, J. Han, and H. Xu, "Primary Exploration of Mobile Learning Mode under a Cloud Computing Environment," in Proceedings of the International Conference on E-Health Networking, Digital Ecosystems and Technologies (EDT), vol. 2, pp. 484 – 487, June 2010.

[21] H. Gao and Y. Zhai, "System Design of Cloud Computing Based on Mobile Learning," in Proceedings of the 3rd International Symposium on Knowledge Acquisition and Modeling (KAM), pp. 293 – 242, November 2010.

[22] J. Li, "Study on the Development of Mobile Learning Promoted by Cloud Computing," in Proceedings of the 2nd International Conference on Information Engineering and Computer Science (ICIECS), pp. 1, December 2010.

[23] Ceo, "On (Mobile) Cloud Computing – Multiple Perspectives to its Benefits, Drivers, and Economics," 25 December 2009. http://weblog.cenriqueortiz.com/mobility/2009/12/25/on-mobile-cloud-computing-angles-to-benefits-drivers-and-economics/ [retrieved: 05, 2012]

[24] Microsoft Server and Cloud Platform (Windows HPC Server 2008 R2). http://www.microsoft.com/en-us/server-cloud/windows-server/high-performance-computing-hpc.aspx [retrieved: 05, 2012]

[25] Socialbrite "Virtual Meeting Smackdown! 15 Top Web Conferencing Services Compared," Social Tools for Social Change, 19 January 2011. http://www.socialbrite.org/2011/01/19/comparison-top-web-conferencing-services/ [retrieved: 05, 2012]

[26] Cisco WebEx Key Features. http://www.webex.com/lp/keyfeatures [retrieved: 05, 2012]

[27] D. Sharma, R.K. Singh, "QoS and QoE Management in Wireless Communication System," International Journal of Engineering Science and Technology (IJEST), ISSN: 0975-5462, Vol. 3, No. 3, pp. 2385 – 2391, March 2011.

[28] N. Muhammad, D. Chiavelli, D. Soldani and M. Li, "QoS and QoE Management in UMTS Cellular Systems," John Wiley & Sons, Ltd. ISBN: 0-470-01639-6, pp. 1-8, 2006.

 ISBN: 978-1-61208-216-5

Provenance in the Cloud: *Why* and *How*?

Muhammad Imran
Research Group Entertainment Computing
University of Vienna, Austria
Email: imran.mm7@gmail.com

Helmut Hlavacs
Research Group Entertainment Computing
University of Vienna, Austria
helmut.hlavacs@univie.ac.at

***Abstract*—Provenance is an important aspect in the verification, audit trails, reproducibility, privacy and security, trust, and reliability in many fields ranging from art, food production, medical sciences, in-silico experiments, and distributed computing. On the other hand, Cloud computing is the business model of distributed computing and is considered the next generation of computing and storage platforms. Cloud computing requires an extension of the architecture of distributed and parallel systems by using virtualization techniques. Key to this extensible architecture is to support properties such as compute "on demand" and "pay as you go" model. Clouds are in use since a few years and they already expanded in the business domain (Amazon EC2, Microsoft Azure, IBM SmartCloud) and research environments (EUCALYPTUS, OpenNebula, Nimbus). Many research domains have already adopted Cloud technology into their existing computational and storage platforms and, thus, a shift of technology is in progress.**

In this paper, we present provenance description in computing sciences. Then, we give an overview of Cloud architecture and answer why provenance is important for Cloud computing. We introduce a mechanism to include provenance in the Cloud which requires minimal knowledge and understanding of underlying services and architecture. Therefore, we detail the importance along with the characteristics identified and present a framework for provenance in Cloud computing. We assure trust by augmenting a Cloud infrastructure with provenance collection in a structured way and present first performance results of the extended architecture. Finally, we discuss the results and summarize challenges and open issues of provenance in Clouds.

Keywords-provenance; research or open Clouds.

I. INTRODUCTION

Oxford dictionary [1] defines provenance as "the place of origin or earliest known history of something". In many fields including art, science and computing, provenance is considered as the first class data of importance for tracing an object to its origin. Provenance is defined by a set of different properties about the process, time, and input and manipulated data. Provenance is used to answer a few basic questions such as when the object was created, the purpose of creation, and where the object originated from (e.g., the creator of the object).

In computing sciences, a provenance system is used to collect, parse, and store related metadata. Such data is used for verification and tracking back, assurance of reproducibility, trust, and security, fault detection, and audit trials. These metadata include functional data required to trace back the creation process of objects and results, but also non-functional data such as the performance of each step including, e.g., energy consumption.

Since Cloud is an evolving technology which is based on virtualization and offer, on-demand computing, pay-as-you-go model, and is highly scalable and more abstract. There is a strong need to propose a provenance scheme for this dynamic, abstract and distributed environment. In addition to challenges for distributed computing, the abstraction and highly flexible usage pose new demands, i.e., a provenance framework for Clouds has to support these issues. Rajendra Bose et al. [2] present a detailed survey of computational models and provenance systems in distributed environment, specifically workflows execution. However, none of the approaches support provenance in the Cloud environment. These existing schemes rely on the support of native services from distributed or workflow computing, e.g., process schedulers. Generally, provenance systems in grid, workflow, and distributed computing are either strongly part of the enactment engine or they use Application Programming Interfaces (APIs), which are enactment engine specific [3].

Cloud infrastructure is not extensible by nature and therefore, existing techniques are not a good fit to Cloud environment and to address Cloud specific challenges. A better approach is to follow an independent and modular provenance scheme as described in [4]. Such a scheme is possible by extending the middleware of Cloud infrastructure where various components and services are deployed (extension of third party tools and libraries). This scheme which is a loosely coupled (domain and application independent) works independently of Cloud infrastructure, client tools and is of high importance to support future e-science.

In this paper, we provide a general discussion of provenance in different fields with a particular focus on open or research Clouds. We present underlying architecture of open Cloud, and propose a framework for provenance data collection in the Cloud. Hereby, we address the most important properties of a provenance system that is, independence of the Cloud architecture, low storage and computational overhead of provenance data, and usability. Provenance for Clouds to the best of our knowledge has not been fully addressed yet. The major contributions of this paper are

 ISBN: 978-1-61208-216-5

following:

- analysis of provenance in distributed computing, giving reasons of the importance and highlight challenges of provenance in the Clouds and distributed environment;
- a novel proposed scheme which can be deployed to the Cloud environment while addressing different vendors and architectures;
- first performance test results of the provenance framework.

The rest of the paper is organized as follows. In Section II, we discuss the related work in computing sciences. In Section III, Cloud architecture is discussed along with a presentation of Eucalyptus Cloud and its dependencies tools and applications. Section IV presents challenges and provenance data applied to Cloud computing. In Section V, we discuss the proposed framework, configuration of provenance system to Cloud middleware and its main components. Section VI describes first test results and Section VII concludes our work and details future implementation directions.

II. Related Work

Numerous techniques and projects have been proposed during the last few years for provenance systems in computational sciences for validation, reproduction, trust, audit trials and fault tolerance. These techniques range from tightly coupled provenance system to loosely coupled systems [5]–[8]. Provenance Aware Service Oriented Architecture (PASOA) [9], [10] uses Service Oriented Architecture (SOA) [11] for provenance collection and its usage in distributed computing for workflow management systems. myGrid [12] and Kepler [13] are examples of projects for executing in-silico experiments developed as workflows and they use Taverna [14] and Chimera [15] schemes respectively for Provenance data management in these computational systems. However, none of these approaches were designed specifically for Cloud computing architecture. Recently, Muniswamy-Reddy et al. [16] discussed the importance of provenance for Cloud computing services offered by AMAZON EC2 [17] using Provenance-Aware Storage Systems (PASS) [18] system.

In the e-science domain, experiments are performed in dry labs (in-silico); provenance system has to address data collection and availability in distributed environment. Provenance systems use different methods and approaches to address these challenges. Each approach has pros and cons which are related properties of a provenance system in distributed computing. Distributed computing challenges in general and Cloud specific challenges in particular are discussed in detail in Section IV-B, where Section IV-C gives a brief overview of Cloud specific provenance data.

III. Cloud architecture

Cloud vision is to address a complex engineering, medical or social problem by mega scale simulation and handling huge amount of data with a massive computation power. Clouds are generally categorized as business cloud, research or private cloud and hybrid cloud. IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service) are the terms heavily used in a Cloud computing paradigm and is mostly broken into these three segments.

IaaS: a service provided for the infrastructure (hardware and software) over the internet. Such an architecture provides servers, virtualized operating systems and data storage units. Elastic Cloud is a commonly used term for IaaS and users pay for required resources as they go. Amazon Elastic Compute Cloud (Amazon EC2), Nimbus [19], OpenNebula [20] and EUCALYPTUS [21] are the leading examples of IaaS. PaaS and SaaS are built on top of IaaS. PaaS provides an interface for software developers to build new or extend existing applications, e.g., Google App Engine and Microsoft Azure. SaaS is an application service provided to the end user by a vendor, e.g., google mail.

Private Cloud IaaS schemes are mostly used in a research environment and small businesses by using open source technologies. They are rapidly growing in the size and magnitude and expanding in different domains. With the new technologies and advancements, a private Cloud can be part of other public or private Clouds thus, providing the functionality of a hybrid Cloud.

A. EUCALYPTUS

Eucalyptus is an open source implementation of Cloud computing IaaS scheme using JAVA and C/C++ for various components. Users can control an entire Virtual Machine (VM) instance deployed on a physical or virtual resource [22]. It supports modularized approach and is compatible with industry standard in Cloud, i.e., Amazon EC2 and its storage service S3. It is one of the most used platforms to create scientific and hybrid Clouds. Eucalyptus gives researchers the opportunity to modify and instrument the software which is been lacking in the business offerings, e.g., Amazon EC2.

Figure 1 presents the extended architecture of Eucalyptus Cloud. There are three main components involved: Cloud Controller (CLC, i.e., middleware), Cluster Controller (CC) and Node Controller (NC). CLC, CC and NC communicates with each other and outside applications using Mule [23] and Apache Axis2/C framework. CLC interacts with CC, where CC is the part of Cloud used to manage clusters in the network. CC interacts and controls different NCs by associating and differentiating them using unique addresses and also balancing load in the cluster. NC assign a VM for the job execution submitted by a user. Walrus is web service used for distributed storage management of virtual images and users metadata. All the communications between different components of Eucalyptus Cloud is achieved by using SOAP, XML, WSDL, and HTTP communication protocols via Axis2/C and Mule framework.

 ISBN: 978-1-61208-216-5

IV. PROVENANCE IN CLOUD: *Why*

There are various definitions of Cloud computing (utility computing, autonomic computing) and is used as per the understandings, knowledge and requirements by different companies and users. Yes, there are some differences from previous computing technologies specifically to mention virtualization, on demand, pay as you go model, extremely flexible and more abstract. Ian Foster et al. [24] present an overview of the major differences between Cloud and grid and mentions the most important feature of Cloud technology is the total dependence on services (SOA architecture). There is underlying architecture for networking of software and hardware but, to the end user it is completely abstract and hidden. The abstraction allows the end user to send data to Cloud and get data back, without bothering about the underlying details. This behavior is fine for a normal user but, in research environment, scientists are more interested in the overall process of execution and a step by step information to keep a log of sub-data and sub-processes to make their experiments believable, trust able, reproducible and to get inside knowledge. With improvements of in-silico experiments, most of the computation and processing is done by using computing resources and not in a real lab.

Users of Cloud environment may not be interested in the physical resources, e.g., brand of computer but, surely they are interested in the invoked service, input and output parameters, time stamps of invocation and completion, overall time used by a process, methods invoked inside a service and the overall process from start to finish. This metadata which provides the user an ability to see a process from start to end or simply track back to find the origin of a final result is called provenance. Generally provenance is used in different domains by scientists and researchers to trust, track back, verify individual input and output parameters to services, sub process information, reproducibility, compare results and change preferences (parameters) for another simulation run. Provenance is still missing in Cloud environment and needs to be explored in detail as mentioned in [16], [25].

A. Implication of a Provenance Enabled Cloud

Introducing the provenance data into Cloud infrastructure would result in following advantages:

- Patterns: The use of provenance data to find patterns in the Cloud resources usage. These patterns can be further utilized to forecast a future request.
- Trust, reliability and data quality: The final data output can be verified based on the source data and transformation applied.
- Resources utilization: In Cloud, provenance data can be used to utilize the existing running resources by allocating copy of a running resource. This will be achieved by comparing a new request to the already running resources and this information is available in provenance data.

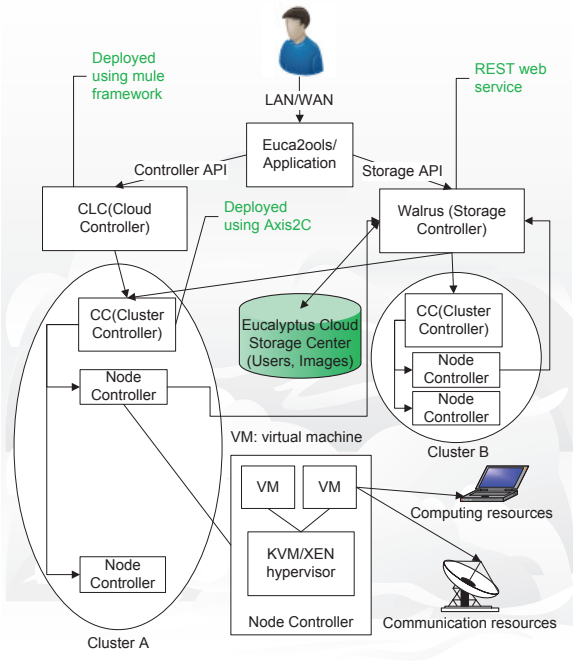


Figure 1. Extended architecture of Eucalyptus Cloud.

- Reduced cost and energy consumption: Provenance data results in a cost and energy efficiency by using patterns to forecast a future request and by utilizing existing running resources.
- Fault detection: Provenance data can pinpoint the exact time, service, method and related data in case of a fault.

B. Provenance Challenges in Cloud

Usual provenance challenges include: collecting provenance data in a seamless way with a modularized design and approach, with minimal overhead to object identification, provenance confidentiality and reliability, storing provenance data in a way so it can be used more efficiently (energy consumption) and presenting such information to the end user (query, visualization). Cloud brings more challenging to these existing challenges because we have to address the scalable, abstract and on demand architecture of Cloud. A provenance system in Cloud should address the following challenges:

- Domain, Platform and Application independence: How the provenance system works with different domain (scientific, business, database), platforms (windows, linux) and applications.
- Computation overhead: How much extra computation overhead is required for a provenance system in a particular domain.
- Storage overhead: How and where is the provenance data stored. It depends on the type, i.e., copy of original data or a link reference to original data, granularity

 ISBN: 978-1-61208-216-5

(coarse-grained or fine-grained) and storage unit (SQL server, mySQL, file system) of provenance data.
- Usability: It determines the ease of use of a provenance system from a user and Cloud resources provider perspective. How to activate, deactivate and embed a provenance system into existing Cloud infrastructure and services, e.g., is it completely independent or modification is required on Cloud services layer.
- Object identification: Identify an object in the Cloud and link the provenance data to source by keeping a reference or by making a copy of the source object.
- Automaticity: With huge amount of data and process computation within Cloud, collecting and storing provenance data should be automatic and consistent.
- Cloud architecture: Addressing the on-demand, abstract and scalable structure of Cloud environment with availability and extensibility of different components.
- Interaction with Cloud services: Cloud services are not extensible therefore, they cannot be modified. Business Clouds are propriety of organizations and open source Clouds needs understanding of every service if change is required. The better approach is to provide an independent provenance scheme which requires no change in the existing services architecture.

C. Provenance Data

A provenance system should address two different perspectives in collecting metadata for Cloud architecture. Applications running on Cloud as SaaS or PaaS and provenance of Cloud infrastructure (IaaS). Users of Cloud are more interested in their application provenance where, providers are interested in IaaS services provenance to observe resource usage and find patterns in applications submitted by users to provide with a more sophisticated model for resources usage. Following, is the list of mandatory metadata in a Cloud environment:

1) Cloud process data: Cloud code execution and control flow between different processes (web services), e.g., in EUCALYPTUS are CLC, CC and NC services. Web service and method name in particular.
2) Cloud data provenance: Data flow, input and output datasets which are consumed and produced and parameters passing between different services.
3) System provenance: System information or physical resources details, e.g., compiler version, operating system and the location of virtualized resources.
4) Timestamps: Invocation and completion time of Cloud services and methods.
5) Provider and user: Details about Cloud users and services provider, e.g., location of clusters and nodes. Different providers have different trust level and there could be laws against usage of resources for a particular geographical area.

V. PROVENANCE FRAMEWORK: *How*

A Cloud infrastructure is deployed and it relies on the open source third party tools, libraries and applications. Eucalyptus Cloud in particular depends on the Apache Axis, Axis2/C, and Mule framework. These third party libraries are used for the communication mechanism between various components of Cloud infrastructure. Cloud infrastructure is the orchestration of different services and the third party libraries works as a middleware to connect these services. The purpose of Cloud computing is more abstraction than previous technologies like Grid and Workflow computing and therefore, Cloud services are not extensible.

One method to implement provenance into the Cloud infrastructure is by changing the source code. This could be very cumbersome as deep understanding of the code is required. This will also restrict the change to the particular version of the Cloud. This method is not feasible to address the provenance challenge for various Cloud providers, domains and applications. The second method is to capture the provenance data on the middleware of a Cloud. This is possible by extending the third party libraries used by Cloud infrastructure and add custom methods to collect provenance data at various different levels. Such a scheme will lead to the minimum efforts and can be deployed across any Cloud scheme. Further, there will be no change required in Cloud services architecture or signature. To understand this techniques and hence the proposed provenance framework, we give a brief overview to the most important Mule and Axis2/C architecture.

A. Mule Enterprise Service Bus

Mule is a lightweight Enterprise Service Bus (ESB) written in JAVA and is based on Service Oriented Architecture (SOA). Mule enables the integration of different application regardless of the communication protocol used by those applications. Eucalyptus CLC services are deployed using Mule framework. CLC services are divided into different components including core, cloud, cluster manager, msgs, etc. These different components are built and deployed as jar files and they use Mule framework messaging protocols (HTTP, SOAP, XML, etc.) to communicate with each other and with other Eucalyptus services (NC and CC).

Extending Mule: Mule framework is based on layered architecture and modular design. Mule offers different kind of interceptors (EnvelopeInterceptor, TimeInterceptor and Interceptor) to intercept and edit the message flow. Since, provenance is metadata information flowing between different components (services) and we do not need to edit the message structure; therefore, we use EnvelopeInterceptor. Envelop interceptors carries the message and are executed before and after a service is invoked.

Configuring Mule Interceptor: There are two steps involved for configuring a Mule interceptors to Cloud services. First step is to built a provenance package (JAVA class files)

 ISBN: 978-1-61208-216-5

and copying to the Cloud services directory. Second step requires editing Mule configuration files used by different CLC components. Interceptors can be configured globally to a particular service or locally to a particular method of a service. Listing 1 is a sample "eucalyptus-userdata.xml" mule configuration file used to verify user credentials and groups.

Listing 1. Configuration of Provenance into Mule

```
<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns="http://www.mulesource.org/...">
  <interceptor-stack name="CLCProvenance">
    <custom-interceptor class="eucalyptus.CLC
                        provenance"/>
!.. indicating path of the package and class name
 for CLC services provenance data
  </interceptor-stack>
  <model name="eucalyptus-userdata">
    <service name="KeyPair">
      <inbound>
        <inbound-endpoint ref="KeyPairWS"/>
      </inbound>
      <component>
        <interceptor-stack ref="CLCProvenance"/>
!.. configuring "keypair service" to provenance
 module
        <class="com.eucalyptus.keys.KeyPair
                                Manager"/>
      </component>
      <outbound>
        <outbound-pass-through-router>
          <outbound-endpoint ref="ReplyQueue
                                Endpoint"/>
        </outbound-pass-through-router>
      </outbound>
    </service>
  </model>
</mule>
```

B. Axis2/C Architecture

Eucalyptus NC and CC services are exposed to other components by using Apache Axis2/C framework. Axis2/C is extensible by using handlers and modules [26]. Handlers are the smallest execution unit in Apache engine and are used for different purposes, e.g., web services addressing [27] and security [28]. A message flow between different components of CC and NC go through Axis2/C engine and we deploy custom handlers for provenance data collection inside Axis2/C. Similar concept is used in [29] for workflow services deployed in a tomcat container. This framework is not extensible to Cloud services and architecture. We differ from that work in many factors including interceptors for Mule, Apache Axis and Apache Axis2/C. There is no tomcat container available for Cloud services to deploy the provenance framework and Cloud services use HTTP, XML, SOAP and REST based protocols. Further, Our framework is developed for Cloud services provenance data collection and therefore, parsing, storing, and accessing provenance data is different than their architecture.

Configuration: Axis2/C modules and handlers can be configured globally to all services by editing axis2.xml file, or to a particular service and method by modifying servies.xml file. Listing 2 describes the configuration of provenance module to Eucalyptus NC service.

Listing 2. Configuration of Provenance into Axis2/C

```
<?xml version="1.0" encoding="UTF-8"?>
<service name = "EucalyptusNC">
  <module ref="NCprovenance"/>
!..this will configure provenance to all methods
 in NC
  <Operation name="ncRunInstance">
    <Parameter name = "wsmapping">
      EucalyptusNCncRunInstance
    </Parameter>
  </Operation>
  <Operation name="ncAttachVolume">
    <module ref="NCprovenance"/>
!..this will configure provenance to this
 particular method
    <Parameter name = "wsmapping">
      EucalyptusNCncAttachVolume
    </Parameter>
  </Operation>
</Service>
```

C. Framework Components

Proposed framework is divided into the following components to address the modularity and layered architecture:

- Provenance collection: Collecting important provenance data in a seamless and modular fashion using Mule, Apache Axis and Axis2/C interceptors.
- Provenance storage: Provenance data can be stored as part of Cloud storage unit or, to a dedicated database system. Properly indexing and linking provenance data to original data objects is compulsory.
- Provenance query and visualization: Providing an interface to query provenance data and visualize the results in a graph or chart form.
- Provenance usage: Using collected provenance data to enhance the trust on Cloud environments, reproducibility of applications and fault detection etc. Provenance usage is the extension of provenance query by providing with a standard output to make it compatible with other systems. Particular usage of provenance data is to find access pattern in resources usage, resources utilization (energy consumption) and faults detection.

Figure 2 describes the extended architecture of Axis2/C (particular version of Apache Axis used by EUCALYPTUS) and Mule, the main components of proposed framework and the deployment of provenance module.

VI. Testing and Evaluation

Test cases are performed on Mule and Axis2/C framework with provenance module for collection, parsing and logging metadata. Here, we present results for time increase with provenance module in Axis2/C for the execution chains called Inflow and Outflow. The underlying architecture and system details are following:

 ISBN: 978-1-61208-216-5

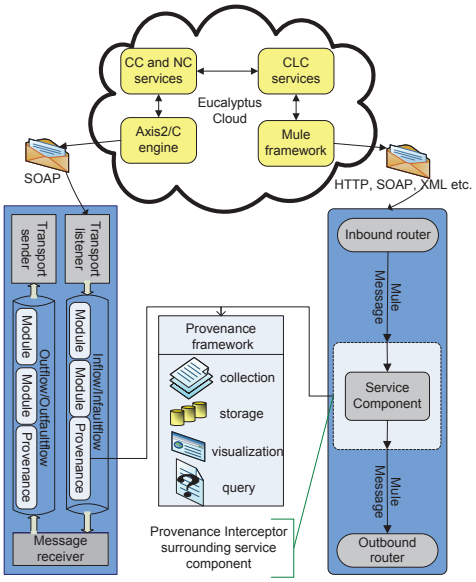


Figure 2. Framework components.

is essential. The increase in time is too less and negligible when considering the advantages like fault tracking, resource utilization, patterns finding and energy consumption of a provenance enabled Cloud. Furthermore, the successful deployment of provenance collection to Axis2/C and Mule frameworks support our theory of a generalized and independent provenance framework

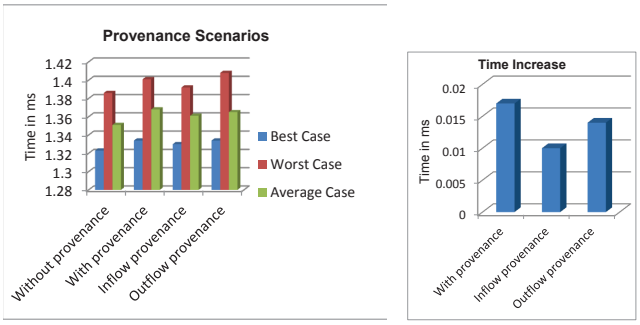


Figure 3. Test results of Echo service.

Operating system: Ubuntu 10.04, Processor: Intel Core 2 (2 GHz), RAM: 2 GB, Axis2/C version: 1.6.0, Web service: Echo

Echo service is invoked 100 times in a row for getting real data for comparison. Five multiple runs are performed for the calculation of best time, worst time and average time of execution. The process is executed by considering overall (Inflow and Outflow), only Inflow and only Outflow provenance. Apache Axis2/C engine is extended by using custom handlers and modules in the corresponding flows.

Figure 3 presents the performance of these different execution runs on Axis2/C engine. Left side of the figure details multiple runs of echo service without provenance, with provenance (Inflow and Outflow), only Inflow and only Outflow provenance. Y-axis represents the time required for execution. Right side of the figure shows the increase in time by comparison to without provenance. This increase in time is calculated for overall provenance, only Inflow provenance and only outflow provenance. The comparison is done for average values by using formula 1, where T_2 is time including provenance and T_1 is time excluding provenance.

$$Time\,increase = T_2 - T_1 \quad (1)$$

The average increase in time for 100 simulation runs of echo service for collecting and logging overall provenance data is only 0.017 ms when compared to the execution without provenance. The average increase in time for only Inflow provenance is 0.009 ms and for only Outflow provenance is 0.013 ms when compared to without provenance. The individual Inflow and Outflow provenance was collected for experimental purposes to observe the respective overhead. In a real lab experiment the overall provenance of process

VII. Conclusion and Future Work

Provenance is an important aspect in e-science. With the technology shift and changes, open Clouds are becoming an important part of e-science. Open Clouds are used in research and private business domain for storage, computation and execution of complex scientific applications. This paper considers provenance as an important metadata for Cloud environment and present provenance properties, Cloud architecture, open Clouds dependencies, and finally propose a framework. Proposed framework can be deployed to any Cloud scheme without modifying the basic services architecture or source code. Further, we gave a brief overview for the need of provenance in Cloud and present the major challenges and properties of such a framework. An independent system is proposed with advantages being simple, easy to use, easy to deploy, and works with open Cloud providers.

In future work, we will give insight details of the framework, simple user interface to configure provenance to Cloud service, evaluation of provenance framework for Cloud services and working example of provenance usage for fault detection and resources utilization (energy consumption).

References

[1] Oxford dictionaries. [retrieved: may, 2012]. [Online]. Available: http://oxforddictionaries.com/definition/provenance

[2] R. Bose and J. Frew, “Lineage retrieval for scientific data processing: a survey,” *ACM Comput. Surv.*, vol. 37, no. 1, pp. 1–28, Mar. 2005.

[3] Y. L. Simmhan, B. Plale, and D. Gannon, “A Survey of Data Provenance Techniques,” Computer Science Department, Indiana University, Bloomington IN, Tech. Rep., 2005.

[4] A. Marinho, L. Murta, C. Werner, V. Braganholo, S. M. S. d. Cruz, E. Ogasawara, and M. Mattoso, "Provmanager: a provenance management system for scientific workflows," *Concurrency and Computation: Practice and Experience*, 2011.

[5] M. Szomszor and L. Moreau, "Recording and reasoning over data provenance in web and grid services." in *CoopIS/DOA/ODBASE*, ser. Lecture Notes in Computer Science, R. Meersman, Z. Tari, and D. C. Schmidt, Eds., vol. 2888. Springer, 2003, pp. 603–620.

[6] Y. Cui and J. Widom, "Lineage tracing for general data warehouse transformations," in *Proceedings of the 27th International Conference on Very Large Data Bases*, ser. VLDB '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 471–480.

[7] P. Buneman, S. Khanna, and W. chiew Tan, "Why and where: A characterization of data provenance," in *ICDT '01: Proceedings of the 8th International Conference on Database Theory*. Springer, 2001, pp. 316–330.

[8] M. Imran and K. A. Hummel, "On using provenance data to increase the reliability of ubiquitous computing environments," in *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, ser. iiWAS '08. New York, NY, USA: ACM, 2008, pp. 547–550.

[9] S. Miles, P. Groth, M. Branco, and L. Moreau, "The requirements of recording and using provenance in e-Science experiments," University of Southampton, Tech. Rep., 2005.

[10] pasoa. [retrieved: may, 2012]. [Online]. Available: http://www.pasoa.org/

[11] oasis. [retrieved: may, 2012]. [Online]. Available: http://www.oasis-open.org/

[12] mygrid project. [retrieved: may, 2012]. [Online]. Available: http://www.mygrid.org.uk/

[13] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, "Kepler: an extensible system for design and execution of scientific workflows," in *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, Jun. 2004, pp. 423–424.

[14] Taverna workflow management system. [retrieved: may, 2012]. [Online]. Available: http://www.taverna.org.uk/

[15] I. Altintas, O. Barney, and E. Jaeger-frank, "Provenance collection support in the kepler scientific workflow system," in *In Proceedings of the International Provenance and Annotation Workshop (IPAW)*. Springer-Verlag, 2006, pp. 118–132.

[16] K.-K. Muniswamy-Reddy, P. Macko, and M. Seltzer, "Making a cloud provenance-aware," in *First workshop on on Theory and practice of provenance*, ser. TAPP'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 12:1–12:10.

[17] Amazon elastic compute cloud. [retrieved: may, 2012]. [Online]. Available: http://aws.amazon.com/ec2/

[18] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. I. Seltzer, "Provenance-aware storage systems." in *USENIX Annual Technical Conference, General Track*. USENIX, 2006, pp. 43–56.

[19] Nimbus. [retrieved: may, 2012]. [Online]. Available: http://www.nimbusproject.org/

[20] Opennebula. [retrieved: may, 2012]. [Online]. Available: http://opennebula.org/

[21] Eucalyptus. [retrieved: may, 2012]. [Online]. Available: http://open.eucalyptus.com/

[22] S. Wardley, E. Goyer, and N. Barcet, "Ubuntu enterprise cloud architecture," *Technical White Paper*, Aug. 2009.

[23] Mule esb. [retrieved: may, 2012]. [Online]. Available: http://www.mulesoft.org/what-mule-esb

[24] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," in *2008 Grid Computing Environments Workshop*. IEEE, Nov. 2008, pp. 1–10.

[25] M. A. Sakka, B. Defude, and J. Tellez, "Document provenance in the cloud: constraints and challenges," in *Proceedings of the 16th EUNICE/IFIP WG 6.6 conference on Networked services and applications: engineering, control and management*, ser. EUNICE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 107–117.

[26] A. S. Foundation, "Apache axis2/java - next generation web services," Website http://ws.apache.org/axis2/, 2009.

[27] Axis2- ws-addressing implementation. [retrieved: may, 2012]. [Online]. Available: http://axis.apache.org/axis2/java/core /modules/addressing/index.html

[28] Apache axis2/c manual. [retrieved: may, 2012]. [Online]. Available: http://axis.apache.org/axis2/c/rampart /docs/rampartc_ manual.html

[29] F. A. Khan, S. Hussain, I. Janciak, and P. Brezany, "Enactment engine independent provenance recording for e-science infrastructures." in *Proceedings of the Fourth IEEE International Conference on Research Challenges in Information Science RCIS'10*, 2010, pp. 619–630.

 ISBN: 978-1-61208-216-5

A Secure Data Access Mechanism for Cloud Tenants

Chunming Rong
Department of Electronic Eng & Computer Science
University of Stavanger, 4036, Stavanger, Norway
Stavanger, Norway
Chunming.rong@uis.no

Hongbing Cheng
Department of Computer Science & Technology
Nanjing University
Nanjing, China
cheng.hongbing@uis.no

***Abstract*—As the future big data storage center for tenants, cloud computing has been a hot issue recently, it consists of many large datacenters which are usually geographically distributed and heterogeneous, secure data access from cloud computing platform is a big challenge for cloud tenants. In this paper, we present a secure data access mechanism based on identity-based encryption and biometric authentication for cloud tenants. We review briefly about identity-based encryption and biometric authentication firstly and then we proposed a data access mechanism for cloud tenants, the mechanism set double protection for confidential data of cloud tenants, encryption will make the tenants data secure against the peekers and biometric authentication will eliminate the maloperations over tenants data by root administrator in cloud service. We compared the proposed mechanism with other technology and schemes through comprehensive analysis and experiment data; the results show that the proposed data access mechanism is feasible and suitable for cloud tenants.**

Keywords—Cloud computing; Big data center; Data access; Data security.

I. INTRODUCTION

As the big data center for tenants, cloud computing [1] platforms have many particular types of datacenters, or most commonly, groups of datacenters. Cloud service providers not only offer applications including search, entertainment, email and other services that Internet can provide, but also they have expanded offerings to include compute-related capabilities such as virtual machines, storage, and complete operating system services for science computing and research. At the same time, cloud computing has been proven to be a hopeful application platform and paradigm to provide potential consumers with valuable information technology services over the Internet and these services should be efficient, secure and rapid. In order to meet the above services requirement, cloud computing resources should be rapidly deployed and easily scaled. In cloud computing all processes, applications and services supplied "on demand," no need to regard user's geographic location and computer devices.

Currently, many public and private cloud services are available for tenants. Generally, private cloud computing platforms are for special intention and will not offer servicefor others, but public cloud computing platforms are available to every one with Internet access. According to the type of service provided, public cloud platform include Software as a Service (SaaS) clouds like IBM LotusLive™ [2], Platform as a Service (PaaS) includes Google AppEngine [3], Infrastructure as a Service (IaaS) like the Amazon Web Services (AWS) [4] and famous Apache hadoop [5]. Hadoop includes some subprojects such as Mapreduce and hadoop distributed file system (HDFS) and has developed many open-source software's for reliable, scalable, distributed computing. Private clouds are owned and used by a single organization or department. They provide many of the same services as public clouds, and they give the owner organization greater flexibility and control. What is most important is that private clouds can provide lower latency than public clouds during rush time of Internet occupation. Considering the benefits of the two kinds of clouds, many organizations embrace both of them by integrating the two platforms into hybrid cloud computing models. These hybrid clouds are designed to meet some specific commercial, science and technology requirements, helping to optimize security and privacy for customers in minimum investment

Cloud storage is an excellent solution for tenants' big data, and it is a promising technology and the benefits of it are obvious, but, as a commercial platform, security is the most important. To develop proper security mechanisms for cloud implementations is a big challenge. Except for the usual challenges of developing secure information technology systems, cloud computing is under some special risk [6], because essential services are often performed by a third party that .is unknown to cloud computing platform or users. These "unknown" aspects of cloud outside environment make it harder to maintain data integrity and privacy. In fact, cloud computing always transfers much of the control over data and applied operations from the tenant organization to their cloud providers, in some extent, it is similar with that organizations entrust part of their information processing operations to outsourcing other companies or agent platforms. Even the basic tasks processing, such as applying data updating and configuring network protocols may become the responsibility of the cloud service providers, not the tenants. So, in this circumstance, tenants must establish trust relationships with cloud computing service providers and understand security risk in terms of how these service providers should

 ISBN: 978-1-61208-216-5

undertaketheir responsibility, deploy and manage security on their behalf. This kind of relationship between cloud service providers and tenants is critical because the tenants are obliged to be ultimately responsible for integrality and protection of their critical data and information, even if that tasks processing or programs have moved to the cloud computing platforms. In fact, it is the most difficult to determine the physical location where tenant data is stored inside the cloud computing environment. Security processes and issues that were once visible for tenants are now hidden behind fuzzy structure by cloud computing. This invisibility can arouse a number of security and compliance problems. On the other hand, the massive sharing of infrastructure with cloud computing creates an evident difference between cloud data security and other traditional platforms data security. Tenants who come from different organizations with different security anticipation and privilege often interact with the same set of cloud computing computation resources. On the other side, data-access security concern, cloud resource balancing, changing service-level agreements and other updating dynamic information technology environments will provide intentional-destroyer with more opportunities for misconfiguration. At the same time, data compromise and malicious conduct [7] by adversary, root users or administrator are the risk that the tenants must face. Data access calls for a high degree of standardized and strict operating rules, which can help improve data access security by eliminating the risk of supervisor operator error and intended maloperation. Therefore, the risks inherent with a massively shared infrastructure mean that cloud computing platforms and their secure data access have to pay more attention on identity and authentication.

The rest of the paper is organized as follows; in Section II, security concern on data access is described and in Section III, a secure data access for cloud tenants is proposed. In Section IV, we give a detail analysis and experiment results of the proposed mechanism. Conclusions are drawn in Section V.

II. SECURITY CONCERN ON DATA ACCESS

Generally, in terms of the service level agreements (SLAs) between tenants with Internet Service Providers (ISPs) or Cloud Service Providers (CSPs), we can categorize Internet or cloud services as below [8]:

- *Infrastructure as a Service (IaaS):* Under this kind of service model, ISPs allows tenants to use their database and some public services, at the same time, the tenants can rent computation, storage, networks, and other resources what they do not have to perform science research and commercial operations, such as Amazon and Hadoop. The tenant can directly deploy and run the guest OS and applications provided by ISPs. In general, the tenants have not the privilege to manage or control the underlying cloud infrastructure but have privilege tocontrol OS, storage, deployed applications, and networking components configuration.
- *Platform as a Service (PaaS):* This service model can provide the tenants to deploy and run their tasks and application program onto the platform infrastructure. For example, IBM also provides the tenants this kind of cloud service platform to build their programs based on some popular programming languages and software tools. The tenants can not manage or control the underlying cloud system when they perform their tasks on the platform.
- *Software as a Service (SaaS):* It is a common model that has been adopted by most of ISPs. In this mode, the tenants are passive and only can use what the providers provide. Such as websites browsing, email, and others, service providers undertake all of the responsibilities and develop attractive software and services for the tenants, the tenants make use of these services under some risk because the service providers maybe leak their information or critical data kept on the servers of the infrastructure The advantage of using this kind of service is that there is no upfront investment in servers or software licensing.

Cloud tenants do not want others to access or fetch their confidential data stored in cloud storage [9], so secure data access control is even more critical for data integrity and privacy. On the other hand, in general, there are two critical roles in clouds computing service called privileged users and the third-party system, privileged users refer to root users or administrators who working for the cloud providers. Privileged-users perform physical monitoring, resource scheduling, background checking. Privileged-users must have the capabilities to coordinate authentication and authorization with the tenants and enterprise back-end or third-party systems. The third-party system is a partner of the cloud service providers, it cooperates with cloud provider to easily and quickly leverage cloud services for end users.

It is evident that most famous organizations, enterprises and even general tenants cite data protection as their most important security consideration when using cloud computing service. Typical security concerns [10] include the way how data is stored, accessed and released. Tenant sensitive or regulated data needs to be properly segregated and kept on the cloud storage infrastructure, including important archived data. Finding a suitable way of encrypting and managing encryption keys of data in transit to the cloud platforms or the service provider's data center are critical to protect data privacy ,integrity and usability. The encryption of data and the ability to securely share those encryption keys between the cloud service provider and consumer is an important way that ensures security of data access. On the other hand, it is very expensive to transfer large volumes of data quickly over the Internet, so,

 ISBN: 978-1-61208-216-5

it is very critical to protect the data security when transferring the data from tenants to cloud storage platform. When sending data to the cloud service providers, it is critical that the data is encrypted and that only the cloud service providers and tenants have access to the encryption keys. But when the cloud service providers stealthily violate the agreement and to obtain some information about the transferring data using the encryption keys. So, some significant restrictions regarding with secure data access must be established for both sides to comply with.

How to set the restrictions depends on the feature of the data and the importance of the data to tenants, such as commercial value, personal privacy et. Several member states[11] of the European Union (EU) have set rules to forbid the nonpublic personal information of its citizens to leave their state borders. So, in a full shared cloud computing environment, all parties of the cloud computing participators must agree on their responsibilities to secure data and perform these security policies on a regular basis. These parties must take the responsibilities to make a secure data access environment for each participator in the cloud computing.

III. THE PROPOSED DATA ACCESS MECHANISM

Firstly, we review the identity-based encryption and biometric authentication technology and then we show the proposed data access mechanism for cloud tenants.

3.1. PRELIMINARY

A. Identity-based Encryption

Adi Shamir proposed the concept of identity-based cryptography [12] in 1984 firstly. Shamir's original motivation for identity-based encryption was to simplify certificate management in e-mail systems. When Alice sends mail to Bob at bob@company.com she simply encrypts her message using the public key string "bob@company.com". There is no need for Alice to obtain Bob's public key certificate. When Bob receives the encrypted mail he contacts a third party, which we call the Private Key Generator (PKG). Bob authenticates himself to the PKG in the same way he would authenticate himself to a Center of Authentication (CA) and obtains his private key from the PKG. Bob can then read his e-mail. Note that unlike the existing secure e-mail infrastructure, Alice can send encrypted mail to Bob even if Bob has not yet setup his public key certificate. Also, note that key escrow is inherent in identity-based e-mail systems: the PKG knows Bob's private key.

The distinguishing characteristic of identity-based encryption is the ability to use any string as a public key. The functions that compose a generic IBE can be specified as follows.

In 2001, Boneh and Franklin proposed a practical algorithm[13] firstly, based on IBE technique. To describe the Boneh and Franklin IBE algorithm, from here on, we use Z_q to denote the group $\{0,\cdots q-1\}$ under addition modulo q. For a group G of prime order we use G^* to denote the set $G^* = G \mid O$ where O is the identity element in the group G. We use Z^+ to denote the set of positive integers We give first some definitions and then the basic IBE scheme.

Definition 2.1 A map $\hat{e}: G_1 \times G_1 \to G_2$ *is called a bilinear pairing if, for all* $x, y \in G_1$ *and all* $a, b \in Z$ *, we have* $\hat{e}(x^a, y^b) = \hat{e}(x, y)^{ab}$.

Definition 2.2 The Bilinear-Diffie-Hellman problem (BDH) for a bilinear map $\hat{e}: G_1 \times G_1 \to G_2$ *such that* $|G_1| = |G_2| = q$ *is prime is defined as follows: given* $g, g^a, g^b, g^c \in G_1$, *compute* $\hat{e}(g,g)^{abc}$, *where* g *is a generator and* $a, b, c \in Z$ *. An algorithm A is said to solve the BDH problem with advantage* ε *if*

$$\Pr[A(g, g^a, g^b, g^c) = \hat{e}(g,g)^{abc}] \geq \varepsilon$$

where the probability is over the random choice of $a, b, c, g,$ *and the random bits of A*

Definition 2.3 A randomized algorithm G that takes as input a security parameter $k \in Z^+$ *is a BDH parameter generator if it turns in time polynomial in k and outputs the description of two groups* G_1, G_2 *and a bilinear function* $\hat{e}: G_1 \times G_1 \to G_2$, *with* $|G_1| = |G_2| = q$ *for some prime* q. *Denote the output of the algorithm by* $G(1^k) = < G_1, G_2, \hat{e}, q >$.

Definition 2.4. We say that G satisfies the BDH assumption if no probabilistic polynomial algorithm A can solve BDH with non-negligible advantage.

The detail on the basic identity-based encryption algorithm can obtain in [13]

B. Biometric Recognition

Biometric recognition is a process of automatically recognizing the identity of a person based upon one or more intrinsic physiological or behavioral traits that the person possesses. Physiological characteristics are related to the shape of the body and the widely deployed ones include fingerprint, face, iris and hand geometry[14]. Behavioral are related to the behavior of a person and voice and gait are among the mostly researched.

From the viewpoint of pattern recognition, biometric recognition is a typical classification problem, which generally includes two main modules: feature extraction and classification. Through feature extraction, discrimitive and compact digital representation of biometric sample is

 ISBN: 978-1-61208-216-5

generated. In classification, statistical techniques are generally applied to learn biometric pattern for each person during training, and make decision on identity during test by using the learned patterns.

A biometric recognition system can operate in two modes: verification and identification. Verification (orauthentication) accepts or rejects the identity claim of a person (for example, Bob). Identification determines which of the registered persons a given biometric data comes from. The idea can be described as follows, when any person say, q, want to use authentication system, first, he must get a legal ID from system and pass the system check by sys_checker. Then, uses the ID to create his biometric template, all of the created templates are storied in system database such as sys_database. In verification phase, q's template is sent to the system matcher to match with the extracted biometric feature from q. Otherwise, in identification phase, the q's extracted biometric feature will have a match with all the storied templates in the system database. Algorithm 1 describes the process of biometric authentication.

Algorithm 1. Biometric authentication process for the Person q

```
Bio_Au_process(Person q){
    sys_checker ←q.ID
   if (sys_checker){
    for(i=0;  ;i++)
    {Template[i]←Extract Biologic feature of q
          sys_database ←Template[i]}
                        }
   //create personal biologic feature template
 if someone p claims that he or she is q
     {p.tmp←Extract Biologic feature of  p
          matcher ←p.tmp
          matcher ←Template[i]
   //send p's biologic feature template to matcher
          If matcher (p.tmp == Template[i])
               p is q
                        }
 else  {p.tmp←Extract Biologic feature of  p
          matcher ←p.tmp
          matcher ←Template
   //send all biologic feature templates to matcher
          for(j=0; ;j++)
          {If matcher (p.tmp == Template[j])
               p has passed authentication
                        }
                          }
          return
                        }
```

Biometric authentication is a statistical hypothesis testing problem involving in a tradeoff between two error types: false reject and false alarm. The performance measures of such system are the false reject rate (FRR) and the false alarm rate (FAR) which can be adjusted by an acceptance threshold. FRR is the proportion of genuine users that are incorrectly rejected. FAR is the proportion of impostors that are incorrectly accepted as genuine users.

The performance of identification is measured asidentification rate which is significantly influenced by population size among other things. For face recognition, it is found that identification rate decreases linearly in the logarithm of the population size.

Being easy-to-use and non-intrusive, biometric recognition technology is widely deployed to control access to restricted services, for example, banking and databases. In the initial phase, users are required to enroll in a system, namely, to give examples of their biometric data to the system so that it can build models for them and this should be done only once. This is similar to the sign up procedure to establish ID and password. In the verification phase, the identity claim is accepted or rejected; or in identification phase, the identity is determined. Each time when a user accesses to the service, verification or identification is performed.

Design of a biometric system needs to take into consideration such factors as the available sensors, the performance of various biometric recognition technology, existing security infrastructure, and cost and user acceptance.

With the recent advance in biometric recognition techniques and low-cost sensors, we can expect the increasing deployment of biometric recognition in many fields including cloud computing.

3.2. The Proposed Data Access Mechanism

We design a secure data access mechanism for cloud tenants based on Boneh-Franklin IBE algorithm and biometric authentication, the detailed mechanism is as follows.

Step1: Setup cloud side parameters

1. initialization

On the cloud service side, given a security parameter $k \in Z^+$, the algorithm works as follows:

Run G on input k to generate a prime q, two groups G_1, G_2 of order q, and an admissible bilinear map $\hat{e}: G_1 \times G_1 \to G_2$ .Choose a random $\alpha \in G_1$ .Pick a random $s \in Z_q^*$ and set $\beta = \alpha^s$. Choose cryptographic hash functions for some n , $H_1: \{0,1\}^* \to G_1^*$, $H_2: G_2 \to \{0,1\}^n$, $H_3: \{0,1\}^n \times \{0,1\}^n \to Z_q^*$, $H_4: \{0,1\}^n \to \{0,1\}^n$ .For the security proof, we view all the hash functions as random

oracles. The message space is $M = \{0,1\}^n$. The ciphertext space is $C = G_1^* \times \{0,1\}^n$. The output system parameters are $\pi = \{q, G_1, G_2, \hat{e}, n, \alpha, \beta, H_1, H_2, H_3, H_4\}$. The master key is $s \in Z_q^*$. Where q is a prime number, G_1 and G_2 are two groups of order q, $\hat{e}: G_1 \times G_1 \to G_2$ is a bilinear map, n is the length of plaintext, $\alpha \in G_1$, $\beta = \alpha^s$, $s \in Z_q^*$ is the master key, H_1, H_2, H_3, and H_4 are four hash functions with random oracles respectively. The master key should be kept in a secret place and the parameters can be distributed to all nodes.

2. key generation

When tenants are registered in cloud computing providers, each tenant will obtain a unique identity to identify him or her. In our proposed mechanism, the obtained identity is same with the one used in IBE algorithm. For a given tenant identity $ID \in \{0,1\}^*$ (ID is the cloud tenant's public key. It could be a random string and so it is very convenient and easily realized). According to IBE algorithm, the private key of the tenant can be calculated as following:

Compute $Q_{ID} = H_1(ID) \in G_1^*$.Set the private key of the tenant K_{ID} to be $K_{ID} = (Q_{ID})^s$,where *s* is the master key.

The phase generates private key corresponding to given registered ID of every tenant in cloud computing.

Step2: Generate tenant's biometric template

Cloud computing is a pervasive service environment for tenants, different tenants have different security requirement. To these tenants who have special security concern on data can generate their biometric template and be stored in cloud database. Biometric authentication must be needed when someone wants to access the data. Modern mobile and video technology make the generation of tenant's biometric template very convenient and easy, many tenants can finish the process on the cloud interface through iphone and other mobile devices. The process of generation tenant's biometric template is described in part of Algorithm 2.

Step3: Encrypt cloud data

Input: cloud data (which is created by cloud tenants and stored in the database of cloud platform), a private key (the cloud service providers), and an ID (the cloud tenant who want to access the data); output: encrypted cloud data. The detailed operation is as following.

Input: A cloud data message $m \in \{0,1\}^*$, a private key d_A, an identity ID_B, and the system parameters. Choose a random $\mu \xleftarrow{R} \{0,1\}^n$, compute $r = H_3(\mu, m)$ and $s := e(d_A, H_2(B))$ then output the encrypted cloud data ciphertext $c := < r, \mu \oplus H_1(r,s), E_{H_4(\mu)}(m) >$.

Step4: Biometric authentication

As an excellent storage scheme for tenants' big data, cloud computing has been a hot issue for a lot of consumers, generally, tenants' different data should be processed by different security modes. Biometric authentication has the advantage of exclusive for tenant in data access. When any registered cloud tenant say, p, want to access the data stored in cloud, first, he must pass the cloud system check such as cloud_sys_checker. Then, cloud tenants use registered identity ID to create their biometric template and all of the created templates are stored in cloud_sys_database. In cloud data access, cloud tenant p must pass the biometric authentication performed by biometric matcher in cloud computing. Part of algorithm 2 describes the process.

Algorithm 2. Biometric authentication for tenant p to access cloud data

```
Cloud_Bio_Au(Person p){
 //generation of cloud tenant p biometric template
    Cloud_sys_checker ←p.ID
   if (Cloud_sys_checker){
   for(i=0;  ;i++)
 {Template[i]←Extract Biologic feature of tenant p
            Cloud_sys_database ←Template[i]}
                        }
 //biometric authentication for cloud data access
 If cloud tenant p want to access cloud data
        {  p.tmp←Extract Biologic feature of  p
           matcher ←p.tmp
          matcher ←Template
           for(j=0; ;j++)
           {If matcher (p.tmp == Template[j])
              p has passed authentication
                        }
                          }
              return
                        }
```

Step5: Decrypt cloud data

Input: encrypted cloud data ciphertext (which is generated in Step3), an ID (the cloud service provider's), a private key (the cloud tenant who want to access the data), and output: the corresponding plaintext i.e. cloud data. The

 ISBN: 978-1-61208-216-5

detailed operation is as follows.

Input: An encrypted cloud data c, an identity ID_A, a private key d_B. Compute $s := e(H_2(A), d_B)$, $\mu := V \oplus H_1(U, s)$, $m := D_{H_4(\mu)}(m)$. Check whether $U = H_3(\mu, m)$ holds. If not, reject the ciphertext; otherwise output the plaintext *m* i.e. the cloud data that tenantaccess. Consistency is clear since $e(d_A, H_2(B)) = e(H_2(A), H_2(B))^a = e(H_2(A), d_B)$ by bilinearity.

IV. ANALYSIS AND COMPARISON

In this section, we mainly focus on analysis of feasibility and security of our mechanism. At the same time, we will compare our mechanism with other relational technology including cryptography and Role Based Access Control (RBAC) scheme.

1) Feasibility analysis and comparison

(1) Cloud computing will provide its legal tenants with pervasive communication service anytime and anywhere. Recent development of wireless communication technology has gained a rapid progress, many wireless standards and modes emerged, including 3G, Wi-Fi, et. At the same time, wireless communication devices also have made a quick development; some advanced wireless communication devices, such as iphone and iPod, equipped with many high-class functions that only possessed by lap-top class device before. All of these advances make access of Internet by wireless connection become more and more dominant. In many public places, more and more people rely on such mobile devices to browse web page, to download multimedia and to interact with Internet.

When applied our data access mechanism in cloud computing, latest communication technology can support the running of the proposed data access mechanism well. Users can operate mobile devices on touch screen and push technology. On the other side, the advanced wireless communication devices can be used as camera, can deal with massive multimedia data packet and run a lot of complicated software and program. All of these are fundamental for the proposed data access mechanism and it is possible for the mechanism to be applied in practice.

(2) In the proposed data access mechanism, biometric authentication is an important secure measure for cloud data. On general impression, biometric authentication is complicated and costly for common applications, it is only available in some crucial situations, such as bank counter, airport security, etc., but it is not true now, rapid progress of electronics technology make it realistic to produce cost-efficient and multifunctional mobile communication devices which can read and process tenant's some feature information such as fingerprint, face and iris, etc. It is reported that only the users of iphone in the world will exceed 100 million till 2012 [15]. Now, in some public places of many countries, these kind of advanced multimedia mobile devices are available for tenants free to use. Therefore, all of these prosperous situations make it feasible and convenient to apply the proposed data access mechanism in cloud computing environment.

2) Security analysis and comparison

(1) As we know, except for key leaking, the security of key not only is related with key length, but also depends on encryption algorithm. Symmetric encryption algorithm DES with 64-bit key (DES-64) has been cracked for about 20 years and RSA algorithm with 768-bit key (RSA-768) was cracked in 2009 by some scientists in Switzerland [16]. So, for the sake of making data access secure in cloud computing, we have to find suitable secure encryption algorithm and secure key length. Identity based encryption algorithms are based on Elliptical curve cryptography (ECC). Related research results [17] show that the traditional asymmetric RSA algorithm with 1024-bit key (RSA–1024) provides the currently accepted security level, in order to reach the same security level, ECC key length is 160-bit (IBE-160) and symmetric key length is 80 bits. On a PC with Redhat Linux 9.0, P42.8G processor and 512M DDR, we tested the average encryption and decryption time for different encryption algorithm, these time cost does not include keys distribution and parameters setup, the comparison is listed in Table 1.

TABLE 1 COMPARISON OF DIFFERENT ENCRYPTION ALGORITHM

algorithm	key (bit)	Average encryption cost(s)	Average decryption cost(s)	Has been cracked
RSA	1024	21.2261	34.4025	RSA-768
IDES	80	0.0028	0.0028	DES-64
IBE	160	0.1279	0.1279	NO

From the results in Table 1, we can conclude that the proposed data access mechanism is the safest for cloud tenants. Although symmetric encryption algorithm has some advantage in key bit and time cost, the fatal weakness is that encryption key and decryption key are same and kept by different parties, in addition, the RSA encryption and symmetric encryption had been cracked and the attempt for cracking more bits of them will continue.

(2) Biometric authentication technology has been developed for decades and many of them have been applied in some security scenarios successfully. In detection of criminals, biometric authentication such as finger-print and face recognition have made many pernicious cases come out in the wash. As the rapid development of social economy and technology, biometric authentication can be applied in more and more situations. Of all the biometric authentication technology, the face recognition is a convictive representative. In 1993, the American government launched a project called FERET [18] to found a series of technology development efforts and evaluation cycles, the face

 ISBN: 978-1-61208-216-5

recognition community benefited a lot from this project and built a large datasets collected to test face recognition technology, the large datasets push the research of technology forward quickly. Figure 1 show that from the beginning of the Facial Recognition Technology (FERET) program to the Face Recognition Vendor Test 2006 (FRVT 2006). The remarkable improvement of face recognition has five important milestones since 2003. To each representative algorithm, they wereevaluated on the false reject rate (FRR) at a false accept rate (FAR) of 0.001 (1 in 1000). The algorithm for 1993 was Turk and Pentland's eigenface algorithm [18] and for 1997 is Sept97 FERET evaluation [19]. The 2002 evaluation result is from the FRVT 2002 and the 2006 and 2010 is from the FRVT 2006 and FRVT 2010.

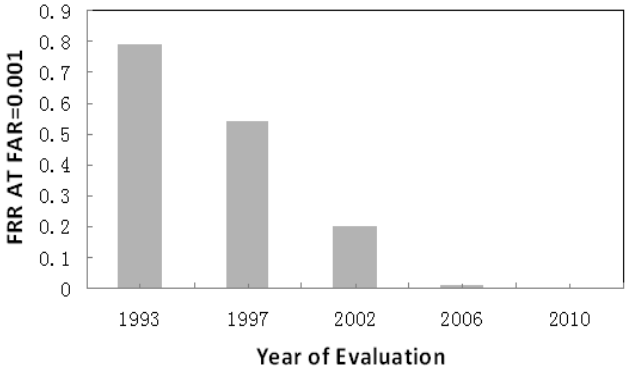


Figure 1 Improvement of face recognition from 1993 to 2010

From the evaluation results in Figure 1, we can conclude that as the representative of the biometric authentication technology, the face recognition attained an enormous improvement these year, especially, from 2002 to 2010, the improvement was very evident. The factors on the improvement due to advancement in algorithm design, advanced multimedia devices and more deep understanding of image processing. So, the low false reject rates (FRR) at a false accept rate (FAR) of the biometric authentication will enhance the security of the proposed data access mechanism.

V. CONCLUSIONS AND FUTURE WORK

Cloud computing has been a hot issue recently, as the future big data storage center for tenants, cloud computing is an Internet-based pervasive information infrastructure to provide tenants with data storage and service on demand it consists of many large datacenters which are usually geographically distributed and heterogeneous, secure data access from cloud computing platform is a big challenge for cloud tenants., how to design a secure data access mechanism for cloud computing is a main concern for service providers and their tenants.

In order to seek a secure data access method for cloud tenants, we presented a secure data access mechanism based on identity-based encryption and biometric authentication in this paper, the mechanism set double protection for confidential data of cloud tenants, encryption will make the tenants data secure against the peekers and biometric authentication will eliminate the maloperations over tenants data by root administrator in cloud service. We compared the proposed mechanism with other technology and schemes through comprehensive analysis and experiment data, the results show that the proposed data access mechanism is feasible and suitable for cloud tenants. In future work, We will make our proposed scheme more efficient and put it into practice.

ACKNOWLEDGMENT

This work was supported in part by the 973 Project under the Grant No. 2011CB302903, the National Natural Science Foundation of China under Grant No. 60873231, by the "Six Kinds Peak Talents Plan" project of Jiangsu Province under Grant No. 11-JY-009 ; the Nature Science Foundation of Jiangsu Normal Higher University under Grant No. 11KJB510003; China Postdoctoral Science Foundation funded project under Grant No.2012M511252 and Jiangsu Province Postdoctoral Science Foundation funded project Grant No.1102014C.

References

[1] "Cloud Computing- The BlueCloud Project ", www.ibm.com/developerworks/websphere /zones /hipods /, Oct. 2007.
[2] http://www.ibm.com/developerworks/lotus/library/lotuslive-intro/. [retrieved; June, 2012].
[3] http://code.google.com/intl/zh-CN/appengine/.[retrieved; June, 2012].
[4] http://aws.amazon.com/ . [retrieved; June, 2012].
[5] http://hadoop.apache.org/.[retrieved; June, 2012].
[6] Cloud Security Alliance, "Security guidance for Critical Areas of Focus in Cloud Computing", April 2009.
[7] J. Girard and J. Pescatore, " Teleworking in Cloud: Security Risks and Services" – A Gartner Report, May 15 2009.
[8] J. Viega, "Cloud Computing and the Common Man", IEEE Computer Magazine, Aug. 2009, pp. 106-108.
[9] R.L. Grossman, "The Case for Cloud Computing," IT Professional, vol. 11, no. 2, 2009, pp. 23–27.
[10] P. Mell and T. Grance, "Effectively and Securely Using the Cloud Computing Paradigm," Nat'l Inst. of Standards and Technology (NIST), 2009.
[11] http://europa.eu/legislation_summaries/consumers/consumer_information/l21253_en.htm, [retrieved; June, 2012].
[12] A. Shamir, "Identity-based cryptography and signature schemes," [J].Advances in Cryptology, CRYPTO'84, Lecture Notes in Computer Science, vol. 196, 1985, pp. 47-53.
[13] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," [J]. in Advances in Cryptology, CRYPTO 2001, Lecture Notes in Computer Science, vol. 2139, 2001,pp. 213-229.
[14] A.K. Jain, "Biometric recognition," Nature, vol. 449, 2009, pp. 38-40.
[15] http://tech.sina.com.cn/t//03246615270.shtml. [retrieved; June, 2012].
[16] A.Kleinjung, K.Franke, F.Lenstra. Factorization of a 768-bit RSA modulus, v 1.0. International Association for Cryptologic Research ePrint archive. January 7 2010.
[17] K.Lauter, "The advantages of elliptic curve cryptography for wireless security," [J].IEEE Wireless Communications, vol. 11, no. 1, Feb 2004, pp. 62-67.
[18] M. Turk and A. Pentland, "Eigenfaces for recognition," J. Cognitive Neuroscience, vol. 3, no. 1, 1991, pp. 71–86.
[19] L. Wiskott, J.-M. Fellous, N. Kruger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," IEEE Trans. PAMI, vol. 17, no. 7, 1997, pp. 1-23.

 ISBN: 978-1-61208-216-5

Dynamic Scenarios of Trust Establishment in the Public Cloud Service Market

Soyoung Kim
Technology Opportunity Research Team
Korea Institute of Science and Technology Information
Seoul, Korea
e-mail: sykim8171@kisti.re.kr

Junseok Hwang, Jörn Altmann
Technology Management, Economics and Policy Program
Seoul National University
Seoul, Korea
e-mail: Junhwang@snu.ac.kr, jorn.altmann@acm.org

***Abstract*—The adoption of the public cloud by firms and individuals has been slowed because of the lack of trust. This research seeks the rules of trust establishment between the public cloud providers and users through signaling game theory, analyses dynamic scenarios in which the pervasive distrust arises, and suggests policy guidelines. The theoretical analysis results suggest that the most critical task is to make a pool of trustworthy public cloud service providers to establish an efficient market. The results also show that prudent policy design is desirable. Specific case studies and simulations will be conducted as further studies.**

Keywords-public cloud computing; trust; signaling; equilibrium; dynamic.

I. INTRODUCTION

The public cloud has been a valuable tool for firms and individual users to reduce their Information Technology costs. A number of public cloud services such as Amazon's AWS or HP's cloud service have been launched. Even telecom vendors, contents providers, web portals and small Information Technology solution vendors are participating in the race between public cloud services. A few companies have been started to compete on price as competition has intensified [1].

However, the users may not select a cloud service only by its price and performance. The criteria for selecting a cloud service are not only these two factors. Trustworthiness and reliability are also important criteria for selecting a cloud service. Therefore, the establishment of trust is one of the major challenges for the growth of the public cloud market [2]. The users' concerns about security and privacy threats hinder the diffusion of the public cloud [3]. The public cloud market now needs policy solutions to address the users' concerns rather than technological solutions [4].

This study analyses, with game theoretical insights, the process of trust establishment and distrust pervasiveness when users select a public cloud service. In particular, the signaling game is adopted to find several types of the equilibrium and to analyze several dynamic paths from equilibrium. Policy guidelines are also discussed with the dynamic scenarios.

The remainder of the paper is organized as follows. The next section reviews the related literature. Section III proposes the trust signaling game in the public cloud market. Section IV investigates the dynamic scenario of trust establishment. Section V suggests preliminary results and discussion. Finally, Section VI provides a conclusion and a future work.

II. LITERATURE REVIEW

Research on trust establishment and management related to cloud services has increased as more kinds of cloud computing have been provided to personal users and private companies. Researchers have focused on the issues of possible risks and threats, such as data loss and personal information disclosure [3]. Some researchers have pointed out that these risks and threats to security and privacy had slowed down the adoption of cloud computing services [5]. Some researchers have proposed identity management and authentication systems [6] for mitigating those risks and threats or have suggested a reputation mechanism based on a trust management framework [7].

Research on trust management related to network based transactions between unknown users has a history of decades [8, 9]. These trust management frameworks mostly have their theoretical background in game theory, particularly 'the prisoners' dilemma' [10].

Another type of game, 'the signaling game [11]' would be useful to analyze the process that could help a user select the most trustworthy (or productive) provider among several of them, especially when information asymmetry exists between a user and a provider so that a user cannot know the exact type of a provider. Several studies adopted the signaling game to develop the autonomous agents' strategies for selecting their partners on a network [12, 13]. Most of these studies focused on finding the best strategy of an individual agent rather than finding policies that make a socially efficient equilibrium.

A particular piece of research in the political science field adopted the signaling game to analyze the dynamics of general trust in society [14]. It showed how society's trust tends to oscillate between high and low levels in the long run. However, the study more focused on scrutinizing in the cycles of the general trust levels in a society rather than finding a solution to address problems of pervasive distrust.

Based upon previous research, this paper focuses more on finding policies and solutions to make a socially efficient equilibrium and to address an emergence of generalized distrust.

 ISBN: 978-1-61208-216-5

III. TRUST SIGNALING GAME IN THE PUBLIC CLOUD MARKET

Recently, the public cloud service market has had a number of providers, so it is almost a competitive market. Vendors try to increase the probability of being selected by users through advertising their performance, service prices, or trustworthiness. Users make their decisions based on these signals from vendors. This section firstly investigates the criteria for the existence of a stable equilibrium when a number of providers and users send and receive signals and make partnerships.

A. Process of Trust Establishment

The criteria for selecting partners for users are price, performance, trustworthiness, and so on. This theoretical analysis focuses on trustworthiness.

The process of trust establishment has roughly three steps [15]. The first step is when the market initiates before the trust develops concretely. A user faces the signaling game situation in which a user meets an unknown cloud service provider. The provider sends a characteristic signal to the user and the user makes a decision of selecting a partner by investigating the provider's signals with proper price. Then the connected partners transact or communicate.

The second step is the process of trust formation. As the first step is repeated, the transaction history and a trust relationship are accumulated. The total trust level of the market can increase or decrease with specific paths of trust formation.

The third step is the steady-state. Once the trust establishment reaches a stable equilibrium, a small loss or disturbance of trust cannot affect the equilibrium. This study focuses on what factors make a successful path from the second step to the third step and what factors make the transition a failure.

B. Fundamental Rules of Trust Establishment

In the first step, a service provider intends to increase the probability of being selected by users by signaling his/her trustworthiness in various ways. Simultaneously, a user observes those signals and decides whether or not to trust the provider. Our previous work briefly analyzed this signaling game model and suggested three propositions about signaling cost structures and market environment conditions in network based transactions [16].

Service providers, or cloud providers in this case, are divided into two types. One type is the good provider who observes the promised rules and the other type is the bad provider who violates the rules or does damage to the partner. The proportion of bad type providers in the total provider population is denoted by π_B ($0 \le \pi_B \le 1$). A provider sends signal e ($0 \le e \le 1$) to users, and a single signal costs c(e).

Users are all the same type. A user receives a signal from a provider, examines the signal, estimates the type of the provider, and suggests a charge for the trustworthy transaction, w(e).

Once the partner and the charge are determined and the transaction conducted, the payoff for the user is subsequently fixed. The payoff for a user varies with the type of partner. If a user meets a good type provider, the user receives the proper value of cloud service, v ($v \ge 0$) and the provider also receives the proper payoff, v. However, a bad type partner does not deliver the proper value of cloud service and does damage to the user with an amount of 'L' ($L \ge 0$). Therefore, the bad type provider extort the payoff v and the additional value L from the user. What is important here is that the user cannot be aware of the type of his/her partner.

The total expected utility for the bad type provider is determined by the following equation: $u_B(e)=v+L+w(e)-c_B(e)$ and the for good type provider is determined by the following equation: $u_G(e)=v+w(e)-c_G(e)$.

Without a signal, the user suggests the fee to the unknown provider for the cloud computing service as the following equation: $\bar{w} = -\pi_B(v+L)+(1-\pi_B)v = v-\pi_B(2v+L)$. This means the expected payoff for a single transaction.

In this model, the following three propositions are concluded.

- Proposition 1. (The separating equilibrium) When the level of trustworthiness of a participant is used as the signal, the signal can be effective in distinguishing one provider from another, assuming the cost of the trust level signaling is sufficiently distinct from each other.
- Proposition 2. (The pooling equilibrium) The equilibrium in which the two types of providers select the same trustworthiness level as a signal is not stable if the signaling cost structure is distinct.
- Proposition 3. The effectiveness of the trustworthiness level signaling depends on the proportion of bad type participants in the market.

For example, if the trust signaling cost of the bad type provider is $c_B(e)=e$ and the cost of good type provider is $c_G(e)=\gamma e$ ($0<\gamma<1$), the user distinguishes the good type provider from the bad type provider with only their signals, as long as the equilibrium signal e* falls into the following range in Equation (1). In this equilibrium the good type provider selects e=e* and the bad type provider selects e=0.

$$v \le e^* \le \frac{v}{\gamma} \tag{1}$$

If two types of provider select their signals in the range of Equation (2), they can select the same level of signal as equilibrium. However, it is an unstable state.

$$e^* \le v - \pi_B(2v+L) \tag{2}$$

Proposition 3 means that the costly signaling regime is useful only if the proportion of bad type providers falls into the range of Equation (3).

$$\gamma\frac{v}{2v+L} < \pi_B < \frac{v}{2v+L} \tag{3}$$

 ISBN: 978-1-61208-216-5

IV. DYNAMIC SCENARIO OF TRUST ESTABLISHMENT

The trust signaling game described in Section III is a static and single round situation. The second process of trust establishment is a dynamic process in which the trust relationships stay in equilibrium or leave it.

A. Potential for a Pareto Improvement in the Equilibrium

When the separating equilibrium has been reached, the equilibrium signal of a good type provider is e* and the signaling cost is γe. A bad type provider does not send a signal and pay any cost. In a dynamic situation, bad type providers gradually leave the market and the ratio of bad type providers, π_B, decreases.

If π_B decreases down to this level, good type providers have incentives to lower their signaling costs so that increases the total payoff. In terms of individual rationality, the expected payoff of a good type provider if he/she decides not to send a signal in this situation is shown in the following Equation (4).

$$u_G(e)|_{e=0} = 2v - \pi_B(2v+L) \quad (4)$$

The expected payoff of Equation (4) is more than 2v-γe*. There is potential for a Pareto improvement when π_B decreases gradually. It means that the expected payoff of one player can increase without decrease of the other's expected payoff. It reaches the Pareto efficient state when π_B finally falls to zero. However, users stay with the same payoffs because of the assumption of the zero profit condition.

The situations where providers and users believe each other to conduct themselves properly and choose each other as partners make the transactions and communications more efficient. This is the benefit of an economy of trust.

B. The Countinuous Needs of Costly Signals

The Pareto optimum described in the previous subsection is not stable, because a good type provider can become a traitor or change his/her type in the real world market. Or, a newcomer provider of bad type can enter the market.

When a single bad type provider appears in the market with no signaling, π_B turns into a higher value than zero. This traitor or newcomer can gain a higher payoff than any other good type providers with an amount of 'L'. The users lose their payoff by the same amount. The sum of payoffs of all market participants does not change; however the share of users transfers to the share of traitors or newcomers.

Once this transformation happens, users calculate the proportion of bad type providers again, and introduce the price related to the proportion, and finally the market adopts the costly signaling regime.

C. The Dynamics of the Trust Equilibrium Shift

The last situation of dynamic trust transition is when the proportion of bad type providers exceeds the range defined by the third proposition of Section III. The separating equilibrium with costly signaling is in stable equilibrium; therefore users can still distinguish a good type provider from a signal only if the value of πB is in the range defined by Equation (3). When the damage from a bad type provider's behavior increases exceptionally, the separating equilibrium in which the two types of providers select the different trustworthiness level as a signal fails to stay stable. Figure 1 illustrates the relationship between the dynamic states of trust establishment and the proportion of bad type providers. Part (a) indicates the possible region of separating equilibrium, part (b) is the transition region of separating equilibrium and non-signaling pervasive trust and (c) is the market reduction region.

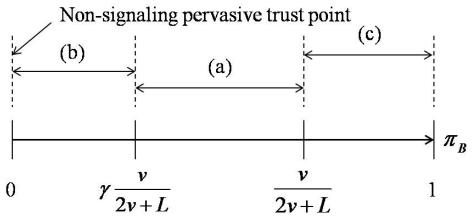


Figure 1. Dynamic states of trust establishment

V. SIMULATION DESIGN AND DISCUSSION

The dynamics of trust can be more clearly understood with a simulation based on parameters which reflect the market conditions in the real world. Figure 2 shows the causal loop diagram of a dynamic model of trust establishment in the public cloud service market. The proportion of bad type providers, π_B, is the most central variable which affects many other variables and receives feedback. This variable can be controlled by these exogenous variables which are denoted by 'E' with policy decisions.

The ratio of a good type provider's signaling cost to a bad type provider's cost, γ, affects the signaling costs of two type providers and the levels of signals are affected by these costs. The probability of being selected by a user and the signaling cost affect the utility of a provider as well as the non-signaling price, $\bar{w}$. The utility of a provider affects the entrance and leaving rate of a provider. The amount of damage from misbehavior by a bad type provider, L affects the utility of a bad type provider

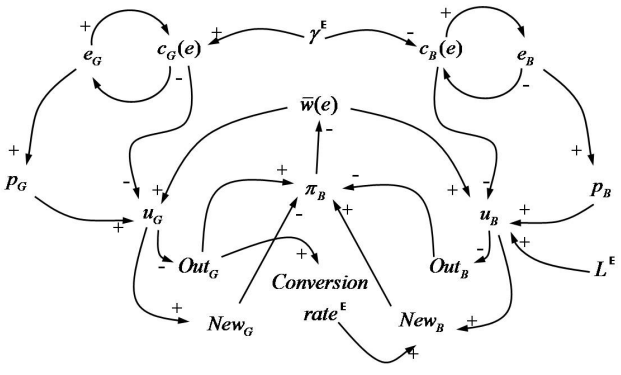


Figure 2. The causal loop diagram of a dynamic model of trust establishment in the public cloud service market

The results of designed simulation is expected to show the quantitative relationship between the variables which are illustrated in the Figure 2.

It is obvious that non-signaling pervasive trust is the optimal state of the market. The second best state is when users can easily distinguish the good type providers with signaling in the separating equilibrium. The best or second best states can be realized by prudent policy design which can control several related variables in the causal loop diagram.

The basic condition is to increase the signaling cost for bad type providers more than for good type providers. Reputation based mechanisms or third party authorization mechanisms can be possible methods to increase the signaling cost of a bad type provider.

If the costly signaling is maintained after most of the bad type providers have retired, the conversion of good type providers into bad ones or the entrance of new bad type providers can be blocked. However, costly signaling is inefficient when the proportion of bad type providers is substantially low. Then, it is worth considering the community of good type providers or their agreement for an efficient market. Either monitoring and penalty contracts or agreements have to exist in such communities [17].

VI. CONCLUSION AND FUTURE WORK

The fundamental rules and dynamic scenario of trust establishment are important factors that should influence the decision makers in the industry sector or a government which intends to promote the public cloud service market.

The theoretical analysis results of this research suggest that the most critical task is to make a pool of trustworthy public cloud service providers to establish an efficient market. The results also show that prudent policy design, which makes signaling costs different for different types of providers is desirable. It also shows that even in a trustworthy market, minimum monitoring and penalty contracts are needed and individual users have to invest in security at an optimal level.

Future work will verify the theoretical model of this paper with simulations and specify the dynamic scenarios of trust establishment and transition with several case investigations into various types of cloud computing services. In particular, the presented causal loop diagram will be validated and its parameters will be examined.

ACKNOWLEDGMENT

This research was supported by the KCC (Korea Communications Commission), Korea, under the CPRC (Communications Policy Research Center) support program supervised by the KCA (Korea Communications Agency). (KCA-2012-(11-941-1-005))

REFERENCES

[1] Larry Dignan, Cloud's price race to zero: Microsoft cuts Azure pricing, eyes Amazon, article of www.zdnet.com, March 9, 2012. http://www.zdnet.com/blog/btl/clouds-price-race-to-zero-microsoft-cuts-azure-pricing-eyes-amazon/71246

[2] S. M. Habib, S. Ries, and M. Mühlhäuser, "Cloud Computing Landscape and ResearchChallenges regarding Trust and Reputation," in Symposia and Workshops on Ubiquitous, Autonomic andTrusted Computing, Xi'an, China, 2010, pp. 410–415.

[3] S. Pearson, "Taking account of privacy when designing cloud computing services." In ICSE Workshop on Software Engineering Challenges of Cloud Computing, Vancouver, Canada, May 2009, pp. 44-52.

[4] M. Nelson,"The Cloud, the Crowd, and Public Policy," Issues In Science And Technology, vol. 25, no. 4, 2009.

[5] H. Takabi, J.B.D. Joshi and G.J. Ahn. "Security and Privacy Challenges in Cloud Computing Environments." IEEE Security & Privacy, vol. 8, no. 6, 2010, pp. 24-31.

[6] L. Yan, C. Rong, and G. Zhao. "Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography,". In The First International Conference on Cloud Computing, pp. 167–177, 2009.

[7] K. Hwang, S. Kulkarni, and Y. Hu, "Cloud Security with Virtualized Defense and Reputation-Based Trust Management," IEEE Int'l Conf. Dependable, Autonomic, and Secure Computing (DASC 09), IEEE CS Press, 2009.

[8] P. Resnick and R. Zeckhauser, "Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system," working Paper for the NBERWorkshop on Empirical Studies of Electronic Commerce, 2000.

[9] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," Decis. Support Syst. Vol. 43, 2007, pp. 618–644.

[10] R. Axelrod, The Evolution of Cooperation, Basic Books: New York, 1984.

[11] A. M. Spence, (1973). "Job market signaling," Quarterly Journal of Economics, vol. 87, no. 3, pp. 355–374.

[12] A. Lopez-Paredes, M. Posada, C. Hernandez, and J. Pajares, "Agent based experimental economics in signaling games in Complexity and artificial markets," Lecture Notes in Economics and Mathematical Systems, vol. 614, 2008, pp.121–129.

[13] A. Patcha and J. Park, "A Game Theoretic Formulation for Intrusion Detection in Mobile Ad Hoc Networks, " International Journal of Network Security, vol.2, no2, 2006, pp.131–137.

[14] T. Ahn, and J. Esarey "A Dynamic Model of Generalized Social Trust," Journal of Theoretical Politics, vol. 20, no. 2, 2008, pp. 151–180.

[15] M. Head, and K. Hassanein, "Trust in e-Commerce: Evaluating the Impact of Third-Party Seals", Quarterly Journal of Electronic Commerce, vol. 3, no. 3, 2002, pp. 307-325.

[16] S. Kim, and J. Hwang, "Theoretical Analysis and Simulation to Investigate the Fundamental Rules of Trust Signaling Games in Network-based Transactions", The Third International Conference on Future Computational Technologies and Applications, 2011.

[17] J. Hwang, S. Kim, H. Kim and J. Park, "An optimal trust management method to protect privacy and strengthen objectivity in utility computing services," International Journal of Information Technology & Decision Making, vol. 10, issue 02, 2011, pp. 287-308.

 ISBN: 978-1-61208-216-5

De-replication: A Dynamic Memory-aware Mechanism

Manu Vardhan, Paras Gupta, Dharmender Singh Kushwaha
Department of Computer Science and Engineering
Motilal Nehru National Institute of Technology Allahabad
Allahabad, INDIA
e-mail: {rcs1002, cs1006, dsk}@mnnit.ac.in

***Abstract*—Resource replication in distributed environment produces issues of secondary storage. De-replication of resources is required when replication mechanism is hindered due to lack of secondary storage. This paper introduces de-replication approaches that depend upon last modification time, number of replica available and resource size. Comparative study shows that de-replication can be used to overcome the space overhead issue and reduces the de-replication time. Result shows that in case the space required is same but number of files to be de-replicated varies, de-replication time also varies depending on number of files to be de-replicated. De-replication time will be more for case having large number of files. With the proposed approach, if file size increases by the multiple of 7, de-replication time will get increase just by the multiple of 1.5. This shows that de-replication time is decoupled from size of files that are de-replicated on the fly dynamically and does not increase proportionally with respect to file size.**

Keywords-De-replication; Distributed Systems; Replication

I. INTRODUCTION

Use of computer systems and Internet is becoming the part of day to day life, with the increasing demand for the services provided by them. To fulfill the requirement of services requested by an individual, service availability is an important issue. Distributed systems provide the environment to various experts, where services, resources and information are distributed and can be accessed by the members of that environment, as compared to the centralized systems.

A basic definition of a distributed system in [1] is that a distributed system is a collection of independent entities that cooperate to solve a problem that cannot be individually solved. This is a term that describes a wide range of computers, from weakly coupled systems, such as wide area networks, to strongly coupled systems, such as local area networks, to very strongly coupled systems such as multiprocessor systems [2].

Replication is a mechanism of service or resource placement to provide their availability in case of unavailability of resources and services. Replication is how to replicate data and request actors using adaptive and predictive techniques for selecting where, when and how fast replication should proceed [3].

De-replication is a mechanism to de-replicate / garbage-collect data or request actors and optimizes utilization of distributed storage based on current system load and expected future demands for the object [3].

De-replication is done to optimize the utilization of storage space when the demand for a resource arises. The file to be de-replicated must be carefully taken into consideration of the future demands of a file. File currently being serviced cannot be de-replicated. The number of previously replicated files selected for de-replication can fulfill the requirement for storage space need of the upcoming file to be replicated. De-replication is considered as a part of resource management process where as replication is considered as a part of resource placement process.

The rest of the paper is organized as follows. The next section discusses a brief literature survey of existing theories and work done so far. Section III discusses the problem definition. Section IV describes the proposed solution, followed by the results and discussion section. Finally, Section V concludes the work followed by references.

II. RELATED WORK

Various resource management policies and mechanisms are globally available that represent a step towards the adaptive resource management techniques, thus improving the utilization of resources, which results in improving the overall performance of the system by reducing several overheads. Venkatasubramanian [3] discusses about the security and timeliness application requirements using a customizable and safe middleware framework called CompOSE|Q. He describes the design and implementation of CompOSE|Q, which is a QoS-enabled reflective middleware framework. Also, to improve the performance of the system in the field of continuous media application, resource management technique is helpful in improving the utilization of resources. Chou [4] describes various resource management policies on threshold basis in context of continuous media (CM) servers in the area of multimedia application. Venkatasubramanian[5] discusses the two replication policies, these are static and dynamic. The division is based upon the number of copies of a file which is termed as degree of replication. In static replication policies, the degree of replication is constant, while dynamic replication policies allow it to vary with time. Santry [6] identified four file retention policies for Elephant and have implemented these policies in their prototype. The policies are viz. Keep One, Keep All, Keep Safe and Keep Landmarks. Keep One provides the non versioned semantics of a standard file system. Keep All retains every version of the file. Keep Safe provides versioning for undo but does not retain any long term history. Keep Landmarks enhances Keep Safe to also retain a long-term history of landmark versions. Hurley and Yeap [7] propose a file de-replication

 ISBN: 978-1-61208-216-5

method based on beta time interval that decides the frequency of invoking the de-replication operation. Over time, all files will eventually be candidates for migration/replication. Although many exist, the one we choose is as follows: every beta time units (where beta is a uniform time interval which defines the time between de-replication events), storage sites will decide which file qualifies for de-replication. The de-replication policy chosen applies the least recently used concept (i.e., the file selected for de-replication is the file which was not requested for the longest period of time at the storage site). Once the file has been selected, it will be removed from this storage site. Using beta, it is possible to create a variety of de-replication policies: the smaller the value of beta, the greater the frequency of de-replication, and the larger the value of beta, the longer a file copy remains in the system. Resource replication is basically of two types, active and passive. In passive replication, all the resources are fixed in advance depending upon the application requirement. In active replication, mutual information about the peer nodes is maintained and the replicated resources can be accessed at any site. The traditional resource replication is passive and does not participate in the decision on when to replicate, where to replicate and the number of copies to replicate. In a blind-replica service model proposed by Tang [10], request routing is independent of where the replicas are located. Each replica simply serves the requests flowing through it under a given routing strategy. Various replication strategies have been proposed on the basis of the relative popularity of individual files, based on their query rate. Helen [8] proposed a query-based file popularity approach for replication. Common techniques include the square-root, proportional, and uniform distributions. File clustering-based replication algorithm in a grid environment is proposed by Hitoshi [9], which presents the location based replication mechanism. The files stored in a grid environment are grouped together based on the relationship of simultaneous file accesses and on the file access behavior.

III. PROBLEM DEFINITION

During replication, when a File Replicating Server (FRS) creates a replica of file on the peer nodes, space management issue arises, i.e., whether space is available or not in the secondary storage of the peer nodes on which the file needs to be replicated. If space is available, the file will get copied, but if space is not available de-replication of previously replicated files needs to be done in the secondary storage of that peer node.

De-replication of files will take place in a manner such that it will fulfill the size requirement of upcoming files. While maintaining the space management overhead, decision for deleting a file, depends on three criteria that are discussed in Section III-A.

A. Parameters to be Used

Solution to this problem will be represented on the basis of three parameters of a file which are last modification time of the file, number of replica available of a file and file size.

- *Last Modification Time of a File:* Last modification time is the time at which the file was last modified or last used.
- *Number of Replicas Available of a File:* Number of replicas available of a file is a count on number of copies available for a particular file. Whenever a copy of file is created, it will increase the number of replicas available of a file.
- *Size of a File:* File size is the size of a file required on a disk.

IV. PROPOSED SOLUTION

With everything being lodged on Internet, computing paradigm is changing fast to harness this capability. Many information servers and files are resident on various machines and this can be effectively utilized by the users. We present a scenario discussed in Section IV-A, although on a smaller scale where geographically disparate clusters interact with each other for information sharing through replication. Each of these clusters are owned by respective organizations.

In our proposed model, we talk about space overhead in replicating file on the storage site. If space is available, the file will get replicated; otherwise, de-replication of previously replicated files needs to be done in that directory.

A. Architecture Used

One node in each cluster is designated as FRS. FRS can also be replicated on some other node in the cluster for backup and recovery. The scenario presented in the paper is illustrated in Figure 1 and is elaborated subsequently.

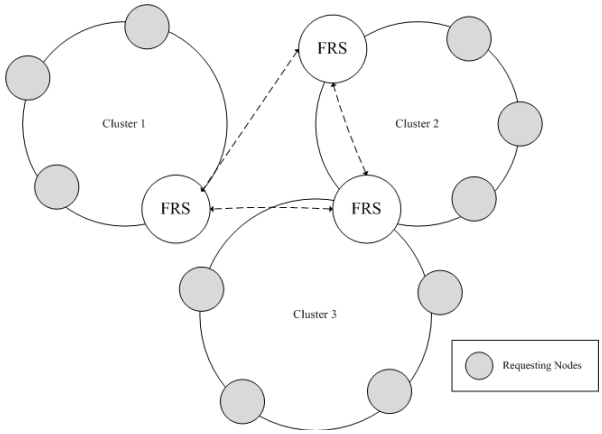


Figure 1. Architecture

The proposed architecture consists of loosely coupled systems, capable of providing various kinds of services like replication, storage, I/O specific, computation specific and discovery of resources. Based on the application requirement, the resources are made available to other nodes. Figure 1 shows a network of three clusters that are connected to each other via intercommunication network. Each cluster consists of a group of trusted nodes and a File Replicating Server (FRS) assigned to these nodes. A FRS

can be 'local' or 'remote'. A FRS is assigned to a subset of nodes known as local FRS and FRS positioned outside that cluster, will be called as remote FRS. Each subset of nodes (denoted as requesting nodes) receives the list having IP-address of remote FRS, to increase fault tolerance capability. But, the nodes of a cluster will send the file request only to the local FRS. In case of the failure of the local FRS, a node can automatically select a remote FRS from the list and file request will be routed to the selected remote FRS. This makes the model robust and capable of handling crashes in case of local or even remote FRS fails. The system will keep functioning under all circumstances and will never come to halt. Each FRS maintains two tables:

- File request count table with the following attributes: <file_id, file_name, request_count, meta data>.
- Peer FRS table with the following attributes: <FRS_IP, FRS_PORT>.

Each FRS is informed whenever a new FRS is added to the network, to updates its peer FRS table. FRS does not monitor and maintains the status of remote FRS, instead FRS request for the current status of remote FRS on-demand. FRS status can either be 'busy' or 'ready'.

Threshold based file replication works as follows:

Each local FRS is responsible for accepting the file request and based on its current status (checks if the number of requests currently serving for a particular file is below the threshold or not), in the following manner:

- If the status of local FRS is 'ready', the local FRS will fulfill the request.
- If the status of local FRS is busy, it looks for a remote FRS that can handle the request, by one of the following manner, described as under:

The local FRS contacts the remote FRS that can handle the request by the available copy of the requested file i.e. the status of remote FRS is ready. If not so, the local FRS contacts those remote FRS on which the requested file is not available. In that case file replication will be initiated, by the local FRS of the cluster and the file replica will be created on remote FRS on which the file is not available. For both the cases mentioned above, IP address of the remote FRS that can handle the request will be send to the requesting node. On receiving the IP address, the requesting node will connect to the remote FRS and receives the file, without any user intervention. Thus the overhead of polling and broadcasting is reduced.

B. Approaches Proposed for De-replication

De-replication of files will take place in a manner such that it will fulfill the size requirement of upcoming files. While maintaining the space management overhead, three approaches for file de-replication are discussed below.

1) Last Modification Time Based Approach: In this approach, files are sorted on the usage basis file that was not requested for longest period of time will be selected for de-replication. A drawback of this approach is that if only one requested file is there before deletion, it causes loosing of information. So, a check is performed before de-replicationwhich will be done on number of replica available basis approach.

2) Number of Replicas Available of a File Based Approach: In this approach, files having many copies or the files with more than one replica are de-replicated only when there is not sufficient space available for new replicated files. Files with one replica are not de-replicated to avoid losing information of the file. In this case, before the de-replication of file, a check is performed, whether or not there are other copies of file available or not. If only single copy of file exists in the system, in that case next probable file for de-replication will be selected from the sorted file list on the basis of last modification time.

3) File Size Based Approach: File size based de-replication approach is used when time required for de-replication considered as important factor. When there is a very little difference in the last modification time of the two files and number of replicas available of both files is more than one, de-replication of file with minimum file size among them will take place to avoid the delay in the process and complete it in the less time.

The proposed approach for de-replication will be described in Figure 2. The detailed description of the number labeled arcs will be described in sequential manner as follows:

1. *Node A of cluster$_1$* sends connection request to *FRS$_1$*.
2. *FRS$_1$* sends ip addresses of peer FRS and resource list to n*ode A of cluster$_1$*.
3. *Node A of cluster$_1$* sends request for file f_1 to *FRS$_1$* at time t_0.
4. *Node A of cluster$_1$* starts receiving requested file f_1 from *FRS$_1$*.
5. *Node D of cluster$_1$* sends connection request to *FRS$_1$*.
6. *FRS$_1$* sends IP addresses of peer FRS and resource list to n*ode D of cluster$_1$*.
7. *Node D of cluster$_1$* sends request for same file f_1 to *FRS$_1$* at time t_1.

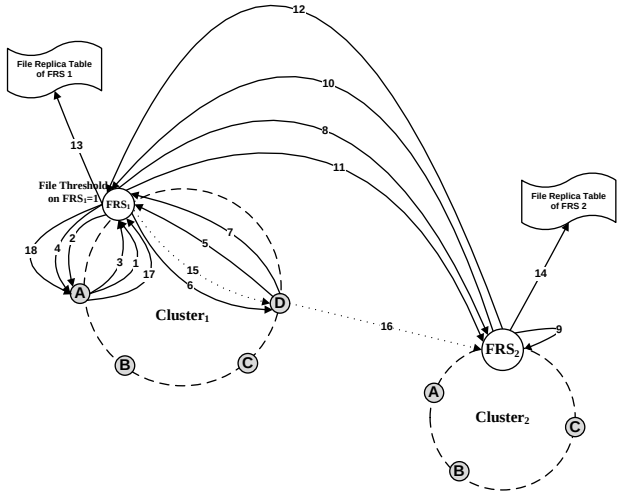


Figure 2. Proposed Model

8. As FRS_1 can fulfill only one request at a time because the threshold value for a particular file on FRS_1 is *1*, so *node D of cluster*$_1$ will receive the requested file from another *FRS* in the system, here FRS_2, to fulfill its request. To fulfill the request of *node D of cluster*$_1$, replication of requested files is initiated by FRS_1 as the requested file is not present on FRS_2. This is because the FRS does not maintain any information about the "requesting node (e.g. *node D*)" at any point of time. So FRS_1will replicate the requested file to other FRS as its shared resource information is being maintained, as discussed in section IV-A. Now, FRS_1sends the size of the file to be replicated to FRS_2.
9. FRS_2 does not accept the file replication request because of space/storage scarcity. FRS_2 initiates de-replication operation on set of previously replicated files. The required amount of space is made available on FRS_2. If the secondary storage on FRS_2did not contains any replicated files then user interruption will come, as de-replication of non-replicated file is not allowed.
10. FRS_2 sends message 'ready to receive file f_1' to FRS_1.
11. FRS_1 starts replicating the file f_1 to FRS_2.
12. FRS_2 sends message 'replication of file $f_1$to be done successfully' to FRS_1.
13. FRS_1 updates its file replica table.
14. FRS_2 updates its file replica table.
15. FRS_1 sends IP address and port of FRS_2 to n*ode D of cluster*$_1$informing that the file f_1 is now available on FRS_2.
16. Request of *node D of Cluster*$_1$ for file f_1 will now be fulfilled by peer FRS, FRS_2.
17. After some time *node A of cluster*$_1$ request same file $f_1$from FRS_1.
18. In case file with the same name already exists on the n*ode A of Cluster*$_1$,file de-replication will be done on that node then the file transfer from FRS_1 to *node A of cluster*$_1$ will be initiated.

C. Stability Analysis

According to Figure 3, the communication between a requesting node and a FRS (*Source* A and FRS_1) is described as follows: *Source A* sends a file request to FRS_1through$\overline{M_1}$.FRS_1 will receive the request of *Source A* represented as M_1. In return, FRS_1 sends file to *Source A* shown by M_3 received on *Source A* using $\overline{M_3}$.

The total communication between requesting node *Source* A and FRS_1 with internal actions (τ) will be given by equation 1 as follows:

$$SourceA \stackrel{def}{=} \overline{M_1}.M_3.\tau.SourceA \quad (1)$$

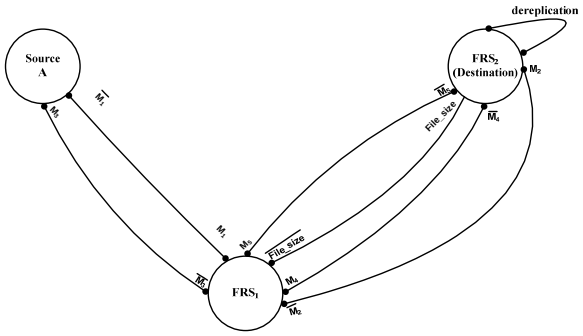


Figure 3. File De-replication Model Flow Graph in Process Algebraic Approach

Also as shown in Figure 3, communication between the two existing FRS in the architecture (FRS_1 and FRS_2) is described as follows: FRS_1 will send file size of the file to be replicated using$\overline{File_size}$ which will be received at FRS_2 end by$File_size$. When file size is received by FRS_2, it initiates de-replication operation on set of previously replicated files which will be represented by $\tau.derplicate\_file$, which is file de-replication with internal actions (τ). After the successful completion of de-replication operation, the required size for replication will be available on FRS_2. Now, FRS_2 will send 'ready to receive replicated file' message to FRS_1 represented through$\overline{M_4}$. FRS_1 received this message using M_4. After receiving the message, FRS_1 will send the file to be replicated to FRS_2 represented by message $\overline{M_2}$. FRS_2 will receive the file send by FRS_1 represented as M_2. When the file will be replicated successfully on FRS_2, it will send a message 'successful replication done' to FRS_1 by$\overline{M_5}$which was received by FRS_1 using M_5.

1) Illustration of State Transition of Source Node: As shown in Figure 3, FRS_1 will act as a source node. Status change illustration of source node (FRS_1), as shown in Figure 4, will be described as follows:

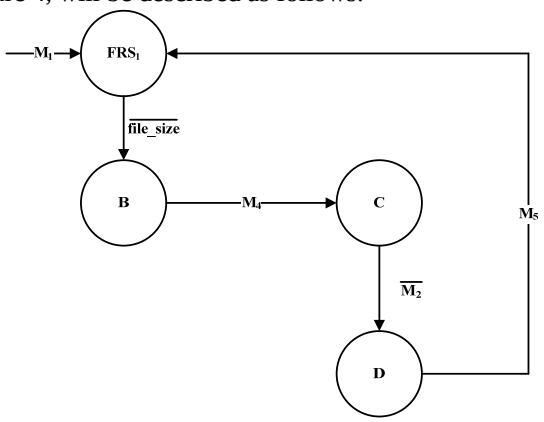


Figure 4. State Transition Diagram of Source Node (FRS_1)

After the action of sending file size of the file to be replicated through message $\overline{File\_size}$, source node (FRS_1) transit to state B of FRS_1 shown in equation 2,

$$FRS_1 \stackrel{def}{=} \overline{File\_size}.B \quad (2)$$

State B of FRS_1 switch to state C of FRS_1 through message M_4, which represents the action of receiving 'ready to receive replicated file' message by FRS_1shown in equation 3,

$$B \stackrel{def}{=} M_4.C \quad (3)$$

State C of FRS_1 switch to state D of FRS_1 through message $\overline{M_2}$ which represents the action of sending the file to be replicated by FRS_1 shown in equation 4,

$$C \stackrel{def}{=} \overline{M_2}.D \quad (4)$$

State D of FRS_1 upon the action of receiving message 'successful replication done' by FRS_1through message M_5 switch to starting state FRS_1 shown in equation 5,

$$D \stackrel{def}{=} M_5.FRS_1 \quad (5)$$

From the equations 2, 3, 4 and 5 of various states of FRS_1, we can build the definition of FRS_1, which is defined as by the equation 6:

$$FRS_1 \stackrel{def}{=} M_1.\overline{File\_size}.M_4.\overline{M_2}.M_5.\overline{M_3}.FRS_1 \quad (6)$$

*2) Illustration of State Transition of Destination Node:*As shown in Figure 3, FRS_2 will act as a destination node. Status change illustration of destination node (FRS_2) as shown in Figure 5 will be described as follows:

After the action of receiving file size of the file to be replicated through message$File_siz$, destination node (FRS_2) transit to state E of FRS_2 shown in equation 7,

$$FRS_2 \stackrel{def}{=} File\_size.E \quad (7)$$

State E of FRS_2 switch to state F of FRS_2 through message $\overline{dereplicate\_file}$ which represents the action of de-replication on previously replicated files by FRS_2 shown in equation 8,

$$E \stackrel{def}{=} \overline{dereplicate\_file}.F \quad (8)$$

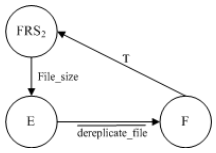


Figure 5. State Transition Diagram of Destination Node (FRS_2)

State F of FRS_2 upon some internal actions (τ) by FRS_2 switch to starting state FRS_2 shown in equation 9,

$$F \stackrel{def}{=} \tau.FRS_2 \quad (9)$$

From the equations 7, 8 and 9 of various states of FRS_2, we can build the definition of FRS_2 which is defined as by the equation 10:

$$FRS_2 \stackrel{def}{=} File\_size.\tau.\overline{dereplicate\_file}.\overline{M_4}.M_2.\overline{M_5}.FRS_2 \quad (10)$$

From the equations 1, 6 and 10, we can build the complete system as defined by the equation 11:

$$FDM \stackrel{def}{=} Source \,||\, FRS \,||\, Destination \quad (11)$$

V. RESULTS AND DISCUSSION

To overcome from the overhead of space management issue, a data structure consisting of a table considered which is described in Table 1. The proposed model is simulated on linux platform and LAN having speed of 100.0 Mbps.

TABLE I. ATTRIBUTES

Attribute Name	Type
Last Modification Date	yyyy-mm-dd
Last Modification Time	hh:mm
File Name	String
File Size	Long
File replica	Integer

Replicated files on the storage site will be sorted based on least recently used parameter which will be obtained using the combination of both last modification date and last modification time. The list of replicated files will be sorted in descending order. Example of a data structure of available files maintained at the storage site is described in Table 2.

TABLE II. DATA STRUCTURE EXAMPLE FOR COMPARISON BETWEEN APPROACHES

Last Modification Date	Last Modification Time	File Name	File Size (in MB)	File replica
2011-12-21	20:08	a.mp3	3	4
2011-12-08	22:48	b.mp3	500	1
2011-11-23	16:36	c.mp3	100	2
2011-11-23	16:03	d.mp3	250	1
2011-11-09	20:11	e.mp3	50	1
2011-11-09	18:47	f.mp3	5	4
2011-11-09	18:43	g.mp3	10	2

The Figure 6 plots efficiency of all the three approaches versus load based on the data shown in Table 2 and the three approaches based on least recently used parameter, replica counts and file size parameters. Efficiency calculated is proportional to the reciprocal of extra memory size vacated during de-replication based on low, low-medium, medium-high and high load for which range of file size (R_f) is R_f<20MB, 20MB<R_f<60MB, 60MB<R_f<100MB, and R_f>100MB, respectively.

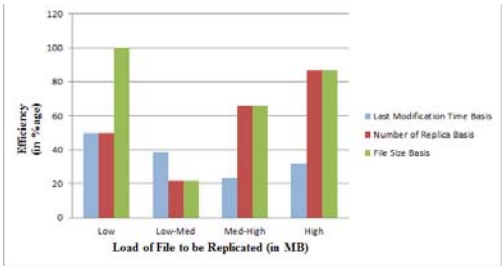


Figure 6. Comparison of the Three Approaches

Unlike 2nd and 3rd approaches (i.e., number of replica available of a file basis and file size basis respectively), 1stapproach(i.e. last modification time basis) is based only on least recently used parameter and disregards the replica counts and file size parameters. Thus it may even delete the last replica of file present in system. While 2ndapproach is based on both least recently used and replica counts parameters and disregards the file size parameter. The 3rd approach is based on all the three parameters, least recently used, replica counts parameters and file size parameter. Most of the time, the percentage efficiency of the 2nd and 3rd approach is equal and better than of the 1st approach, except in case low-medium load. Only in case of low load percentage efficiency of 3rdapproach is better than 2ndapproach. All the three approaches said to be 100% efficient only when space required before and after de-replication is exactly the same.

De-replication time increases, as the number of files not accessed for the longest period and smaller in size, are more as compared to the files that are larger in size. Table 3 shows when the space required is same but the number of files to be de-replicated varies, de-replication time also varies depending on the number of files to be de-replicated. De-replication time will be more for the case having large number of files. Table 3 shows that if file size increases by the multiple of 7, i.e., from 6 MB to 43.9405 MB, de-replication time will get increase by the multiple of 1.5, i.e., from 60millisecond to 98 millisecond. This shows that the de-replication time is decoupled from the size of files that are de-replicated dynamically and does not increase proportionally with respect to the file size.

TABLE III. DE-REPLICATION TIME IN REQUIRED SPACE

Number of Files de-replicated	Space Required (in MB)	Space Freed (in MB)	De-replication Time (in msec)
1	6	6.0523	60
2	7.8607	13.1792	75
3	20.0399	21.0399	77
3	36.2634	39.7985	79
5	36.2634	59.7151	96
5	43.9405	51.0140	98

Finally, equation 11 establishes a relationship between the formal aspect of file de-replication server and its architectural model through process algebra approach. The stability analysis ensures that the system will run in the finite sequences of interaction and transitions. On the basis of these equations, a transparent, reliable and safe file de-replication model is build.

VI. CONCLUSION

This paper proposed an approach that tackles the issue of space overhead in a distributed system environment. The proposed solution resolves this issue of space overhead. De-replication time increases, as the number of files increases that are not accessed for the longest time period and smaller in size as compared to the files that are larger in size. Result shows that, in case when the space required is same but the number of files to be de-replicated varies, de-replication time also varies depending on the number of files to be de-replicated. De-replication time will be more for the case having large number of files. With the proposed approach, if file size increases by the multiple of 7, de-replication time will get increase just by the multiple of 1.5. This shows that the de-replication time is decoupled from the size of files that are de-replicated on the fly dynamically and does not increase proportionally with respect to the file size.

REFERENCES

[1] A. D. Kshemkalyani and M. Singhal, "Distributed Computing: Principles, Algorithms, and Systems", ISBN: 9780521189842, paperback edition, Cambridge University Press, March 2011 (corrects the errata in the 2008 edition). 756 pages.

[2] M. Gupta, M. H. Ammar, and M. Ahamad, "Trade-offs between reliability and overheads in peer-to-peer reputation tracking," Computer Networks, pp. 501-522, 2006.

[3] N. Venkatasubramanian, "CompOSE|Q-a QoS-enabled customizable middleware framework for distributed computing," Electronic Commerce and Web-based Applications/Middleware. in 19th IEEE International Conference on Distributed Computing Systems, pp. 134-139, 1999.

[4] Cheng-Fu Chou, L. Golubchik, and J. C. S. Lui, "Striping doesn't scale: how to achieve scalability for continuous media servers with replication," in 20th International Conference on Distributed Computing Systems, pp. 64-71, 2000.

[5] N. Venkatasubramanian, M. Deshpande, S. Mohapatra, S. Gutierrez-Nolasco, and J. Wickramasuriya, "Design and implementation of a composable reflective middleware framework," in 21st International Conference on Distributed Computing Systems, pp. 644-653, Apr 2001.

[6] D. S. Santry, M. J. Feeley, N. C. Hutchinson, A. C. Veitch, R. W. Carton, and J. Ofir, "Deciding when to forget in the Elephant file system," vol. 33, issue 5, pp. 110-123, Dec 1999.

[7] R. T. Hurley, and Soon Aun Yeap, "File migration and file replication: a symbiotic relationship," IEEE Transactions on Parallel and Distributed Systems, vol. 7, no. 6, pp. 578-586, Jun 1996.

[8] S. Helen, "IRM: Integrated file replication and consistency maintenance in P2P Systems", IEEE Trans. on Parallel and Distributed Systems, Vol. 21, No. 1, pp. 100-113, 2010.

[9] S. Hitoshi, S. Matsuoka and T. Endo, "File Clustering Based Replication Algorithm in a Grid Environment", 9th IEEE/ACM Int. Sym. on Cluster Computing and the Grid, pp. 204-211, 2009.

[10] X. Tang, C. Huicheng, and S. T. Chanson, "Optimal Replica Placement under TTL-Based Consistency", IEEE Transactions on Parallel and Distributed Systems, vol.18, no. 3, pp. 351-363, March 2007.

 ISBN: 978-1-61208-216-5

Evaluating Eucalyptus Virtual Machine Instance Types: A Study Considering Distinct Workload Demand

Erica Teixeira Gomes de Sousa, Paulo Romero Martins Maciel, Erico Moutinho Medeiros, Débora Stefani Lima de Souza, Fernando Antonio Aires Lins, Eduardo Antonio Guimaraes Tavares
Center of Informatics
Federal University of Pernambuco
Recife, Brazil
{etgs,prmm,emm2,dsls,faal2,eagt}@cin.ufpe.br

***Abstract*—Cloud computing paradigm provides virtualized computing resources as a service on demand. This paradigm allows companies focus on their business issues rather than conceiving and managing complex infrastructures. As a consequence, performance evaluation of cloud computing infrastructures has been receiving considerable attention by service providers as a prominent activity for improving service quality infrastructure planning, and selection of software platforms (e.g., Eucalyptus). This paper presents the performance evaluation of virtual machines on Eucalyptus platform considering different workloads. This study provides insights about Eucalyptus system infrastructure suitability for applications with high processing and storage performance requirements. This work provided the evaluation of virtual machine instances types from a private cloud, which was configured with Eucalyptus platform.**

Keywords-cloud computing; eucalyptus platform; performance evaluation.

I. INTRODUCTION

Cloud computing is a combination of technologies that have been developed over the last several decades, which includes virtualization, grid computing, cluster computing and utility computing. Due to market pressure, cloud computing technologies have rapidly evolved in order to let users focus on business aspects rather than complex infrastructures issues, hence fostering business competitiveness [1][2].

Currently, cloud providers provide guarantees on their service levels and when service failures occur, they only offer to refund their customers regarding the infrastructure outages. However, service providers are not inclined to pay penalties that would refund customers for loss of business revenue [3][4]. Cloud providers are not only required to supply correct services but, also, to meet their expectations in the context of performance. Indeed, cloud computing services have been massively expanding, thus demanding companies to offer reliable services, high availability, scalability and security at affordable costs. In such a case, performance evaluation is a prominent activity for improving service quality, infrastructure planning, and for tuning system components [5][6].

Some software systems, such as credit and debit processing applications require different performance levels, quality of services, reliability, and security, which are generally not guaranteed by a public cloud. In these clouds, computing resources are shared with other companies. These companies do not have any knowledge or control of where the applications run. Private cloud is an alternative to companies that need more control over that data. In private clouds, data-center resources of a company are controlled by company's IT staff [7][8]. Eucalyptus is an open source cloud computing platform that implements infrastructure as a service (IaaS) on a collection of server clusters, and such platform allows the creation of private clouds [9].

Different works [10][11] propose an evaluation of the performance of various public cloud computing infrastructures for suitability in scientific applications. Some other papers [12][13] present the performance evaluation of public and private cloud computing storage resources.

In this work, we evaluate the performance of different virtual machines types on the Eucalyptus platform [9] to determine its suitability for applications that demands different processing and storage performance. More specifically, the aim of this work is to evaluate these virtual machines according to a specified workload. Different from presented papers, this work proposes the evaluation of the Eucalyptus virtual machine instance types considering distinct workload demand.

This paper is structured as follows: Section 2 presents related works on performance evaluation in cloud computing environments. Section 3 introduces basic concepts on Eucalyptus platform and Performance Evaluation. Section 4 presents the adopted methodology for performance evaluation and Section 5 shows a real case study. Finally, Section 6 presents concluding remarks and presents future works.

II. RELATED WORKS

In the last few years, some works have been conducted to evaluate performance of public cloud computing infrastructures for scientific applications. Ostermann et al. [10] present a performance analysis of Amazon EC2 platform for scientific computing using benchmarks. Similarly, in [11], the authors propose a performance evaluation of four commercial cloud computing services

 ISBN: 978-1-61208-216-5

using benchmarks. These clouds computing are Amazon EC2, GoGrid (GG), Elastic Hosts (EH) and Mosso [11]. These papers compare only the performance and cost of clouds with scientific computing alternatives such as grids and parallel production infrastructures. The result is that the current cloud computing services are insufficient for scientific computing at large, but it may still be a good solution for the scientists who need resources instantly and temporarily [10][11].

Some papers focus on performance evaluation of storage resources from public and private clouds for scientific application. In [12], the Eucalyptus Platform is tested in a variety of configurations to determine its suitability for applications with high I/O performance requirements, such as the Hadoop MapReduce framework for data-intensive computing. Applications running in the Eucalyptus cloud computing framework suffer from I/O virtualization bottlenecks in KVM and Xen. The default configurations that use fully virtualized drivers and on-disk files perform poorly out of the box. Even using the best default configuration (with the Xen hypervisor), the guest domain can achieve only 51% and 77% of non-virtualized performance for storage writes and reads, respectively [12]. In [13], a public cloud platform and a private cloud platform are evaluated. The authors select the Amazon as the public cloud platform and the Magellan cloud testbed as the private cloud computing. Such a work compares the I/O performance using IOR benchmarks. The I/O performance results clearly highlight that I/O can be one of the causes for bottleneck on virtualized cloud environments. Performance in VMs is lower than on physical machines, which may be attributed to an additional level of abstraction between the VM and the hardware [13].

Differently from previous works, this paper presents the performance evaluation of different virtual machines on Eucalyptus platform, focusing on processing and storage infrastructures.

III. PRELIMINARIES

This section presents a summary of concepts for a better understanding of this work. Initially, an overview of Eucalyptus platform is provided. Finally, performance evaluation concepts are presented.

A. Eucalyptus Platform

Eucalytpus is an open-source cloud computing platform that allows the creation of private clusters in enterprise datacenters [14]. Eucalyptus provides API compatibility with the most popular commercial cloud computing infrastructure, namely, Amazon Web Services (AWS), which allows management tools to be adopted in both environments. This framework is designed for compatibility across a broad spectrum of Linux distributions (e.g., Ubuntu, RHEL, OpenSUSE) and virtualization hypervisors (e.g., KVM, Xen). Figure 1 shows the Eucalyptus architecture.

Eucalyptus system is composed of several components that interact through interfaces [14]. There are five components, each with of which its own Web-service interface, that comprise a Eucalyptus system [15]:

- **Cloud Controller (CLC).** The CLC is the entry-point into the cloud for users and administrators. It queries node managers for information about resources, performs high-level scheduling decisions, and implements them by making requests to cluster controllers.
- **Cluster Controller (CC)**. The CC acts as a gateway between the CLC and individual nodes in the data center. This component collects information on schedules and execution of virtual machine (VM) on specific node controllers, and manages the virtual instance network. The CC must be in the same Ethernet broadcast domain as the nodes it manages.
- **Node Controller (NC)**. The NC contains a pool of physical computers that provide generic computation resources to the cluster. Each of these machines contains a node controller service that is responsible for controls the execution, inspection, and terminating of virtual machine (VM) instances. This component also configures the hypervisor and host OS as directed by the CC. The node controller executes in the host domain (in KVM) or driver domain (in Xen) [9][15].
- **Storage Controller (SC)**. The SC is a put/get storage service that implements Amazon's S3 interface, providing a mechanism for storing and accessing virtual machine images and user data.

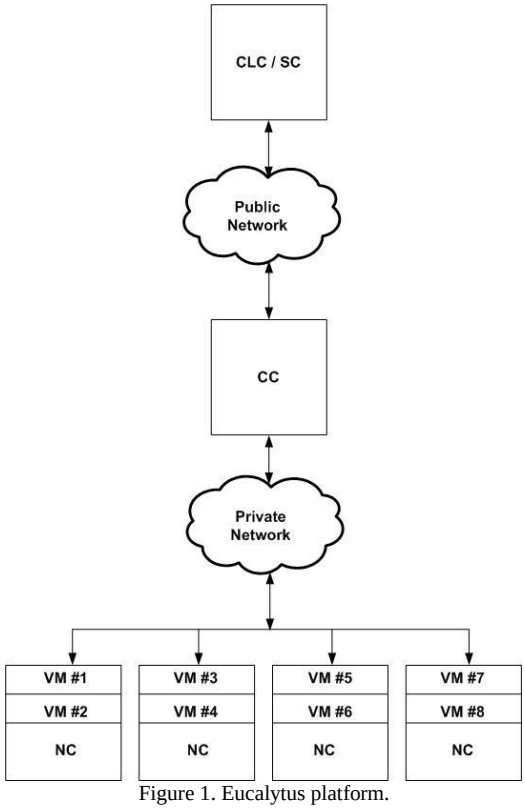


Figure 1. Eucalytus platform.

Eucalyptus platform supports different virtual machine (VM) [9][15]. The supported virtual machines and the

respective characteristics are presented in Table 1. The computational resources can be allocated to these virtual machines according to the client demand.

TABLE I. VIRTUAL MACHINE INSTANCE TYPES

VM	CPU (Core)	Memory (MB)	Disk (GB)
m1.small	1	192	2
c1.medium	1	256	5
m1.large	2	512	10
m1.xlarge	2	1024	20
c1.xlarge	4	2048	20

B. Performance Evaluation

Performance evaluation consists of a technique set classified as those based on measurement and based on modeling, which provide means for deciding about suitable configurations concerning further customers' demands, fluctuations and attaining assured service levels [16].

Modeling is usually adopted in early stages of the design process, when actual systems are not available for measurement. Measurement is utilized for understanding systems that are already built or prototyped. Measurement is an essential activity for tuning the systems, validating the performance models, and for improving the design of future systems [5][17].

A variety of different benchmark programs have been developed over the years to measure the performance of many different types of computer systems in different application domains. A natural benchmark consists of programs that mimic a real workload. Synthetic benchmark programs are artificial programs. These programs perform a mix of operations that are carefully chosen to match the relative mix of operations observed in some class of application [6][16].

Bonnie++ and LINPACK [5][16][18] are benchmarks adopted in this paper to provide workload to disk and processor components, respectively. Bonnie++ is a benchmark suite that executes a number of hard drive and file system performance tests [18]. LINPACK is a benchmark that solves dense system of linear equations in double precision and is commonly used for performance evaluation of parallel computers as a cluster. LINPACK is a benchmark that allows defining the size of the system of linear equations in order to evaluate the performance computer systems [5].

Linux monitoring tools IOstat and MPstat [19] are adopted in this work to collect professor and disk figures, respectively.

IV. PERFORMANCE EVALUATION METHODOLOGY

This section presents the methodology used for performance evaluation adopted for analyzing the Eucalyptus platform. Figure 2 shows the activity diagram of the adopted methodology.

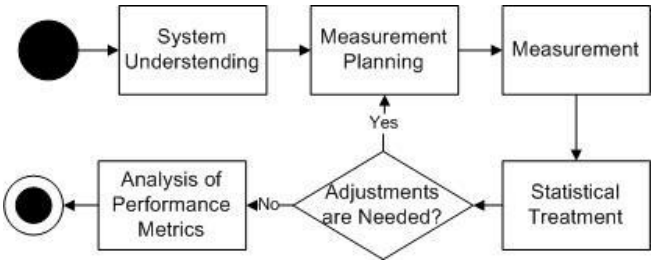


Figure 2. Methodology.

The methodology consists of five activities, which are system understanding, measurement planning, measurement, analysis of performance metrics and statistical analysis. The first activity concerns understanding the system, its components, their interfaces and interactions. This activity should provide the set of metrics that should be evaluated. Among such metrics, some might be highlighted such as utilization, average service time, average response time, average queue time, average queue length.

The second activity results in a document that describes how the measurement should be performed, the tools calibration, the frequency of data collection and how to store the measured data.

The measurement activity (see Figure 3) consists of five steps, which were implemented through a script. These steps are described below.

The first step instantiates the virtual machines. The second step starts the performance monitoring tools IOstat and MPstat on virtual machines. The third step configures the benchmarks Bonnie++ and LINPACK on virtual machines. The fourth step executes the benchmarks. These benchmarks are the workload adopted. When benchmarks execution end, in the fifth step, the monitoring processing finishes. Then, the virtual machines are shutdown. Next, logs with the measured data are created. After, the measurement process is restarted.

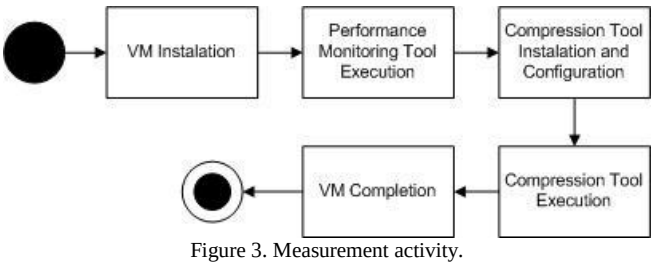


Figure 3. Measurement activity.

In each measurement process, the monitoring tools are started before the benchmarks execution. These measured data must be removed; therefore, the fourth activity analyzes the measured data.

Finally, the fifth activity applies statistical methods in measured data with the aim of providing accurate information about the evaluated system.

V. EXPERIMENTAL RESULTS

This section presents the conducted experiments to evaluate bottlenecks regarding processing and storage

 ISBN: 978-1-61208-216-5

resources running over Eucalyptus system virtual machines. This case study allows the performance evaluation of default virtual machine instances types from a private cloud configured with the Eucalyptus system. The performance of the processing and storage infrastructures is evaluated with benchmarks.

These experiments aim to evaluate the performance of cloud infrastructure presented in Figure 1 assuming a workload based on benchmarks Bonnie++ and LINPACK. These experiments consisted of some steps, which are described in Section IV, as shown in Figure 2.

In this case study, Cloud Controller (CLC), Cluster Controller (CC) and Storage Controller (SC) are running on the same computer, whereas Node Controllers (NC) are distributed on different computers. The front-end node and the back-end nodes were equipped with Intel^{R} Core^{TM} Duo CPU E6550, 2.33GHz, 2GB DDR2 RAM, 160 GB Hard Disk and 100MB Ethernet interface. These computers were configured to run the CLC, CC, SC and NC services. The front-end node was configured to run the CLC, CC and SC services. The back-end nodes were configured to run the NC service and all virtual machine images.

In this case study, the virtual machine environment is based on Eucalyptus system 2.0.0 and Ubuntu Enterprise Cloud 10.10. The Eucalyptus was configured with kernel-based virtual machines (KVM) as a hypervisor.

The virtual machine types evaluated were m1.small, c1.medium, m1.large and m1.xlarge (see TABLE I). These virtual machines have different processing and storage infrastructures, which can be allocated and released according to users demand.

The scenarios evaluated in the performance experiments are described in TABLE II. This table shows the types and number of virtual machines in each scenario analyzed. The numbers of virtual machines for each scenario varies according to the processing and storage resources of these machines and node controllers processing and storage infrastructures.

TABLE II. SCENARIOS

Scenario	VM Number	VM Types
1	8	m1.small
2	8	c1.medium
3	4	m1.large
4	4	m1.xlarge

As previously mentioned, we adopt the benchmarks LINPACK and Bonnie++ and the performance monitoring tools IOstat and MPstat in order to obtain the desired performance metrics. TABLE III shows the relationship between the benchmarks adopted to provide workload to processor and disk and monitoring tools adopted in order to provide processor and disk metrics.

TABLE III. BENCHMARK AND MONITORING TOOL

Resource	Benchmark	Tool
Disk	Bonnie++	IOstat
Processor	LINPACK	MPstat

After setting up and stabilizing the environment, measurements of performance metrics were initiated through the IOstat and MPstat. During the measurements, processes that are not strictly necessary for the experiments were removed so as to avoid interference in the collected data [5].

Measurements of processor metrics were performed for a period of 12 hours with an interval of 1 minute between data collections. The processor experiments considered the LINPACK benchmark as workload, where the number of linear equations adopted to evaluate the system was N = 1000 [5].

Measurements of disk metrics were performed for a period of 24 hours considering reading and writing tests in which files with 512MB and 1GB were considered. On the other hand, reading and writing tests considering files with 1.5GB and 2GB sizes were performed for a period of 12 hours. Each sample was collected considering an interval of 1 minute between data collections. The Bonnie++ benchmark stresses the system by performing reading and writing operations on a file system. For this experiment, files with 512MB, 1GB, 1.5GB and 2GB sizes were created for all evaluated scenarios. These files are created when the benchmark Bonnie++ is running. Furthermore, these files aim to evaluate the performance of the disk in the scenarios described in TABLE II.

These collected data were stored in logs generated through the scripts. The collected data were stored on a disk partition isolated from the measuring environment in order to prevent the measured data from being affected.

These collected data were statistically analyzed to remove possible outliers [20]. TABLE IV presents the execution times of the performance tests. The processor performance tests were performed during 12 hours and the disk performance tests occurred according to files sizes adopted in reading and writing testes.

TABLE IV. PERFORMANCE TEST

Benchmark	File Size	Execution Time (hour)
Bonnie++	512 MB	24
Bonnie++	1 GB	24
Bonnie++	1.5 GB	12
Bonnie++	2 GB	12
LINPACK	-	12

Table V shows the performance metrics evaluated in this case study.

TABLE V. PERFORMANCE METRICS

Resource	Metric
Disk	Utilization, Service Time, Response Time
Processor	Utilization

 ISBN: 978-1-61208-216-5

Figure 4 presents the processor utilization for each virtual machine (see TABLE II). This work adopts 80% as threshold [21].

Processor results reveal that the processor utilization of virtual machines exceed 80% for all scenarios [21], hence these processing infrastructures should be updated or the processor resource of the node controllers should be replaced as a preventive measure for taking into account further workload demands.

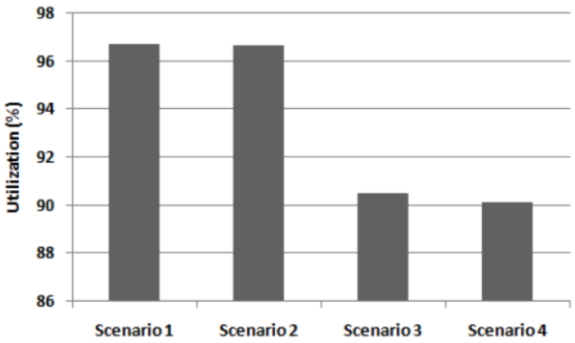


Figure 4. Processor.

Figure 5, Figure 6 and Figure 7 show the results of disk metrics which are utilization, average service time and average response time, respectively. The values of these disk metrics vary proportionally to the testing of reading and writing considering files with 512MB and 1GB sizes. These tests were performed for all scenarios evaluated (see TABLE II).

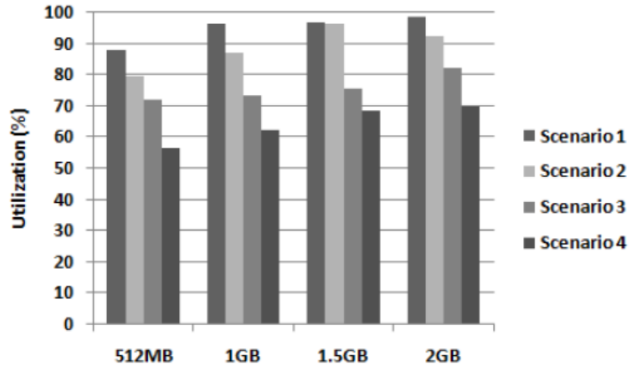


Figure 5. Utilization.

Figure 5 shows a percentage reduction in the disk utilization level when analyzing the Scenarios 1 to 4 for all reading and writing tests (files with 512MB, 1GB, 1.5GB and 2GB sizes). This metric indicates the percentage of time that the disk was used. If the disk utilization is high, the resource must be carefully evaluated. The threshold for this disk metric is close to 100%. Hence, this metric should be investigated if it is close to 100% [21].

These disk results reveal that the disk utilization level of Scenario 1 exceed 90% when considering tests of reading and writing adopting files with 1GB, 1.5GB and 2GB sizes. In a similar way, the disk utilization level of Scenario 2 exceed 90% when considering tests of reading and writing adopting files with 1.5GB and 2GB sizes. These results demonstrate that the storage infrastructure must be carefully analyzed to avoid a bottleneck and hence degradation in service levels [21].

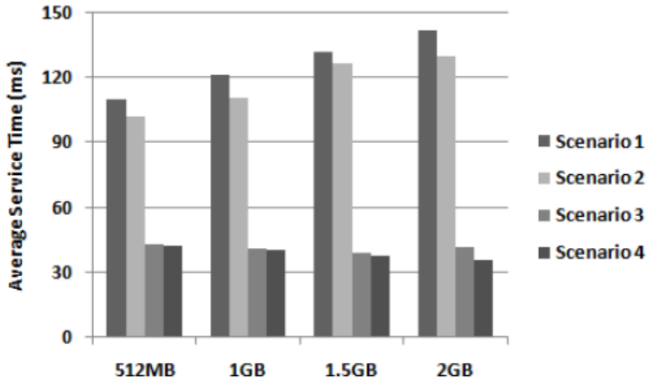


Figure 6. Average service time.

Figure 6 presents a decrease in the average service time (in milliseconds) for Scenarios 1 to 4 and all reading and writing tests performed by benchmark Bonnie++. This metric describes how long time the disk is taking to fulfill the requests. When more time is spent on fulfilling the requests, slower is the disk controller. It is recommended that the values of these metric be less than 270 ms [21]. The results show that the storage resources do not exceed the threshold of this disk metric.

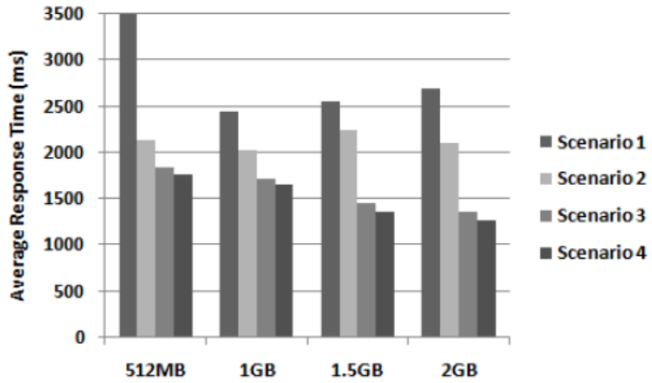


Figure 7. Average response time.

Figure 7 presents a decrease in the average response time (in milliseconds) for Scenarios 1 to 4 when considering all reading and writing tests (files with 512MB, 1GB, 1.5GB and 2GB sizes). This disk metric includes the time spent by the requests in queue and the time spent servicing them. If the average response time is high, the resource must be carefully evaluated. The threshold for this disk metric is 2.7 seconds [21]. The average response time of Scenario 1 exceed 2.7 seconds when considering tests of reading and

 ISBN: 978-1-61208-216-5

writing performed by benchmark Bonnie++ adopting files with 512MB and 2GB sizes [21].

Processing infrastructures results had lower performance, considering Scenarios 1 and 2, in relation to the others, for the same workload. Scenario 1 shown a storage with lower performance in comparison to the others scenarios, for the same workload.

This analysis permits the planning of processing and storage infrastructure of the private cloud in the enterprise.

VI. CONCLUSIONS AND FUTURE WORK

This work presented the performance evaluation of different virtual machines on Eucalyptus platform taking into account workload based on benchmarks.

This paper analyzed the performance of critical levels of virtual machine instances types on private cloud, which is configured with Eucalyptus system. This analysis permits sustain the quality of service and prevents the performance degradation related to the workload fluctuations in private clouds. In addition, this performance analysis is intended to support suitable hardware and software configurations for ensuring performance agreements of applications with high level of processing and storage requirements.

The results allowed evaluation of performance figures, such as disk response time, disk service time, disk utilization and processor utilization, for planning the Eucalyptus system processing and storage infrastructures.

Other performance issues related to Eucalyptus system can be studied and analyzed as well as other metrics than those discussed in the paper. As future work, we intend to analyze the performance of processing and storage infrastructures of virtual machine instances types from a private cloud, considering credit and debit transactions (Electronic Funds Transfer) as workload.

REFERENCES

[1] Velte, A., Velte, T., Elsenpeter, R., and Babcock, C., Cloud computing: a practical approach. McGraw-Hill, 2010.

[2] Chee, B. and Franklin Jr, C., Cloud computing: technologies and strategies of the ubiquitous data center. CRC Press, 2009.

[3] Hugos, M. H. and Hulitzky, D., Business in the Cloud: What Every Business Needs to Know About Cloud Computing. Wiley, 2010.

[4] Chorafas, D. and Francis, T., Cloud computing strategies. CRC Press, 2011.

[5] Jain, R., The art of computer systems performance analysis, vol. 182. John Wiley & Sons New York, 1991.

[6] Menasce, D. A., Almeida, V. A. F., Dowdy, L. W., and Dowdy, L., Performance by design: computer capacity planning by example. Prentice Hall, 2004.

[7] Shroff, G., Enterprise Cloud Computing: Technology, Architecture, Applications. Cambridge Univ Pr, 2010.

[8] Rosenberg, J. and Mateos, A., The cloud at your service. Manning, 2010.

[9] Johnson, D., and Murari, K., Raju, M., Suseendran, R. B., and Girikumar, Y., "Eucalyptus Beginners Guide - UEC Edition," 2010.

[10] Ostermann, S., Iosup, A., Yigitbasi, N., Prodan R., Fahringer, T., and Epema, D., "A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing," pp. 115–131, 2010.

[11] Iosup, A., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T., and Epema, D., "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," IEEE Transactions on Parallel and Distributed Systems, pp. 1–16, 2010.

[12] Shafer, J., "I/O virtualization bottlenecks in cloud computing today," in Proceedings of the 2nd conference on I/O virtualization, pp. 5–5, USENIX ssociation, 2010.

[13] Ghoshal, D., Canon, R., and Ramakrishnan, L., "Understanding I/O performance of virtualized cloud environments", The Second International Workshop on Data Intensive Computing in the Clouds (DataCloud-SC11), 2011.

[14] Amazon Web Services, "Eucalyptus open-source cloud computing infrastructure - an overview. technical report, eucalyptus, inc.," 2011.

[15] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., and Zagorodnov, D., "The eucalyptus open-source cloud-computing system," in Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 124–131, IEEE Computer Society, 2009.

[16] Lilja, D. J., Measuring computer performance: a practitioner's guide. Cambridge Univ Pr, 2005.

[17] John, L. K. and Eeckhout, L., Performance evaluation and benchmarking. CRC Press, 2006.

[18] Coker, R., "The bonnie++ benchmark," 2001. URL http://www. coker. com. au/bonnie+ Last visit in April 2, 2012.

[19] Godard, S., Sysstat:System performance tools for the Linux OS. 2004.

[20] Montgomery, D. and Runger, G., Applied statistics and probability for engineers. Wiley, 2010.

[21] Ciliendo, E. and Kunimasa, T., Linux Performance and Tuning Guidelines. ibm.com/redbooks, 2007.

 ISBN: 978-1-61208-216-5

Towards a SLA-compliant Cloud Resource Allocator for N-tier Applications

Aaron McConnell, Gerard Parr, Sally McClean, Philip Morrow and Bryan Scotney
School of Computing and Information Engineering
University of Ulster
Coleraine, Northern Ireland
Email: a.mcconnell@ulster.ac.uk, gp.parr@ulster.ac.uk, si.mclean@ulster.ac.uk, pj.morrow@ulster.ac.uk, bw.scotney@ulster.ac.uk

***Abstract*—Cloud vendors commonly offer users IaaS where virtual machines (VMs) can be created and run on cloud resources. The resource allocation for each VM is defined by the user and the VM is created on a physical machine (PM) where ample resources exist to support the VM's operation at its maximum capacity. There are a number of opportunities for improvement when allocating host resources to VMs. VM-resident applications are often n-tier, with different VMs responsible for parts of the distributed application. It may be important that these VMs are placed within a given network proximity to one another. The network proximity to the user may also be an issue for some applications. Resource allocation to VMs should also be such that, rather than a user over-provisioning the VM, the VM's minimal operational requirements are specified so that the VM can be resource-throttled at times of heavy load. This paper presents an outline for a system called Innkeeper, which aims to allocate resources to a VM in a way that ensures the VM will always function adequately, but where the VM is not over-provisioned. Innkeeper also aims to place VMs so that a VM "family" are kept within a necessary network proximity to one another and where the proximity to the user is also considered when placing VMs.**

Keywords-cloud computing; resource allocation; virtualisation.

I. INTRODUCTION

Cloud Computing offers virtualised data centre resources to users as a service. Organisations can use cloud platforms to outsource their IT infrastructure, resulting in reduced Capital Expenditure (CAPEX) and Operating Expediture (OPEX) and dynamically scaling capacity. Cloud providers offer pay-per-use pricing schemes where users pay an amount to reserve certain resources and a further amount for the amount of a resource used, e.g., bandwidth. Allocating physical host resources to users and their VMs is carried out on-the-fly and is often achieved using greedy algorithms. It is necessary for a set of operational constraints to be established before any optimisation method can be successful. PMs have finite sets of resources and VMs have operational requirements in terms of resources. A VM should only be placed on a physical host where the physical host has sufficient resources to satisfy the demands of all resident VMs. VMs also have a set of network requirements that are often ignored when defining a VM's resource needs. Cloud applications are often n-tier applications [1], with one element of the application residing within a separate VM from other elements. The network distance (in terms of bandwidth and latency) between application elements should be short enough that the entire application functions to the user's requirements. This is especially a concern in the situation where a cloud provider has resources in more than one geographic location, or where the user migrates part of his organisation's application to the cloud, whilst retaining other parts within the organisation.

The work presented in this short paper is at an early stage and is ongoing. The proposed solution provides a system, which 1) provides a Service Level Agreement (SLA) framework for n-tier cloud applications, 2) provides an automated scalable, three-tiered approach for assessing the suitability of distributed resources for VM placement, 3) considers network links between all application entities and the user.

The next section discusses related work, followed by a description of the proposed model and its design. Results from some initial experiments are presented in section IV.

II. RELATED WORK

Existing literature details that cloud computing offers different models for VM consumption in cloud environments [2], reservation, on-demand and spot market [3], with each model catering for a different need. Placing VMs in each model requires that a decision is made about placing the VM on an appropriate host [4], [5] or migrating a VM from one host to another in order to provide some kind of optimisation (e.g., performance increase, financial cost saving). The current state-of-the-art focuses heavily on optimisation of VM placement with various types of focus on different constraints; there are systems aimed at cloud resource provisioning for existing VMs [6], [7], [8], but there is no work on providing an open, scalable platform for assessing host, cluster and cloud capability, particularly for SLA-compliant n-tier VM placement.

III. MODEL DESCRIPTION

Innkeeper is designed to provide three brokers, one for each host, one for each cluster of hosts and one for the cloud of clusters (see Fig. 1). The function of each broker is as follows:

a) Host Innkeeper: Each Host Innkeeper (HIk) responds to a request for VM placement with either *accept* or *refuse* depending on whether or not the host can accommodate the VM. This decision is based upon a VM SLA, which defines

 ISBN: 978-1-61208-216-5

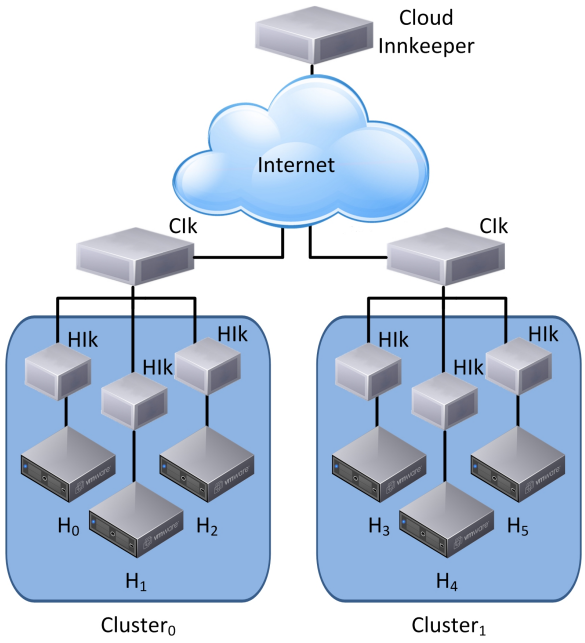


Fig. 1. The Three-tier Innkeeper structure

metric thresholds for minimal VM performance, e.g., a VM may require a minimum of 800 Mhz of CPU share and 1 GB of virtualised memory in order to perform adequately. For each metric, the HIk subtracts the SLA value from the currently available resources for the host, e.g., if the host has 4 GB of memory remaining and the VM's SLA is requesting 1 GB then the HIk would calculate that accepting the VM means 3 GB of free memory would remain for the host. The HIk would define host-level thresholds as well, meaning that it would never be the case that 100% of any host-level metric is ever consumed. In the case that some metric request in a to-be-placed VM caused the host to consume levels of that metric at host level, then the VM would not be placed and a *refuse* response would be generated.

b) Cluster Innkeeper: Each Cluster Innkeeper (CIk) acts as a placement broker at the cluster level. The CIk aims to place VMs on hosts, in the cluster it is responsible for, by interfacing with each HIk and querying whether or not the host can accept a VM with a given SLA. Three outcomes are possible when a CIk attempts to place a VM. Either one of the HIks accepts the VM, more than one HIk accepts the VM or none of the HIks accepts the VM. The first case is straight-forward and the VM is placed via the accepting HIk. In the second scenario, where there are multiple hosts on, which the VM may be placed, there is room for employing some intelligence when choosing which host to place the VM with, e.g., one host may offer more of a desired resource than another. The third scenario, where no hosts can accommodate the VM, presents a situation where either another CIk is used or an optimisation is attempted in order better place existing VMs and free up capacity so that a new VM can be added to a host. This type of optimisation is discussed in sub-section III-B. CIks also provide knowledge about the network proximity, in terms of bandwidth and latency, between CIks. This is necessary in order to ensure that VMs are placed within adequate proximity to other VMs they communicate heavily with, and with the end-user, as defined in the VM's SLA.

c) Cloud Innkeeper: The Cloud Innkeeper (CLIk) is a central system, which acts as a broker for the entire cloud. The CLIk is presented with a user's VM SLA and attempts to place the VM on a cluster by interfacing with each CIk. There are again three possible scenarios, one CIk can accommodate that VM, multiple CIks can accommodate the VM, or no CIks can accommodate the VM. The second scenario again presents an opportunity to place the VM at a host where some benefit may be had over placement at other hosts, e.g., an important resource is more abundant. The third scenario, where no CIks can accommodate the VM, presents a further opportunity to optimise the placement of existing VMs at one or more CIk. It is also possible that the CLIk will be presented with a n-tier set of VMs to place, each with constraints regarding the network proximity to others. An optimisation problem is created in this instance, which may be solved with the implementation of a greedy algorithm.

These brokers provide a highly dynamic and scalable hierarchy for SLA-compliant VM placement. A common Application Programming Interface (API) is shared between the HIk, CIk and CLIk as shown in Table I. An API relevant to network metrics is unique only to CIks, where bandwidth and latency values between a queried CIk and another address, e.g., another CIk or an end-user I.P. address, are returned.

A. Monitoring

Host and VM monitoring is carried out by accessing the monitoring Web services of the underlying virtualisation platform. This monitoring is carried out by each HIk. CIks and the CLIk access monitoring information via the HIk API. Network monitoring, of the links between CIks and with end-users, is carried out by using Bwping [9] to acquire bandwidth and monitoring statistics. Host and VM real-time monitoring statistics are stored in a database within each HIk with network monitoring statistics stored with each CIk.

B. Optimisation

Optimisation opportunities exist when a scenario occurs where a HIk, CIk or the CLIk cannot accommodate a new VM. It may be possible to free up resources at a host by attempting to reconfigure the hosting of existing VMs so that the overall capacity utilisation of a given host is higher while not breaching utilisation thresholds for any given host-level metric. This type of optimisation can be viewed as the Multi-objective Knapsack Problem (MKP), which is a combinatorial optimisation problem [10], [11], [12]. The MKP requires that a compromise or trade-off is made when considering multiple optimisational sub-objectives. Therefore MKP cannot guarantee an optimal solution for each sub-objective. Chu et Beasley [13] formulate the MKP as:

 ISBN: 978-1-61208-216-5

TABLE I
THE INNKEEPER COMMON API

Interface	Description
GetAggCPU (Return Integer)	Returns the aggregated CPU capacity for the broker's hosts (MHz)
GetAggMem (Return Integer)	Returns the aggregated memory capacity for the broker's hosts (MB)
GetAggDisk (Return Integer)	Returns the aggregated disk capacity for the broker's hosts (GB)
GetMaxCPU (Return Integer)	Returns the single largest amount of CPU available at a host (MHz)
GetMaxMem (Return Integer)	Returns the single largest amount of Memory available at a host (MB)
GetVmAllocation (Return Integer)	Returns the current allocation of VMs for the broker's hosts
CanHostVM(VMSLA *sla) (Return Boolean)	Is passed VM's SLA and returns true or false indicating the ability to host the VM
AddVM(VMID *vmId) (Return Boolean)	Is passed an ID for the VM so an attempt can be made to place it at the appropriate host
RemoveVM(VMID *vmId) (Return Boolean)	Is passed an ID for the VM so an attempt can be made to remove the VM from cloud/cluster/host

$$\text{maximise} \sum_{j=1}^{n} p_j \, x_j,$$

$$\text{subject to} \sum_{j=1}^{n} r_{ij} \, x_j \leq b_i, \quad i = 1, ..., m,$$

$$x_j \in \{0, 1\}, \quad j = 1, ..., n,$$

where each of the m constraints is defined as a *knapsack* constraint. The MKP attempts to place a subset of n items such that the total value of all items is as high as possible within given constaints. The value of each item is defined as p, with x defining whether or not item p_j is placed (x is assigned a value of 1 if the item is placed and 0 if it is not). The constraint value for p_j is r_{ij}, with i referring to a given constraint, e.g., for placing a VM it might be CPU or memory. The total constraint value for the sum of constraints for all placed items must not exceed the maximum allowed value for that constraint b_i. This problem is often solved using a greedy algorithm, e.g., a genetic algorithm.

IV. INITIAL RESULTS

Some initial experimentation was carried out in order to assess host metric utilisation as VMs are placed on a host. These experiments were carried out using a Fujitsu Siemens Celsius R550 servers, with two Intel Xenon E5440 processors (2.8 GHz, 6 MB L2 cache) and with 8 GB of main memory. This host ran VMware ESX 4.0 [14] and was used to host VMs, containing a standard Joomla v1.5.20 Web Server [15], each with a resource allocation of 2 virtual CPUs, 512 MB of main memory and an 8 GB thin-provisioned virtual hard drive. Load was placed on each VM's application (two page requests per second) using OpenLoad [16] on the Web Server over port 80 (the standard HTTP port). Fig. 2 illustrates the resource usage for both the host and the first VM placed. The host's CPU runs at 100% utilisation when four loaded VMs (each one had a VM CPU load of $\approx 95\%$) are placed on it. This host CPU load causes resource contention between VMs. It is interesting to note that, after 3 heavily-loaded VMs are placed on the host, the CPU resource consumption of the monitored VM appears to drop. This decrease in CPU consumption by the VM is because it is starved of access to the underlying

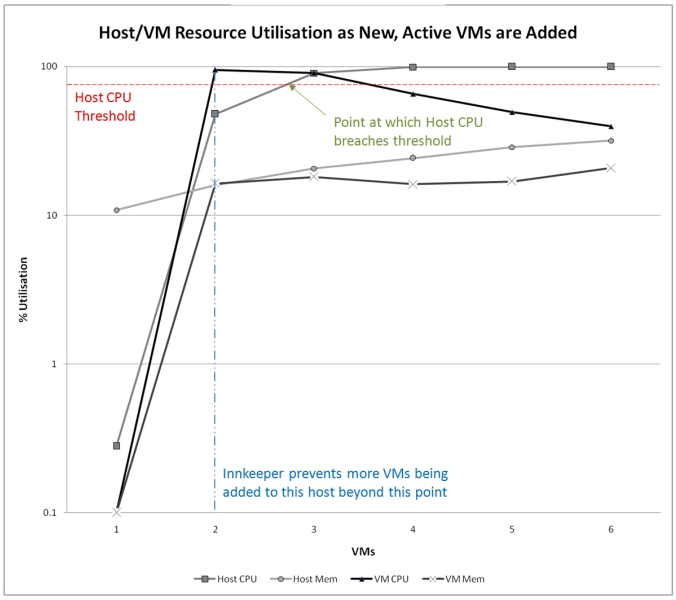


Fig. 2. Host and VM Resource Consumption

host resources and is forced to queue for CPU resources. This creates the illusion, in the reported VM performance metrics, that the VM is not consuming resources due to lack of demand on the VM when in fact the drop in CPU consumption is because the VM does not have the opportunity to consume the host's CPU. This is backed up the graph in Fig. 3 where OpenLoad reports an increasing response time, as the number of VMs on the host is increased, for the monitored VM. The horizontal dashed line on Fig. 2 illustrates a potential host CPU utilisation threshold beyond which no further VMs should be added. With this threshold defined within a HIk, the HIk would prevent further VMs being added beyond the second VM, as illustrated by the vertical dashed line in Fig. 2.

V. CONCLUSIONS AND FUTURE WORK

This model attempts to provide a system for SLA-compliant placement of n-tier VMs in a cloud computing environment. Initial experimentation illustrates a need for such a system, to ensure near-optimal use of distributed cloud resources while enforcing SLA constraints. There is also a need to ensure

 ISBN: 978-1-61208-216-5

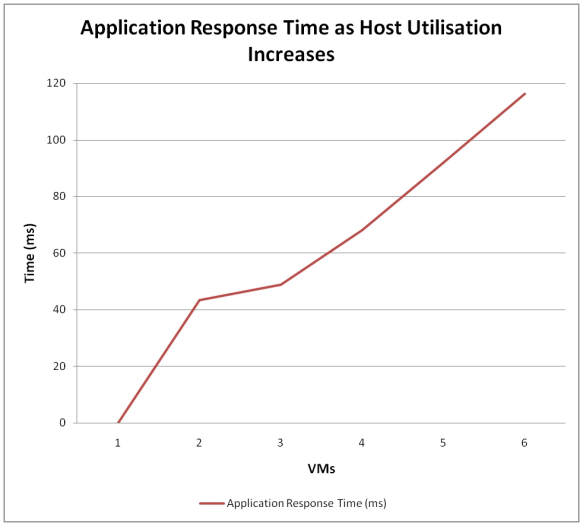


Fig. 3. VM Response Times

that communicating VMs are placed within a relatively close network proximity, in order that they perform adequately. A key requirement for cloud-based systems is the ability to provide scalability. The Innkeeper design offers this scalability and ensures that each of the three tiers is agnostic to the function of the other tiers. This design reduces complexity and provides a relatively simple but powerful means by which hosts can be monitored and VMs placed. One of the key motivators for this work is to create a platform for cloud monitoring and orchestration where intelligent optimisation, with various focuses, e.g., power-saving, reduction in network load, can be easily implemented.

A. Future Work

This body of work presents challenges and opportunities, all of which will be explored in the near future so that a live alpha prototype system is in place before the end of the calendar year. The impending implementation of the prototype system requires that the means of monitoring live VMs, hosts and network links are in place within a cloud test-bed environment. This will most likely be carried out on a VMware test-bed with a number of clusters of industry-standard, multi-core hosts. Network degradation will be created between these clusters (and between clusters and emulated end-users) using WANem [17]. A standardised means of n-tier VM SLA definition remains a challenge, with an associated problem of optimally placing multiple, communicating VMs. An associated issue with n-tier VM placement is that heavy VM load, on existing VMs, may force SLA failure despite the SLA metric thresholds being set for a given amount of maximum load. Failure to ensure that a VM's load does not exceed that for which its SLA is defined may result in the HIk's resource provisioning calculations becoming pointless. A methodology must be developed to ensure that a VM is resource-throttled when its load causes it to consume resources to the extent that resource contention is caused for other VMs on the same host. The other perspective is that this throttling shouldn't be a constant - if host resources are unused then they should be available to VMs, in the hope of increasing the quality of user experience, rather than the host experiencing low utilisation. However, optimisation opportunities exist to place VMs such that, with VM resource-throttling in place, low utilisation on hosts does not occur.

REFERENCES

[1] P. Xiong, Z. Wang, S. Malkowski, Q. Wang, D. Jayasinghe, and C. Pu, "Economical and robust provisioning of n-tier cloud workloads: A multi-level control approach," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, June 2011, pp. 571 –580.

[2] K. Mills, J. Filliben, and C. Dabrowski, "Comparing vm-placement algorithms for on-demand clouds," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 29 2011-Dec. 1 2011, pp. 91 –98.

[3] I. Fujiwara, K. Aida, and I. Ono, "Applying double-sided combinational auctions to resource allocation in cloud computing," in *Applications and the Internet (SAINT), 2010 10th IEEE/IPSJ International Symposium on*, July 2010, pp. 7 –14.

[4] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM, 2010 Proceedings IEEE*, March 2010, pp. 1 –9.

[5] J. Xu and J. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, Dec. 2010, pp. 179 –188.

[6] N. Bonvin, T. G. Papaioannou, and K. Aberer, "Autonomic SLA-driven Provisioning for Cloud Applications," *2011 11th IEEEACM International Symposium on Cluster Cloud and Grid Computing*, pp. 434–443, 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5948634

[7] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments," *2011 International Conference on Parallel Processing*, pp. 295–304, 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6047198

[8] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 871–879, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.future.2010.10.016

[9] O. Derevenetz, "Bwping - open source bandwidth measurement tool," http://bwping.sourceforge.net/, Jun. 2011, [retrieved: April 2012].

[10] C. Cheng, Y. Huang, Z. Chen, X. Zhang, and J. Xu, "Solving the 0-1 multi-objective knapsack problem using self-assembly of dna tiles," in *Bio-Inspired Computing, 2009. BIC-TA '09. Fourth International Conference on*, Oct. 2009, pp. 1 –9.

[11] Z. Shurong, W. Jihai, and Z. Hongwei, "Multi-population cooperative ga and multi-objective knapsack problem," in *Management and Service Science (MASS), 2010 International Conference on*, Aug. 2010, pp. 1 –4.

[12] D. Vianna and J. Arroyo, "A grasp algorithm for the multi-objective knapsack problem," in *Computer Science Society, 2004. SCCC 2004. 24th International Conference of the Chilean*, Nov. 2004, pp. 69 – 75.

[13] P. C. Chu and J. E. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 4, no. 1, pp. 63–86, Jun. 1998. [Online]. Available: http://dx.doi.org/10.1023/A:1009642405419

[14] Vmware, "Virtualization overview," whitepaper, http://www.vmware.com/pdf/virtualization.pdf, Jan. 2010, [retrieved: April 2012].

[15] Turnkey-Linux, "Joomla 1.5 appliance - cutting edge content management — turnkey linux virtual appliance library," online, http://www.turnkeylinux.org/joomla15, Nov. 2011, [retrieved: April 2012].

[16] P. Johnsen, "Openload," online, http://freecode.com/projects/openload, Jun. 2001, [retrieved: April 2012].

[17] WANem, "Wanem - the wide area network emulator," online, http://wanem.sourceforge.net, Dec. 2009, [retrieved: April 2012].

 ISBN: 978-1-61208-216-5

Simulation-based Evaluation of an Intercloud Service Broker

Foued Jrad, Jie Tao and Achim Streit
Steinbuch Centre for Computing, SCC
Karlsruhe Institute of Technology, KIT
Karlsruhe, Germany
{foued.jrad, jie.tao, achim.streit}@kit.edu

***Abstract*—The lack of common standards in a fast emerging Cloud computing market over the last years resulted in "vendor lock in" and interoperability issues across heterogeneous Cloud platforms. Therefore, the Cloud user is facing now a challenging problem of selecting the Cloud provider that fits his needs. A new promising research approach is the use of intermediate broker services to assist the user in finding the appropriate Cloud resources that satisfy his requirements. In this paper, we present a generic simulation framework based on the CloudSim toolkit for the validation and evaluation of a Cloud service broker deployed on an Intercloud environment. A unique feature of the framework is the integration of several state of the art technologies and standards, which makes it easy to deploy on real production Clouds. After presenting the framework fundamental architecture, we discuss in detail the solved implementation challenges. Finally, we present some initial evaluation results.**

Keywords—Cloud Brokering, Intercloud Computing, Simulation Environment, Cloud Interoperability, ClouSim Toolkit.

I. INTRODUCTION

The on-demand delivery of Cloud computing services over the Internet is now a needed reality rather than only a new marketing hype. Due to the fast emerging Cloud computing market over the last years, the number of Cloud service providers has significantly increased. On the other hand, "vendor lock in" issues and the lack of common Cloud standards hindered the interoperability across these providers. Thus, today the Cloud customer is facing a challenging problem of selecting the appropriate Cloud offers that fit his needs. Therefore, standardized interfaces and intermediate services are needed to prevent monopolies of single Cloud providers.

One of the promising use cases of the Intercloud vision defined in [1] is market transactions via brokers. In such a use case, a broker entity acts as a mediator between the Cloud consumer and multiple interoperable Cloud providers to support the former in selecting the provider that better meets his requirements. Another value-added broker service is the easy deployment and management of the user's service regardless of the selected provider through a uniform interface.

The lack of standardization across Cloud providers makes the deployment of Cloud service brokers on real production Clouds a challenging task for Cloud developers and researchers. Amongst others, many vendor compatible adapters are needed by the broker to interface the heterogeneous Cloud platforms. Furthermore, the evaluation of the broker using a real testbed is usually cost- and time-consuming, as a large number of Cloud resources is required to achieve realistic results. A more promising and cost-saving approach for the broker evaluation is the use of simulation environments.

Motivated by the above considerations, we present in this paper an extensible simulation-based framework to evaluate Cloud service brokers. The contribution from the developed framework is threefold: (1) It implements a Cloud service broker featuring automatic Service Level Agreement (SLA) negotiation and service deployment; (2) It enables through a standardized abstraction layer the monitoring and management of services deployed on heterogeneous Cloud providers while hiding their technical details; (3) It allows the easy integration and evaluation of custom resource matching policies.

The remainder of the paper is organized as follows: In the next section, we discuss prior works related to Cloud service brokering frameworks. We also identify how our work differs from related work. This is followed by the framework fundamental architecture in Section III. The simulation environment details are discussed in Section IV. In Section V and VI, we present and discuss initial evaluation results, respectively. Finally, we conclude the paper in Section VII with a brief summary and describe our future research directions.

II. RELATED WORK

The idea of service brokering in Cloud is currently a subject for many research works.

A well-known research project is the Cloudbus toolkit [2] that defines a complete architecture for market-oriented Cloud computing. The three key components of this architecture are a Cloud Broker, a Market Maker and an InterCloud [3]. The Cloud Broker schedules applications on behalf of the user by specifying the desired Quality of Service (QoS) requirements, whereas the Market-Maker acts as a mediator bringing together Cloud providers and customers. It aggregates infrastructure demands from the Cloud Broker and matches them against the available resources published by the Cloud providers. The InterCloud provides a scalable federated computing environment

 ISBN: 978-1-61208-216-5

composed of heterogeneous interconnected Clouds enabling the Intercloud resource sharing.

The above envisioned architectural framework is still under development. However, first experimental results with Aneka [4] and Amazon EC2 [5] based Clouds demonstrated that the market-oriented Cloudbus architecture brings benefits to user's application performance in optimizing the cost and execution time.

CloudAnalyst [6] is a graphical simulation tool built on top of the CloudSim toolkit [7], developed by the Cloud Computing and Distributed Systems (CLOUDS) laboratory at university of Melbourne to model and analyze the behavior of large social network applications. The Internet traffic routing between the user bases located in different geographic locations and the datacenters, is controlled in CloudAnalyst by a service broker that decides which datacenter should serve the requests from each user base based on different routing policies. The current version of CloudAnalyst implements three different routing policies, which are network-latency-based routing, response-time-based routing and dynamic-load-based routing. A CloudAnalyst simulation case study of the social network application Facebook [8] proved how load balancing managed by a service broker optimizes the performance and cost of large scale Cloud applications.

The EU funded OPTIMIS [9] project drives the development of a toolkit to optimize the full service lifecycle in the Cloud. Its proposed flexible multi-Cloud architecture includes a service broker that allows a decision making taking into account of many business aspects like trust, cost and risk. Although the toolkit is still not implemented, the conducted simulation experiments with real workload traces prove the benefits from the use of cost and risk aspects as elasticity policies in the decision making.

The work in [10] proposed an SLA-based Service Virtualization (SSV) architecture, which is built on three main components: a Meta-Negotiator responsible for agreement negotiations, a Meta-Broker for selecting the proper execution environment and an Automatic Service Deployer for service virtualization and on-demand deployment. The proposed service virtualization architecture has been validated in a simulation environment based on CloudSim using a real biochemical application as a case study. The simulation results showed the performance gains in terms of execution time from the SSV architecture compared to a less heterogeneous Grid meta-brokering solution.

Comparing the previous mentioned service brokering approaches, their implementation on real production Clouds is still ongoing and their current validations and evaluations are mostly based on simulation methodologies. The presented Cloud service broker framework in this paper is also implemented based on a simulation approach. However, its high-level generic architecture combined with the integration of state of the art Cloud technologies and standards prepares a realistic testbed for developers and researchers to easily test and evaluate service brokers before their deployment on real production Clouds. Moreover, the framework implements all the value-added broker services included in previous solutions like SLA negotiation, match making, service deployment and monitoring.

III. FRAMEWORK ARCHITECTURE

As shown in Figure 1, the framework architecture is composed of three main parts: the Client, the Cloud Service Broker and the Cloud provider Intercloud Gateway. The internal components of every architecture part and their provided functionalities are discussed in the following subsections.

A. Client

The Client provides Cloud users with an interactive user interface to submit their service requests to the broker by describing the functional and non-functional service requirements. Moreover, the user is able to manage and monitor the service after its deployment through a single management console. If the requested service requires the involvement of other services, a workflow engine could be deployed to assist users in building and executing complex Cloud services.

B. Cloud Service Broker

The Cloud Service Broker builds the heart part of our implemented framework by offering attractive value-added services to users. Its main task is to find the most suitable Cloud provider while satisfying the users' service requirements in terms of functional and non-functional Service Level Agreement parameters. Additionally, its high-level architecture design allows the deployment and monitoring of services on top of heterogeneous Cloud providers. More detailed descriptions on the internal broker design can be read in [11].

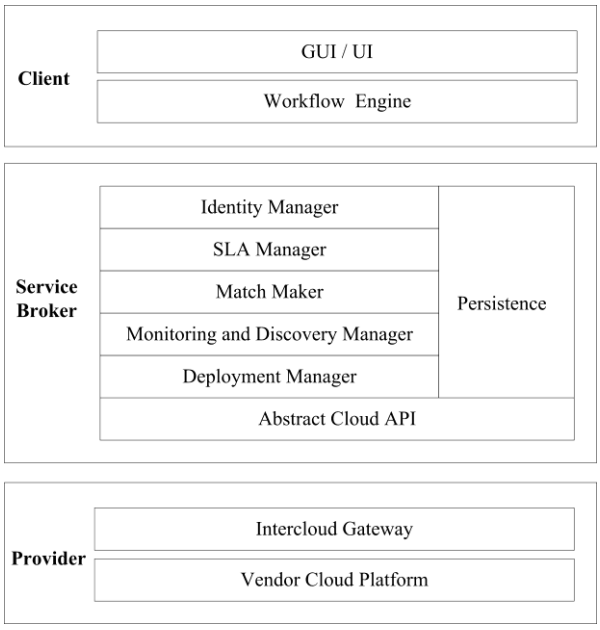


Figure 1. Framework architecture.

 ISBN: 978-1-61208-216-5

The different components of the broker and their roles are briefly described below:

- Identity Manager: It handles the user authentication and admission control.
- The SLA manager: It negotiates the SLA creation and handles the SLA provisioning.
- Monitoring and Discovery Manager: It queries resource information and monitors the SLA metrics.
- Match Maker: It selects the best Cloud providers for user requests using different matching algorithms.
- Deployment Manager: It deploys the service on the selected provider.
- Persistence: It stores broker specific data (e.g., monitoring, SLA templates and resources data).
- Abstract Cloud API: A standard abstract API used to manage Cloud resources on different Cloud providers.

C. Provider Intercloud Gateways

The Intercloud Gateway is the key component of our framework hosted on the provider side to interface the vendor Cloud platform. It acts as a standardized service frontend for the Cloud provider and adds the needed abstraction layer to interact with the broker. Its main role is to provide the broker with common management and monitoring interfaces while hiding the internal provider policies.

IV. SIMULATION-BASED IMPLEMENTATION

We implemented a simulation environment for the framework presented in the previous section. This allows us to validate and evaluate a Cloud service broker without the setup of a testbed with real Cloud providers, which is extremely time- and cost-consuming. The implementation details and the simulation flow are described in the next subsections.

A. Simulation Environment

The simulation environment for the Cloud service broker framework built on top of the CloudSim 2.2.1 simulation toolkit is depicted in Figure 2. In the following subsections we go through all the implemented components by describing the used technologies and tools.

1) CloudSim Toolkit

CloudSim is a scalable open source simulation tool offering features like support for modeling and simulation of large scale Cloud computing infrastructures including datacenters, brokers, hosts and virtual machines (VMs) on a single host. In addition, the support for custom developed scheduling and allocation policies in the simulation made CloudSim an attractive tool for Cloud researchers. Additional information about CloudSim can be found in [12].

In our simulation environment CloudSim is used to model large scale and heterogeneous Cloud providers. This allows us, for the purpose of evaluation, to easily configure the amount of Cloud provider resources accessible by the broker. However, some CloudSim extensions are needed to

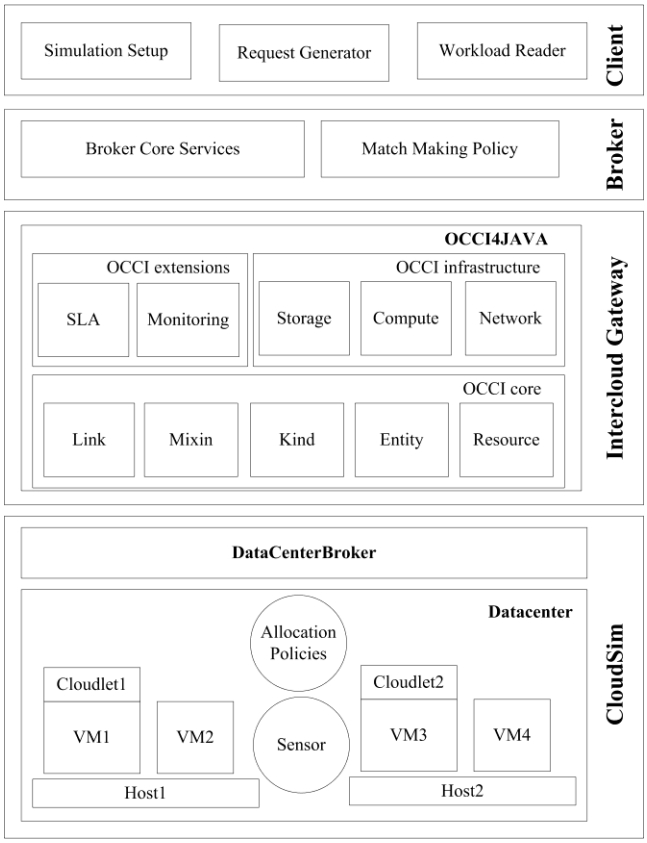


Figure 2. Simulation environment.

allow the dynamic creation, destroying and monitoring of the VMs during simulation runtime and therefore to enable the automatic service deployment in the broker.

2) Cloud Service Broker Implementation

Until the writing of this paper, most core broker services including the Deployment Manager, the Match Maker and the Monitoring Manager have been fully implemented. The SLA Manager is currently under development. Furthermore, two persistence classes named ServiceRegistry and ProviderRegistry are used to store and query all the service and provider data during the simulation.

While looking for an abstract Cloud API to access different Cloud platforms, we found that the Open Cloud Computing Interface specification (OCCI) [13] is the most suitable for our framework. OCCI is an extensible specification for remote management of Cloud infrastructures, allowing the development of interoperable tasks over heterogeneous Clouds. The current OCCI specification, focusing on IaaS Cloud provisioning, defines three abstract resource types, which are compute, storage and network. All the operations on resources can be requested on a REST manner over HTTP methods (GET, POST, PUT and DELETE). The use of OCCI as abstract Cloud API allows the broker to act as OCCI client against the Intercloud Gateway, which runs as OCCI-server on the provider side.

The implemented Match Maker functionality of the broker is extensible enough to permit the easy integration of custom resource matching policies. In order to demonstrate

 ISBN: 978-1-61208-216-5

this feature, we implemented the following primitive match making policies:

- RandomCloudMatcher: It selects randomly a provider regardless of the users' service requirements.
- FunctinalSLACloudMatcher: It selects the provider that fits all the functional SLA service requirements.
- LocationAwareCloudMatcher: It selects a provider located at the same region given in the service request.
- CostAwareCloudMatcher: It selects the cheapest provider below a given cost limit.
- HybridCloudMatcher: It combines both functional SLA and location-aware matching.

3) OCCI-based Intercloud Gateway

In order to simulate the Intercloud Gateway component serving as standard service frontend for Cloud providers, we implemented, based on the open source Java implementation for OCCI called OCCI4JAVA [14], an OCCI frontend for CloudSim. In this way, the entire communication between broker and providers is forwarded to the native CloudSim DatacenterBroker class through standard OCCI-interfaces. In contrast to the OCCI specification, as CloudSim simulations usually run on one host, the broker communicates with the Intercloud Gateway through simple Java object calls instead of using the defined REST-like methods. Furthermore, we extended OCCI4JAVA with an OCCI monitoring mixin to allow the broker to query resource properties like datacenter static information (e.g., location, supported OS, CPU architecture) and current monitoring metrics values from CloudSim.

4) Request Generator

The simulation-based evaluation of the broker requires the submission of real world service requests by the client to achieve valuable evaluation results. Thus, we implemented a service Request Generator helper class that continuously generates VM provisioning requests similar to real Amazon EC2 compute instances at a configurable rate. Some sample VM requests are provided in the next section.

B. Simulation Flow

The needed simulation flow to process the incoming client service requests to the service broker is illustrated by the flow diagram in Figure 3.

The simulation is done as follows: In a first step, CloudSim is initialized according to the desired simulation scenario. Then, the Request Generator starts to generate continuously VM provisioning service requests with a variable request arrival rate. All the request and provider data are maintained in the corresponding ServiceRegistry and ProviderRegistry classes during the simulation. The broker, after receiving the request, asks the Match Maker, if the service can be deployed with the specified requirements. For this, the Match Maker starts a match making process to find the best suitable provider by matching the gathered resource information from the Monitoring Manager with the service requirements and by applying the pre-configured matching algorithms. Upon the existence of a match, the service is automatically deployed and the requested VM is created and started on the selected CloudSim datacenter with the modeled workload traffic (Cloudlet). During the execution time, the VM status is queried periodically by the Monitoring Manager until the VM is destroyed. If none of the providers can be matched, the request is discarded by the broker.

All the aforementioned simulation steps are repeated until reaching the preset maximum number of requests or simulation time limit. In this case, the simulation is terminated and the output results are displayed in the Client.

V. EVALUATION RESULTS

In this section we discuss first evaluation results acquired using the previous implemented simulation framework. We describe in the following subsections the experimental setup and then present the evaluation results.

A. Experiemental Setup

In order to model heterogeneous Cloud providers, we configured six heterogeneous CloudSim datacenters. Each datacenter has a unique identifier (ID) and is located into a different geographical zone. As shown in Table I, we define six different compute zones presenting the six world continents. Each zone has been given a unique code. The detailed configuration for each datacenter is gathered in Table II. The six datacenters have different pricing policies and can support one of two defined operating systems (Linux or Windows) and CPU architectures (x86 or x64). Furthermore, each datacenter is made up of 50 hosts, which are equally divided between two different host types. As can be seen in Table III, the used hosts' setup allows at least the

TABLE I. COMPUTE ZONES

Zone	***North America***	***South America***	***Europe***	***Asia***	***Africa***	***Austr-alia***
Code	0	1	2	3	4	5

TABLE II. DATACENTERS CONFIGURATION

Name	**Datacenter Configuration**				
	ID	***OS***	***Arch***	***Region Code***	***Cost $/hour***
Provider_A	0	Linux	x64	0	0.3
Provider_B	1000	Linux	x64	0	0.45
Provider_C	2000	Windows	x64	2	0.75
Provider_D	3000	Linux	x64	2	0.55
Provider_E	4000	Linux	x64	3	0.15
Provider_F	5000	Windows	x86	5	0.04

TABLE III. HOSTS SETUP

Host Type	**Host Configuration**				
	CPU MHZ	***Cores number***	***RAM GB***	***Bandwidth Gbit/s***	***Storage TB***
Xeon 3040	1860	2	4	1	1
Xeon 3075	2260	2	8	1	1

 ISBN: 978-1-61208-216-5

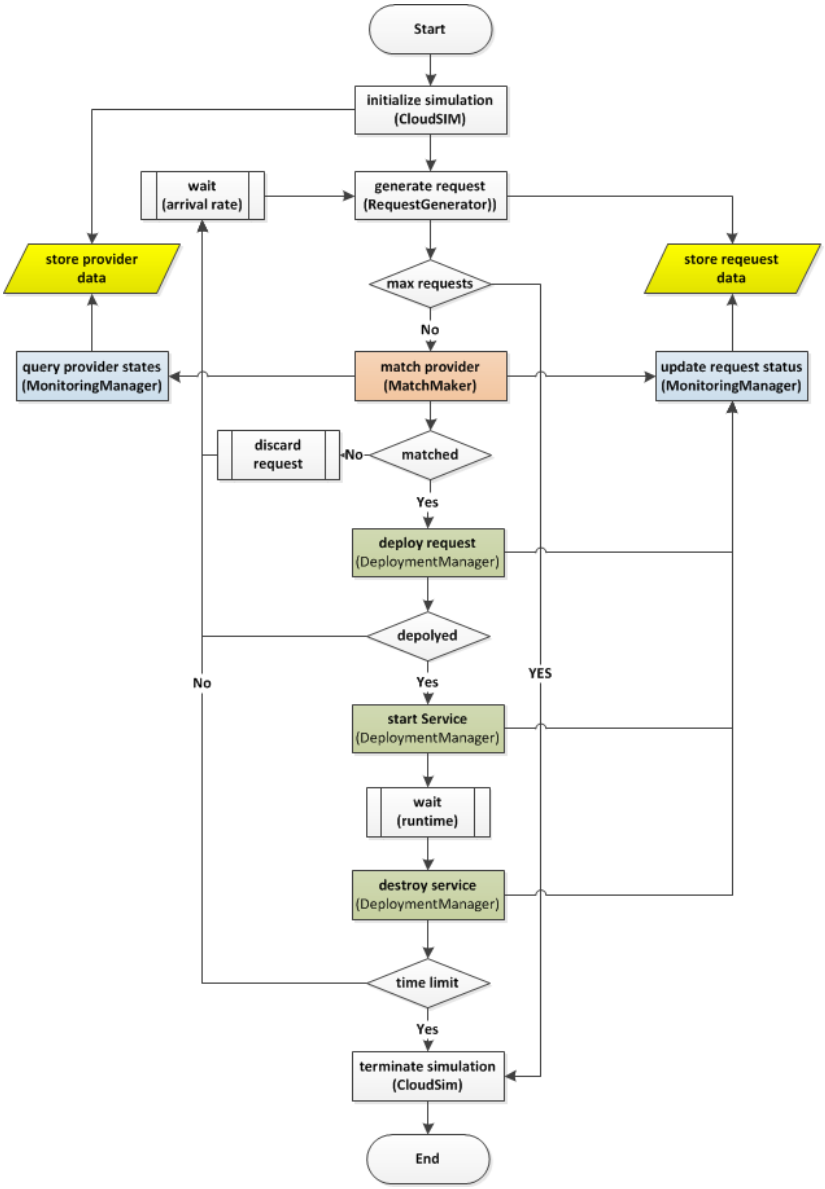


Figure 3. Simulation flow.

deployment of one VM instance per host.

All the experiments are done on a notebook with CPU Intel Core i5 560M 2.67 GHZ, RAM 4 GB and using Windows 7 operating system. The default CloudSim simple VM provisioning policy is used as internal datacenter scheduling policy. This policy allocates VMs to the host with most free cores. In order to permit the dynamic sharing of CPU cores among VMs, we configured CloudSim to use a time-shared VM scheduler policy.

B. Initial Results

We conducted a first experiment to evaluate the broker scalability. We continuously generate VM service requests (at a random rate varying from 0 to 60 seconds) and let the broker randomly select a provider from the six datacenters and then deploy the VM on it. The generated VM requests are equally distributed between four Amazon EC2 instance types and require Linux as operating system and x64 CPUs. Table IV gives the specific requirements of each VM type.

TABLE IV. VM REQUEST TYPES

VM Type	Host Configuration				
	CPU GHZ	*Cores number*	*RAM GB*	*Region Code*	*Cost $/hour*
CPU high	2.5	2	1.7	0	0.17
large	2	2	7.5	2	0.34
small	1	1	1.7	3	0.085
micro	0.5	1	0.63	4	0.02

 ISBN: 978-1-61208-216-5

TABLE V. BROKER DEPLOYMENT PERFORMANCE (%)

Provider number	Requests number					
	50	*250*	*500*	*1000*	*1500*	*2000*
1	98	78	44.4	28	22.1	17.8
3	100	81.6	78.6	61,8	44	36.3
6	100	98.3	82.8	78.6	77.2	62.1

While maintaining the number of datacenters constant, we measured the deployment rate, which is defined as the percentage of successfully deployed VMs, by varying the request number from 50 to 2000. We repeated the same experiment by decreasing the number of datacenters from six to three and then to only one. The results presented in Table V after one day simulation time, show that the broker deployment rate scales well with the increasing number of service requests and Cloud providers.

We conducted another experiment to evaluate the match performance of the four implemented primitive matching policies. We repeated the previous experiment using all six datacenters and by changing each time the matching policy and then we measured the match rate, defined as the percentage of successfully matched requests.

As depicted in Figure 4, when using cost or location as matching policy the match rate remains constant at 75 %, as the requested cost limit and location for the micro VM instance type usually has no match. However, with the functional SLA and hybrid matching policies the match rate decreases continuously with the rising service demand due to the limited capacity of the provided datacenter resources.

VI. DISCUSSION

The previous experiments show that an increase of the number of concurrent providers results in more resource heterogeneity and therefore improves the broker match rate. Furthermore, the results prove that the accuracy of the queried monitoring information by the Monitoring Manager heavily impacts the performance of the matching policy, especially for the functional SLA matching.

In fact, the support of more than one SLA parameter in the matching increases the customer satisfaction, but at the cost of a low match rate. Thus, the matching algorithm should optimize this trade-off by modeling the dependency between the customer utility function and his requested functional and non-functional SLA parameters, while considering the current provider monitoring information.

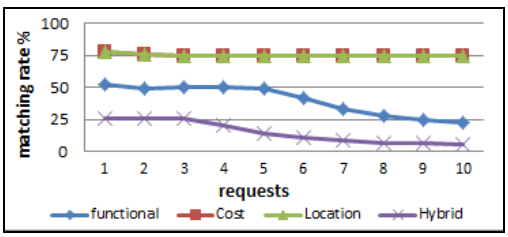


Figure 4. Broker matching performance.

VII. CONCLUSIONS AND FUTURE WORKS

The deployment and evaluation of intermediate broker services on production Clouds is today a challenging task due to the lack of interoperability and the heterogeneity in current Cloud platforms.

In this paper, we described the fundamental architecture and the implementation details of a simulation-based framework used to evaluate a Cloud service broker. We presented also the first simulation results in evaluating the broker scalability and match making policies.

In our future work, we will use the simulation framework to investigate and evaluate more complex SLA-aware match making algorithms to improve the broker matching performance. Furthermore, we will investigate the use of real workload traces instead of using generated requests to get more realistic results.

REFERENCES

[1] Golobal Intercloud Technology Forum GICTF, "Use Cases and Functional Requirements for Inter-Cloud Computing," White paper, August 2010.

[2] R. Buyya, S. Pandey, and C. Vecchiola, "Market-Oriented Cloud Computing and the Cloudbus Toolkit," in Large Scale Network-centric Computing Systems, March 2012, in press.

[3] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services," in ICA3PP 2010, 10th International Conference on Algorithms and Architectures for Parallel Processing, pp. 13–31, 2010.

[4] "Aneka enterprise Cloud platform," [online], March 2012, http://www.manjrasoft.com.

[5] "Amazon Elastic Compute cloud EC2," [online], March 2012, http://aws.amazon.com/ec2.

[6] B. Wickremasinghe, R. N. Calheiros and R. Buyya, "CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications," in AINA 2010, 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446–452, April 2010.

[7] "CloudSim Toolkit 2.1.1," CLOUDS Lab, Unversity of Melbourne, [online], http://www.cloudbus.org/cloudsim/.

[8] Facebook, [online], March 2012, http://www.facebook.com.

[9] A. J. Ferrer, et al., "OPTIMIS: A Holistic Approach to Cloud Service Provisioning," in Future Generation Computer Systems, vol. 28, pp. 66–77, January 2012.

[10] A. Kertesz, G. Kecskemeti and I. Brandic, "Autonomic SLA-aware Service Virtualization for Distributed Systems," in PDP2011, 19th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp. 503–510, February 2011.

[11] F. Jrad, J. Tao and A. Streit, "SLA Based Service Brokering in Intercloud Environments," in CLOSER 2012, 2nd International Conference on Cloud Computing and Services Science., pp. 76–81, April 2012.

[12] R.N. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose and R. Buyya, "CloudSim: a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," in Journal of Sotware: Practice in Experience, vol. 41, pp. 23–50, January 2011.

[13] "Open Cloud Computing Interface specification OCCI," OCCI-WG, [online], March 2012, http://www.occi-wg.org.

[14] "OCCI4JAVA JAVA-based OCCI Implementation," [online], March 2012, https://github.com/occi4java.

 ISBN: 978-1-61208-216-5

A Study of Cloud Mobility in a Mobile Cloud Network based on Future Internet Approach

Dongha Kim, Hyunjun Kim, Gijeong Kim, Sungwon Lee
Department of Computer Engineering
Kyung Hee University
Young-in, South Korea
{dongha, kimhyunjun, kimgijeong, drsungwon}@khu.ac.kr

***Abstract*— In this paper, we extend the limited functionality of the GENI Cloud project as follows. First, we use the OpenStack cloud platform instead of the Eucalyptus cloud platform used in the GENI Cloud. Second, we develop the FiRST Cloud aggregate manager (AM) based on GENI AM Application Programming Interface (API) for the federation between a future internet test-bed and the OpenStack cloud platform. Third, we develop a Cloud Mobility Server and Client for mobile cloud management in order to control the zero-client service. Thus, we confirm that the proposed FiRST Cloud AM is feasible through zero-client mobile cloud service.**

Keywords-cloud; mobile cloud; cloud mobility; future internet; FiRSTCloud AM.

I. INTRODUCTION

Since 1974, when the Internet was first proposed, the Internet has become a global network. Since 2000, however, rapid change of communication environments and various user requirements trigger numerous researches for Future Internet to overcome conventional Internet's problems [1].

A new trend of these Future Internet research is harmonization between the conventional Future Internet and cloud computing. Disadvantages of current cloud computing include limited bandwidth and highly variable latency due to the conventional Internet limitations is able to complemented by Future Internet concept [1].

Cloud computing includes aspects such as grid computing, utility computing, thin-client based computing. Cloud computing requirements for client hardware and software are continually being simplified. Mobile handheld devices are able to take special advantage of these simplified requirements.

An optimized, robust network is needed for cloud computing, along with an optimized protocol for communication between the client device and the cloud server. Researchers are studying projects which use cloud computing in large-scale global test-beds.

In this paper, we develop zero-client based mobile cloud service with open-source cloud platform, Openstack [8]. Furthermore, to improve this service, we develop cloud mobility server, client and FiRST cloud Aggregate Manager (AM).

First, cloud mobility server and client support a client for receives service not from 'local site' but from 'remote site'. Local site is an original server that has client's data and remote site is a closest cloud server to the client.

Second, FiRST Cloud AM is an API based on GENI AM API to interwork with Future Internet test-bed.

As a result, we develop a Cloud Mobility server and client for mobile cloud management to support the zero-client cloud service and to confirm the feasibility of the proposed FiRST Cloud AM with a zero-client mobile cloud service.

This paper organized as follows: related works are introduced in following section. After that section, detailed information about proposed cloud mobility and interaction with future Internet are described. Next section presents result data of performance evaluation. Finally, conclusion and future work is presented

II. RELATED WORK

A. Future Internet Test-bed: GENI

The Global Environment for Network Innovations (GENI) [3] is designed to support experimental research in network science and engineering. This research challenges us to understand networks broadly and at multiple layers of abstraction, from physical substrates through architecture and protocols to networks of people, organizations, and societies. The intellectual space surrounding this challenge is highly interdisciplinary, ranging from new research in networking and distributed system design to understanding the theoretical underpinnings of network science, policy, communication networks and economics,. Such research may generate new knowledge about the structure, behavior, and dynamics of the most complex systems – networks of networks – with potentially huge social and economic impacts [2][3].

B. GENI AM API

The GENI Aggregate Manager API is a common API for reserving disparate resources from multiple GENI aggregates. Prior to this API, each control framework specified a unique interface between aggregates and experimenters.

The GENI Aggregate Manager API specifies a set of functions for reserving resources and describes a common format for certificates and credentials to enable compatibility across all aggregates in GENI. The aggregate is an abstract concept represents set of resources. This API has been implemented in multiple control frameworks, and will serve as the basis for ongoing integration among GENI control frameworks and tools. Using this document, new GENI-interoperable aggregate managers, tools, and clearinghouses may be constructed [4].

 ISBN: 978-1-61208-216-5

C. Eucalyptus Cloud Platform

Eucalyptus [9] stands for 'Elastic Utility Computing Architecture Linking Your Programs To Useful Systems' and is an open source platform in the Infrastructure as a Service (IaaS)-style based on Linux. Eucalyptus is software available under GPL that helps in creating and managing a private or publicly accessible cloud. It provides an EC2-compatible cloud computing platform and a S3-compatible cloud storage platform [9].

D. Openstack Cloud Platform

OpenStack is a global collaboration of developers and cloud computing technologists aiming to produce a ubiquitous open source cloud computing platform for public and private clouds. The project aims to deliver solutions for all types of clouds with simplicity, ease of implementation, scalability, and feature selection.

Founded by Rackspace Hosting and NASA, OpenStack has become a global software community of developers who collaborate on a standard and massively scalable open source cloud operating system. All of the code for OpenStack is freely available under the Apach e 2.0 license, and anyone can run it, add to it, or submit changes back to the project. An open development model is the only way to foster badly needed cloud standards, remove the fear of proprietary lock-in for cloud customers, and create a large ecosystem that spans cloud providers.

The current OpenStack project has been divided into two kinds of software. The first, OpenStack Compute (Nova), is cloud management software used to operate and manage the infrastructure for large-scale provisioning of virtual machines. Second, OpenStack Object Storage (Swift) is storage system software that offers the reliable distribution of a store of objects.

E. GENICloud

GENICloud's goal is to allow the federation of heterogeneous resources like those provided by Eucalyptus, an open-source software framework for cloud computing, to coexist with GENI. Under the federation of Eucalyptus and GENI, a more comprehensive platform is available to users; for example, development, computation and data generation can be completed within the cloud, and deployment of the applications and services can be conducted on the overlay (e.g., PlanetLab).

By taking advantage of cloud computing, GENI users can not only dynamically scale their services on GENI depending on demand, they can also benefit from other services and uses of the cloud. GENICloud is complementary to Future Internet test-bed by federating heterogeneous resources, for example, a cloud platform with PlanetLab. Both PlanetLab and Eucalyptus architectures offer some insights into some of the similarities between the two seemingly disparate systems. PlanetLab comprises nodes scattered around the globe, and Eucalyptus consists of clusters. Both PlanetLab and Eucalyptus start out with some computing resources, namely, physical machines that can be provisioned to users [7].

F. PlanetLab

To provide a more realistic platform for researchers, PlanetLab is a test-bed for exploring disruptive technologies on a global scale. Testing distributed applications and network services on a global scale has always been difficult because deploying such applications and services could have adverse effects on the Internet. Also, PlanetLab is built as an overlay network to be positioned over the Internet. [5]

PlanetLab defines the treatment of a set of distributed virtual machines as a single, compound entity called a slice. The concept comes from the fact that, whenever a service is running on PlanetLab, it receives a slice (virtual machines running on different nodes) of the PlanetLab overlay network. An individual virtual machine within a slice is called a sliver. GENICloud has expanded the concept of slices to include Eucalyptus virtual machines and, in the future, storage capability. Therefore, a slice in GENICloud can have both PlanetLab resources and virtual machines from a Eucalyptus cloud. The users can log into individual slivers in a GENI Cloud slice to conduct their experiments.

G. Eucalyptus Aggregate Manager

Most of the implementation effort of GENICloud is concentrated on implementing the aggregate manager over Eucalyptus. In addition, a resource specification format is formulated for Eucalyptus.

The aggregate manager acts as a mediator between PlanetLab and a Eucalyptus cloud. The manager manages the creation of Eucalyptus instances for the slice and maintains a map of the slices and instances so when the users query the sets of resources allocated for their slices, the information is readily available.

H. Resource Specification (RSpec)

The resource specification is an XML document that can be used by the aggregate manager to return information to the users. The users can then use the specification to send information to the aggregate manager. Since the resource specification is in XML format, the format of the RSpec for a specific network is completely open for the network to define. With such openness, the RSpec can encompass many different types of resources and different network topologies. As a result, many networks (e.g., PlanetLab, VINI, ProtoGENI) have different RSpec formats [4][7].

GENICloud defined an RSpec for Eucalyptus, so that its resources and requests from users can be expressed in XML format. During the workflow, users interact with the slice manager using RSpec devised for Eucalyptus.

I. Definition of the Problem

Mobile cloud service has problems like packet loss, low bandwidth, bandwidth fluctuation, and delay fluctuation based on broadband communication and delay based on WAN. These problems obstruct users who want to use mobile cloud services. GENI Cloud supports interaction among heterogeneous resources on the Future Internet and Eucalyptus cloud computing platform. It provides better communication circumstance. However, the GENI Cloud project provides limited functionality, which includes few features of cloud computing capabilities.

III. CLOUD MOBILITY CONTROL FOR A MOBILE CLOUD

A. Key Features of Cloud Mobility Control

Mobile cloud service has problems like Packet loss, lower bandwidth, bandwidth fluctuation and delay fluctuation based

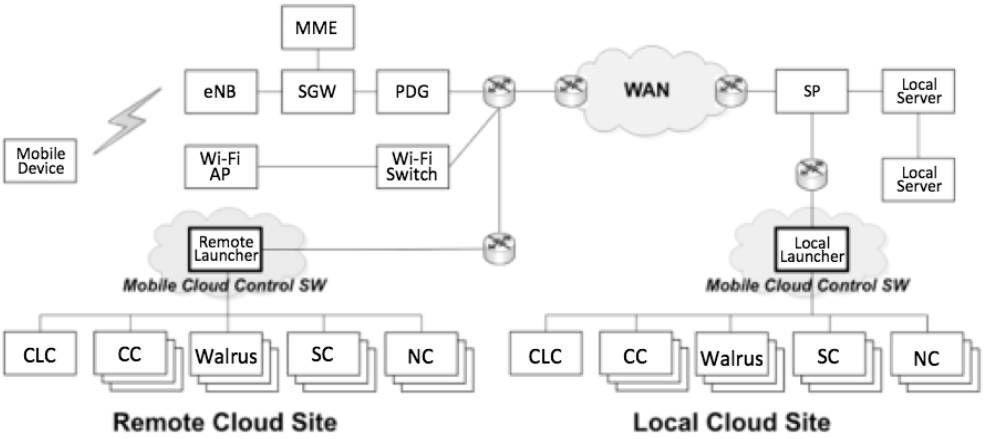


Fig. 1. The concept of proposed remote cloud based on mobile cloud architecture

on broadband communication and delay because of the WAN area. To solve problems with mobile cloud service, we proposed software which has a remote control function to improve cloud service. Followings are explained based on Eucalyptus as a cloud platform because of our first cloud mobility concept is established using Eucalyptus. In fact, it's more easy and clear to explain our concept of cloud mobility using Eucalyptus. However, the Openstack is very similar with Eucalyptus; it's very easy to apply this concept from Eucalyptus to Openstack.

B. Cloud Platform Based on Remote Cloud Mobility Control

In the mobile cloud platform based on a remote cloud, the mobile cloud provides an on-demand/pre-reserved virtualized service by incorporating the cloud server into the telecommunications and wireless carrier networks rather than using the server outside of the WAN, as shown in Figure 1.

Remote cloud's environment-information managing function is able to provide cloud service environment-information (which is originally at the 'local' cloud server on the outside of WAN) to cloud server in 'remote' mobile operator network. When user requests cloud service, Cloud Controller (CLC) determines the location of the new Virtual Machine (VM) to create. If service is requested through a mobile operator, CLC requests user environment-information from the remote cloud server and uses the information to create a VM. If service is terminated, CLC returns the user environment-information to the cloud server. Both the local cloud and the remote cloud provide mobile cloud services which use proxy server software based on the Eucalyptus cloud platform. It is simple software shown as 'Remote Launcher' and 'Local Launcher' in Figure 1. Each launcher substantially controls cloud mobility as explained in the next sections.

C. Design of Cloud Mobility Control

We establish a design based on a remote cloud system for the proposed cloud mobility control. Table I represents the common message header for cloud mobility control. First, we divide local and remote cloud systems into categories based on 'Kind of cloud system.' In addition, we should be able to support seamless mobility control by including 'Type of component,' 'Type of message-passing,' 'Message order,' and 'Source and Destination IP addresses' in the header.

We propose a design for cloud mobility control software based on the following four basic functions for controlling local and remote cloud systems.

D. Cloud Computing Mobility Environment Configuration

Local launcher is a gateway for administering the local cloud system in a remote cloud architecture environment. Each cloud system (local and remote) exchanges environment-information with another cloud system and utilizes the appropriate information from service requests and communicates with another.

TABLE I. COMMON MESSAGE HEADER FOR CLOUD MOBILITY CONTROL

Component	Size (Octet)	Default	Meaning	
Kind of cloud system	1	0x00	0x00	Local cloud system
			0x01	Remote cloud system
Source component type	1	0x00	0x00	Default(App launcher)
			0x01	CLC(Cloud controller)
			0x02	CC(Cluster controller)
			0x03	SC(Storage controller)
			0x04	Walrus
			0x05	NC(Node controller)
Type of message passing	1	0x00	0x00	Default
			0x01	Request
			0x02	Response
Message order	1	0x00	Message order	
Source IP	4	0x00000000	Source IP address of a message	
Destination IP	4	0x00000000	Destination IP address of a message	

E. Mutual Recognition and Authentication between Cloud Systems

The cloud systems also process mutual recognition and authentication. If a remote user makes a request to the cloud system, the local cloud exchanges authentication information for remote cloud service between the local cloud and the remote cloud. By exchanging authentication information, the connection setting is established.

Each cloud system checks for available components on its own system. CLC shows the process for periodically checking for system components through Eucalyptus API to CC, SC, Walrus, and NC. The remote launcher and local launcher return information about the available system to CLC using a query.

F. Activation of the Remote Cloud Server

The remote cloud server's activation function shows remote cloud activation in the cloud environment. If a remote user requests cloud service from the cloud system, the local cloud system activates the remote cloud service function. After a recognition step between the local cloud and the remote cloud, each cloud's connection settings are established, and the remote cloud system activates cloud service. Connection is established between the user and the remote cloud system. Users can utilize cloud service in the remote cloud system.

If the user requests a service, the local cloud launcher uses the user network. After that, the system request to the remote cloud system regarding OS image information, the remote cloud and local cloud synchronize the image list and transfer image files. The local cloud system requested information from the remote cloud system about the user requested application.

The user's operating information consists of a kernel, ramdisk, and image file. The remote application launcher registers on the eucalyptus cloud system. If operating information is registered on the system, the system returns the ID values of EKI (Eucalyptus Kernel Image), ERI (Eucalyptus Ramdisk Image), and EMI (Eucalyptus Machine Image).

If the operating image file completes registration on the Eucalyptus system, the remote application launcher creates a keypair with a key value for communication with each Openstack instance. After creating the keypair, the remote application launcher requests creation of an instance on the Eucalyptus cloud system.

During generation of the instance, the Eucalyptus system uses the appropriate needed parameters like key-pair's name, EMI ID and VM type. Upon completion of instance creation, the Eucalyptus system returns the ID of the instance for registration.

If the instance is normally driven on the cloud system, the application launcher periodically checks the status of the instance. In this process, the remote application launches a connection with the instance and transfers the user's application. After this, the remote application launcher returns an IP address for the instance from the local application launcher to receive cloud services. A user re-requests the cloud service based on the IP address, which returns the remote application launcher.

Each cloud's application launchers check the CPU usage, RAM usage, and HDD usage for status information in order to manage resources.

G. Deactivation of the Remote Cloud Server

If usage of cloud resources is low, the local cloud system is deactivated from the remote cloud system, and the user requests cloud service termination. After this, the instance in operation is stopped on the remote cloud system, and the user's image and instance information is transferred to the local cloud system.

IV. FIRSTCLOUD FOR FUTURE INTERNET

A. Key Features of FiRSTCloud

In this section, we propose the FiRST Cloud AM based on GENI AM API for the cloud computing platform to extend the limited functionality of GENI Cloud project [4][6][7].

FiRST Cloud AM acts as a moderator between the OpenStack cloud and the future internet test-bed. Also, FiRST Cloud AM manages mapping from an instance to the slice when a user queries about resource allocation on the slice. Therefore, FiRST Cloud AM maintains this mapping information between an instance and a slice. To moderate between instance and slice, FiRST Cloud AM creates a database of openstack instances and slice information, as in Table II.

FiRSTCloud AM provides six APIs (all except RenewSliver()) using GENI AM API: GetVersion(), ListResource(), CreateSliver(), DeleteSliver(), SliverStatus() and Shutdown(). Additional features may exist depending on the existing API.

FiRSTCloud AM defines the RSpec which is submitted by the user to describe instance-specific information and resource information. RSpec is managed differently depending on the items after parsing. The RSpec contains items such as cloud image information that includes image id and state, key-pair, instance and vm information.

B. GetVersion() API of FiRSTCloud AM

FiRST Cloud AM returns the version of the GENI Aggregate Manager API supported by this aggregate. Version information includes the OpenStack cloud version.

TABLE II. DB TABLE OF SLICE AND INSTANCE

Slice		
Name	***Type***	***Key***
ID	INTEGER	PRIMARY KEY
Slice urn	TEXT	

Openstack Insatance		
Name	***Type***	***Key***
ID	INTEGER	PRIMARY KEY
Instance ID	TEXT	
Kernel ID	TEXT	
Image ID	TEXT	
Ramdisk ID	TEXT	
Instance type	TEXT	
Key pair	TEXT	
Slice ID	INTEGER	REFERENCES Slice(ID)

 ISBN: 978-1-61208-216-5

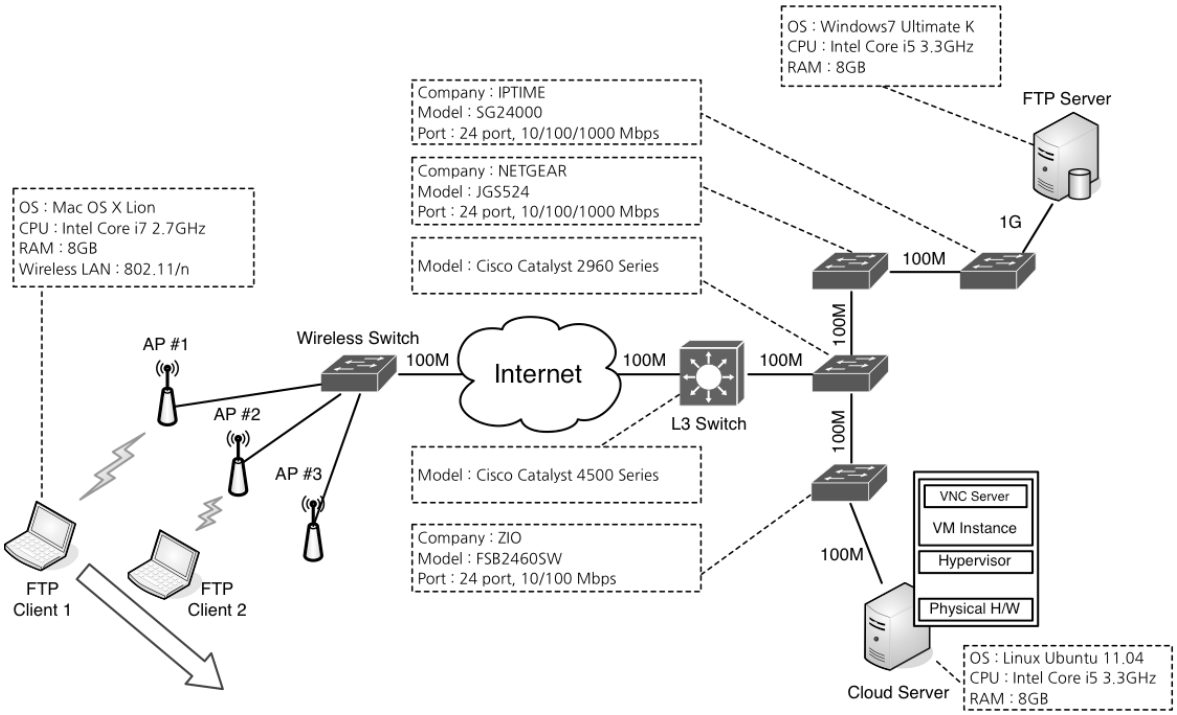


Figure 2. System environment for proposed mobile zero-client.

C. *CreateSliver() API of FiRSTCloud AM*

FiRST Cloud AM is able to allocate resources to a slice. Also, this operation is expected to asynchronously activate the allocated resources after the operation has been successfully completed.

Callers can check on the status of the resources using SliverStatus API.

To connect with OpenStack Cloud, first, use the boto library which is compatible with the EC2 proceeds. Then, initialize the database information of instance and slice. Create a new instance from RSpec by parsing the image information, virtual machine type, and keypair information. Finally, return the id of the created instance by creating a new RSpec.

D. *ListResources() API of FiRSTCloud AM*

FiRST Cloud AM returns information about available cloud resources or resources allocated to a slice. To connect with OpenStack Cloud, use the boto library which is compatible with the EC2 to connect. Then, request the available zone information, registered image information, and keypair information; instances of OpenStack Cloud AM are returned in the form of a list of values. Finally, return the cloud information by creating a new RSpec.

E. *DeleteSliver() API of FiRSTCloud AM*

FiRST Cloud AM is able to stop sliver and delete if the sliver is running. AM search instance information occurs in the DB which is mapped to slices to be deleted. If AM finds an instance, it can be terminated using the boto library, followed by a DB update.

F. *SliverStatus() API of FiRSTCloud AM*

FiRST Cloud AM is given the status of a sliver. Additionally, AM requests the connection to the OpenStack Cloud that verifies the status of the instance as well as the sliver. Returned status information is based on the instance information for the corresponding slice_urn (uniform resource name). Based on this instance information, the final status of the sliver is determined and returned to the client as in ListResources() API.

G. *Shutdown() API of FiRSTCloud AM*

FiRSTCloud performs an emergency shutdown of a sliver. This operation is intended for administrative use. In addition, this API is obtained from a database associated with slice_urn and the instance, then terminates and manages the instance.

V. PERFORMANCE EVALUATION AND ANALYSIS

A. *Key Features of the Mobile Zero-Client*

In this paper, we proposed the mobile zero-client based on cloud mobility control and FiRSTCloud AM. For mobile cloud service, we used Virtual Networking Computing (VNC) which includes a graphic desktop share system through a Remote Frame Buffer (RFB). Zero-client means end user device has no local storage and just has weak process power to communicate with server. Following evaluations, however, zero-client is substituted by common laptop running VNC viewer only. On the cloud server, there is a Linux OS installed instance to run VNC server application.

 ISBN: 978-1-61208-216-5

B. *Performance Analysis of the Mobile Zero-Client*

Network topology is constructed for mobile zero-client performance analysis as in Figure 2. In this performance analysis, we want to present mobile zero-client on mobile cloud performs better than normal mobile device.

There are two hypotheses before analyze performance of our proposal. First, we assume that the cloud mobility server and client are well operated. Therefore whole information and data in local site downloaded to remote site already.

Second, we did not consider about interworking with Future Internet test-bed in performance analysis. It's for comparison with normal mobile device and for convenience of experiment.

The performance analysis is operated by receiving a file from FTP server.

There are two different traffic cases. In the first case, no traffic is generated on the network. In the second case, another mobile device generates traffic at the second static AP. The second mobile device also downloads the same file from the FTP server. In these two network environments, the mobile device downloads a 700MB video file using 802.11n WLAN. Downloading occurs in two ways, through the use of a mobile device to download directly from the FTP server and connects to Openstack instance using VNC client as a mobile zero-client. When using mobile zero-client, the mobile device does not download from the ftp server but from an Openstack cloud to which the mobile zero-client is connected.

Table III shows the result of performance evaluation. There are four scenarios, and each row represents a scenario. Scenario 1 means the data transfer rate of mobile devices when downloading a video file from an FTP server with no traffic on the network. In this scenario, data transfer rate is unstable because of the wireless network environment. The average data transfer rate measured 29.75 Mbps. Scenario 2 means the data transfer rate of the mobile zero-client download video file from an FTP server with no traffic on the network environment. Data transfer rate is stable because the mobile zero-client received the data through OpenStack cloud instance (VM). Average data transfer rate measured 67.65 Mbps. When using a mobile zero-client, we achieve a similar performance to that of a wired network user in our wireless network environment. Scenario 3 means the data transfer rate of the mobile device when it is directly downloading a video file from the FTP server. In addition, this and next scenario correspond to the second traffic case which is mentioned. Therefore the data transfer rate little bit decreased when wireless AP was shared with another client, producing an average data transfer rate of 27.19 Mbps. Scenario 4 means the data transfer rate of a Mobile Zero-Client downloading a video file from the FTP server with traffic using another mobile device. The average data transfer rate measures 65.89 Mbps.

TABLE III. AVERAGE DATA RATE ON 4 SCENARIOS

	Traffic load	**Client type**	**Average data rate (Elapsed time)**
Scenario 1	N/A	Normal client	29.75 Mbps (193.26 sec)
Scenario 2	N/A	Zero-client	67.65 Mbps (84.92 sec)
Scenario 3	Another device	Normal client	27.19 Mbps (211.40 sec)
Scenario 4	Another device	Zero-client	65.89 Mbps (87.23 sec)

VI. CONCLUSION AND FUTURE WORK

Future Internet research emphasizes harmonization of the conventional system with Future Internet research, network virtualization, and cloud computing. Providing high bandwidth and low delay is possible, but computationally intensive services or computing operations cannot be performed. A cloud computing platform can perform many service and computation operations. Its disadvantages include limited bandwidth and highly variable latency.

Researchers have begun to test cloud computing environments in large-scale global test-bed systems. In this paper, we developed cloud mobility for mobile communication between a device and the cloud server. Second, we developed the FiRST Cloud aggregate manager (AM) based on GENI AM API for interaction between the future internet test-bed and the OpenStack cloud platform. Third, we developed a Cloud Mobility Client/Server for mobile cloud management in order to control the zero-client service. We confirmed that the proposed FiRSTCloud AM works with zero-client mobile cloud service.

Through this work, mobile cloud service was shown to have a consistent quality regardless of mobile device performance or wireless environment.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012R1A1A1006620).

REFERENCES

[1] M. K. Shin, "Trend on the Future Internet Technologies and Standardization," Electronics and Telecommunications Trends, vol. 22, no.6, pp.116-128, Dec. 2007.

[2] K. H. Nam, S.J. Jeong, M. K. Shin and H. J. Kim, "Technology and Trends of GENI Control Framework," Electronics and Telecommunications Trends, vol. 25, no.6, pp.157-166, Dec. 2011.

[3] GENI white paper, "GENI at a glance", [Online] http://www.geni.net/wp-content/uploads/2011/06/GENI-at-a-Glance-1Jun2011.pdf, <link> 07.16.2012.

[4] GENI API wiki page, http://groups.geni.net/geni/wiki/GeniApi, <link> 07.16.2012.

[5] PlanetLab, http://www.planet-lab.org, <link> 07.16.2012.

[6] S. W. Lee, S. W. Han, J. W. Kim, S. G. Lee, "FiRST: Korean Future Internet Testbed for Media-Oriented Service Overlay Network Architecture," Journal of Internet Technology, vol. 11, no. 4, pp. 553-559, Jul. 2010.

[7] M. Yuen, "GENI in the Cloud", University of Victoria, 2010.

[8] Openstack official homepage, [Online] http://www.openstack.org, <link> 07.16.2012.

[9] Eucalyptus official homepage, [Online] http://www.eucalyptus.com/, <link> 07.16.2012

Provenance Framework for the Cloud Environment (IaaS)

Muhammad Imran
Research Group Entertainment Computing
University of Vienna, Austria
Email: imran.mm7@gmail.com

Helmut Hlavacs
Research Group Entertainment Computing
University of Vienna, Austria
helmut.hlavacs@univie.ac.at

***Abstract*—Cloud providers can optimize resource utilization and energy consumption by finding patterns in their usage. One way of finding such patterns is to study the history of Cloud resources activity. This approach is known as Cloud provenance. Provenance can also be used to track errors and faults in Cloud services. We have developed a provenance framework for research Clouds in order to find the history of the resources usage. Our framework collects provenance data in response to the request of users for IaaS scheme. In this paper, we discuss a provenance framework in the Clouds and present different possible approaches of the provenance collection process. To the best of our knowledge, provenance is yet to be addressed in the Cloud environment. Hereby, we provide details of our proposed framework and present its performance evaluation. The experimental results show that our provenance framework has a very low overhead (less than milliseconds), which makes it ideal for the Cloud infrastructure.**

Keywords-provenance framework; cloud IaaS.

I. INTRODUCTION

The vision of Cloud is to address a complex engineering, medical or social problem. Cloud enables the end user to process huge amount of data and/or satisfy his needs for mass computational power via resource virtualization. The experiments are performed on Cloud on a large scale and shift of technology is already in progress [1], [2]. Infrastructure as a Service (IaaS) is the new paradigm for researchers to deploy complex applications into Cloud. This is different than Grid [3] and distributed environments where a user had to adopt their application to the grid infrastructure and policies. IaaS scheme provides a raw resource which is hired and updated according to the requirement of the application by a user without knowing the complexity and details of the underlying architecture. A resource is hired when a match is found based on a user and application requirements such as memory, disk space, resource type and/or Cloud provider. This is called on-demand computing and in the process of resource allocation, a user is charged with some price. Once a resource is updated and used, the user may take a snapshot of the resource if the same resource is to be used later on.

Workflow [4] is designed to execute activities in order for a complex application in e-Science. Provenance of a workflow activities [5] is the information about intermediate data and processes to verify the execution of an application. Provenance in general means; "the origin or source of an object". In Clouds, provenance can be broadly categorized into user data (applications installed on a virtual machine), instance type (memory, disk size, number of instances) and resource type (image ID, location). Such information is of high importance to utilize the cloud resources, e.g., a resource already built and updated by one user can be used by others with minimum or no change of the installed applications and components. Furthermore, mining provenance data can be used to forecast a future request, e.g., Eddy Caron [6] used string matching algorithm on recent history data to forecast a next request. Similarly, networks in general and Clouds in particular are prone to errors, and the history data can be utilized in Clouds to resolve the errors with minimum effort.

Clouds are still in the process of evolution and provenance is yet to be implemented (addressed) in Clouds. Contributions of this paper are the following:

- A brief overview of research Clouds IaaS and a detailed discussion of possible schemes to incorporate provenance into Cloud environment.
- A use case of provenance usage and example metadata from IaaS Cloud.
- Detailed architecture of our provenance framework for Cloud IaaS and the evaluation of collecting and storing provenance data.

The rest of the paper is organized as follows. Section II summarizes the research Cloud architecture and discusses the possible provenance schemes. Section III gives the details of the underlying architecture (middleware) used by the research Clouds and the extension of this architecture to collect provenance data. Section IV gives a brief overview to use the provenance data and utilize Cloud resources. In Section V, we present the test results of the collection and storage module. Section VI concludes our work and presents the directions for the future implementations.

II. RESEARCH CLOUD IAAS ARCHITECTURE AND DISCUSSION OF PROVENANCE SCHEMES

A. Research Cloud IaaS

Cloud computing is generally categorized into three types which are business, Research or private and hybrid Clouds.

 ISBN: 978-1-61208-216-5

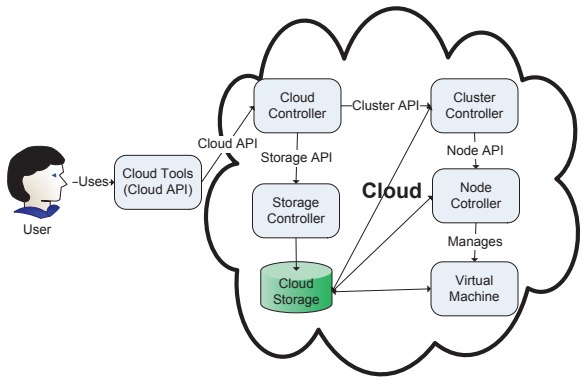


Figure 1. Generalized Cloud Architecture.

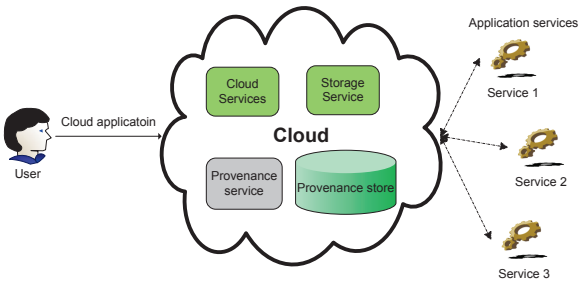


Figure 2. Provenance Service as Part of Cloud Services.

They are further subdivided into three schemes which are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Three different Clouds, i.e., EUCALYPTUS [7], Nimbus [8] and OpenNebula [9] were explored to understand their internal architecture and communication mechanism between various components. This helped us to discuss the possible provenance schemes for Cloud IaaS and implication of our proposed framework. The main components of IaaS Cloud are summarized below:

- Application tools: Application Programming Interface (API) available to communicate with Cloud services, e.g., resource hiring, starting, stopping, saving and/or describing the state of a particular resource.
- Cloud, Cluster and Node: The request of users is handled by the Cloud and routed to Clusters and Nodes respectively. The Node communicates with Virtual Machine (VM) and the job of Cluster is to manage different nodes in the network.
- Storage: Cloud offers storage unit (file system) to save user data and raw disk images to be run as resources. Communication with a storage unit is controlled by a service, e.g., Walrus in Eucalyptus Cloud and a user can save the updated state of a running machine. The process of saving the updated machine into Cloud is called snapshot.

Figure 1 presents a generalized architectural overview and control flow from user to VM in Cloud IaaS.

B. Provenance as a Part of Cloud Services

In this scheme, the Cloud provider needs to provide a service which will communicate with other Cloud services including cluster, node and storage to collect provenance data. This scheme proposes the application of provenance as a part of overall Cloud Infrastructure. The following list the advantages of provenance inside the Cloud IaaS.

- Easy to use as provenance is already a part of Cloud infrastructure and a user can decide to turn it on/off just like other Cloud services.
- Users will prefer this scheme as they do not need to understand the structure of provenance framework and is the responsibility of the Cloud provider to embed such a framework.

The following lists the disadvantages of such a provenance scheme.

- Cloud providers cannot charge users for such a scheme unless it has some benefits of resource utilization and initialization for users.
- In case of Cloud services failure, provenance system will also fail and there is no way to trace the reason for the failure.
- There will be extra burden on the Cloud provider because the usage of Cloud resources must increase due to incorporating the provenance system as a part of Cloud framework.
- Such scheme can only work with a particular version of Cloud IaaS. Any change in Cloud model or services signature needs an appropriate change in provenance application.

In distributed, grid and workflow computing, there are many examples of provenance data management and schemes [10]–[13]. Each of these schemes is designed for a particular environment and they rely on the underlying services model. Therefore, the existing techniques cannot be applied to Cloud environment and further, Cloud services are not extensible to third party applications. Figure 2 presents a provenance system as a part of Cloud services.

C. Provenance is Independent of Cloud Services

A provenance scheme which adopts a modular and an agent like approach to address cross platform, applications and different Cloud providers is independent of Cloud infrastructure. Such a scheme must address on-demand, pay as you go and extremely flexible Cloud architecture. Advantages of an independent provenance scheme are:

- Independent of Cloud services and various applications domain.
- Failure of Cloud will not affect provenance scheme as it is not a part of Cloud.

 ISBN: 978-1-61208-216-5

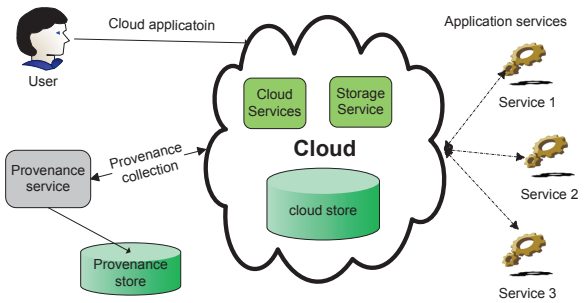


Figure 3. Provenance as an Independent Module

- The users and Cloud providers will be able to track faults and errors if some Cloud services failed to work properly.
- Usability and simplicity of such a scheme is very high because the user has complete control of the provenance system.

Disadvantages of such a scheme are as follows:

- Complete understanding of Cloud services is required to make any changes and communicate with the Cloud infrastructure.
- Trust is required on behalf of the Cloud provider because of request, permission and response from the Cloud services to the provenance module.
- Any change in Cloud services, their signature, or communication mechanism will need an appropriate change in provenance scheme.

In workflow computing, Karma [14] is using a notification broker where all the activities are published to and stored in a provenance store. The technique proposed by the Karma service is not part of a workflow enactment engine and it works as a bridge between the provenance store and the enactment engine. Figure 3 gives a brief overview of an independent provenance scheme in Cloud.

D. Discussion

Both of these approaches have their pros and cons. While considering provenance for Cloud IaaS, the major challenge is to address the Cloud extensibility. Clouds are not extensible by nature and in case of open Clouds, a developer needs a deep understanding of the source code in order to make any changes. Keeping this point in view, we propose a provenance framework which is independent of Cloud IaaS provider and with minimal or no changes required in Cloud services.

III. Provenance Framework

Research Clouds rely on the open source technologies to provide an infrastructure (IaaS). These open source technologies includes JAVA and C/C++ languages, and Apache, Axis and Mule [15] communication frameworks. A general consensus is that Cloud would not be possible without these open source technologies. Research Cloud, e.g., Eucalyptus use Apache, Axis2/C and Mule engines to deploy Cloud services as IaaS and built a communication mechanism between different components. Apache is widely used for its speed, lightweight engine and its support of SOAP, WSDL and REST interfaces. Similarly, Mule is an integration platform used to connect various applications and/or services. These technologies are used to connect various Cloud components and are called middleware.

This middleware is extensible and a developer can add custom methods to the already deployed applications and service. The proposed framework is based on this feature of extensibility from Apache, Mule and other third party tools and consists of the following components: **Provenance Collection**, **Provenance Parsing**, **Provenance Storage**, **Provenance Query** and **Provenance Visualization**. First, the explanation of Apache architecture and its extension for the development of the provenance framework is given.

A. Apache (Axis2/C)

To develop a provenance framework, the following components of the Apache architecture are utilized.

Handler: or interceptor is the smallest execution unit in the message passing system of the Apache Engine. The idea is to intercept the flow of a message and perform the additional task submitted by a user. Handler can read and write to the message context (apache messaging system). A handler has two parts: header and body. The header specifies the name and body the operation. There are predefined handlers in the Apache Axis execution chain and also the ability to provide custom handlers developed by the developers [16]. A group of handlers that is orchestrated and deployed within the Apache engine is called a module.

Phase: is the concept in Apache Axis to support the dynamic ordering of the handlers. It acts like a bucket in which where the handler is put. A phase can have one or more handlers. Apache provides different kinds of phases spanning from global (for overall axis communication) to operational (for a particular operation or web method).

Flow: is a collection of phases. Phase is more like a logical collection where flow is a real execution chain. There are four types of flows in Apache engine.

- InFlow
- OutFlow
- InFaultFlow
- OutFaultFlow

Similarly, other third party libraries and frameworks used by Clouds are also extensible. Examples of such frameworks are the use of Mule in Eucalyptus, Axis in Nimbus and Apache xml-rpc in OpenNebula. The basic architecture of these libraries is different but the main idea of an interceptor or handler is the same. For example, in Mule, the message context is referred to as Mule Message. Interceptors can be deployed before and after a component is invoked in the

 ISBN: 978-1-61208-216-5

Mule framework. The Mule message before and after resides in flows which are called inbound-router and outbound-router respectively.

B. Provenance Collection

When a message enters Apache engine, it goes through InFlow and invokes all the handlers inside. InFaultFlow is similar and handles a faulty incoming request, e.g., sending wrong arguments to the web service method or any other unexpected condition that prevents the request to succeed. OutFlow is invoked when a message is moving out of Apache engine (invoking all handlers in OutFlow) and the OutFaultFlow is invoked when something goes wrong in the out path, e.g., a host is shut down unexpectedly. Various Flows within Apache engine and the execution of a service with input and output messages is described in figure 4. The left side of figure details the different flows and the right side gives an overview of one single flow with phases and handlers concepts (both built in and user defined). Custom handlers, using C/C++ for provenance collection are deployed in four different Flows of the Apache execution chain. When a component inside Cloud IaaS is invoked, provenance collection module intercepts the flow, collects and parses the message for provenance data in the corresponding execution flow.

C. Provenance Parsing and Storing into XML File

SOAP message inside Apache engine is intercepted by the collector module which passes this message to the parser. The parser reads the SOAP message, parse it accordingly and store the data in a well defined XML file. We used XML schema for the collected provenance data because it is widely used model for data representation. The XML can be used to maximize the advantages of custom algorithms and third party applications. To query the provenance data, it is better to provide a standard schema and hence the usage according to individual preferences.

TableI presents a sample of collected, parsed and stored provenance data by our provenance framework. This data represent user activity for methods of Eucalyptus cluster service and detail the timestamps, resource type and instance specific information. <UserData> is the list of applications specified by user to populate the resource and <TimeStamp> are corresponding start and finish time for a web service method.

D. Provenance Query

Custom applications can query provenance data based on the user requirements. We find the activity pattern in Cloud IaaS based on a resource type, instance type, time used or user ID in our example query. This information can be used to monitor Cloud IaaS and the frequently used resources can be moved to a faster CPU/disk unit for better performance. Algorithm 1 is used to find activity patterns based on the the resource-ID.

Algorithm 1 Solve Query Q: Q = Return Resource Types (emi-IDs) in XML Store

Require: XMLStore, ClusterName
Ensure: XMLStore is not Empty
  Begin
  Array ResouceType[] T
  OpenXMLFile(XMLStoreLocation)
  FindCluster(ClusterName)
  while ParentNode<MethodName> == RunInstance) **do**
    $T \leftarrow$ ChildNode(<ImageID>)
  end while
  End

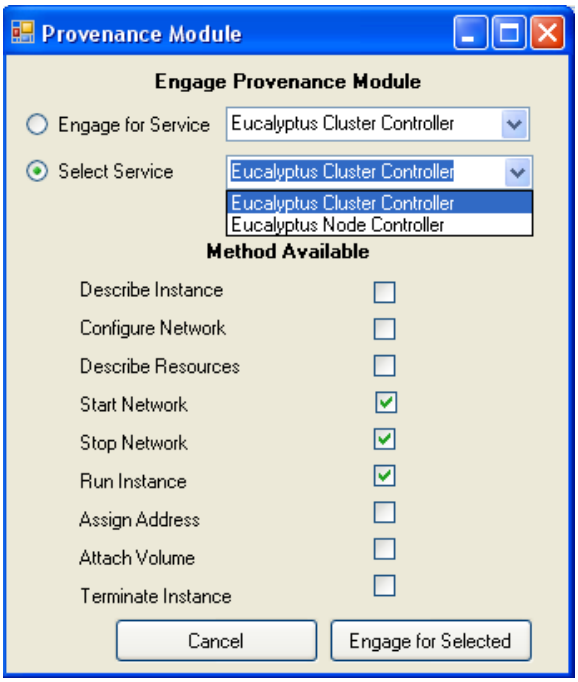


Figure 5. User Interface for Engaging Provenance Module into Cloud

E. User Interface

Usability of the proposed provenance framework is very high. Cloud providers can enable/disable the provenance module according to their choice. Different options are available to enable/disable the provenance module based on the requirements of the Cloud provider. Some of the options are: to enable/disable provenance module for all clusters, a particular cluster, all nodes, a particular node, or selected methods from a particular cluster or node. Figure 5 presents a prototype of user interface available to Cloud IaaS provider.

F. Framework Experience

By extending the middleware (Apache and Mule) using handlers, we are independent of Cloud provider and different IaaS schemes. We followed a modular approach and divided

 ISBN: 978-1-61208-216-5

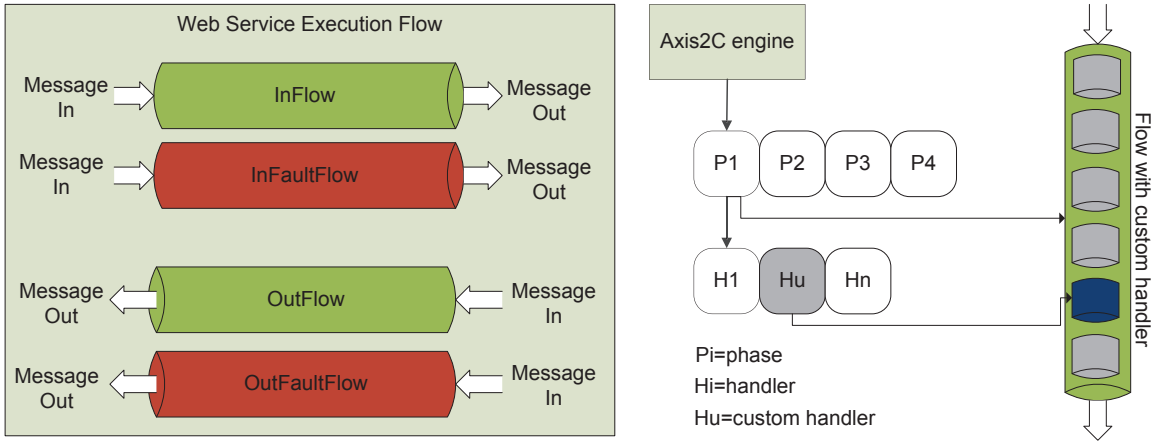


Figure 4. Apache Axis2C Architecture

<EucalyptusServiceName> ClusterController</EucalyptusServiceName>		
<MethodName>StartNetwork</MethodName>	<TimeStamp> Start and End Time of Method</TimeStamp> <ClusterAddress> 131.130.32.12</ClusterAddress> <UserID>admin</UserID>	
<MethodName>RunInstance</MethodName>	<ImageID>emi-392B15F8</ImageID> <KernelID>eki-AE1D17D7</KernelID> <RamdiskID>eri-16981920</RamdiskID> <ImageURL>emi-URL</ImageURL> <RamDiskURL>eri-URL</RamDiskURL> <KernelURL>eki-URL</KernelURL>	Instance Type <Name>m1.small</Name> <Memory>512</Memory> <Cores>1</Cores> <Disk>6</Disk> <UserData>DataFile</UserData>
<MethodName>StopNetwork</MethodName>	<TimeStamp> Start and End Time of Method</TimeStamp> <UserID>admin</UserID>	

Table I
SAMPLE METADATA FOR CLOUD IAAS

our framework into different components. The future of provenance in Cloud lies in a lightweight and independent provenance scheme to address cross platform, Clouds IaaS and application domains. The proposed framework can be deployed without making any changes to the Cloud services or architecture. **Advantages** of the scheme are:

- It is independent of Cloud services and platform and it works with any Cloud IaaS which use the Apache, Mule or similar frameworks.
- The proposed framework follows a soft deployment approach and therefore, no installation is required.
- Some of the challenges offered by Cloud infrastructure are virtualization, “on demand” computing, “pay as you go” model, more abstract, extremely flexible and the services are not extensible by nature. The proposed framework address these challenges in automatic fashion as being part of Cloud middleware.

Major **Disadvantage** of proposed framework is:

- Rely completely on the extension of the middleware and cannot work on any other Cloud IaaS where middleware is not extensible.

IV. CLOUD PROVENANCE: A USE CASE FOR EFFICIENT RESOURCE INITIALIZATION AND ENERGY CONSUMPTION

Description: Resource utilization is critically important both from the resource provider and Cloud performance perspective. In the Cloud resource allocation process, a user may request a resource with the input file of required applications that is the same as a previously initialized resource but will still need to build the resource from scratch. The Cloud resource utilization can be maximized if one is able to provide automatic discovery of already running instances, saved volumes and snapshots. The automatic discovery will not only help in resource utilization but will also provide means to reduce the time and energy consumed. Our proposed framework collects the metadata information regarding time, user, cluster and location of newly created volumes or snapshots and stores it in a provenance database. To make the process of resource allocation efficient and automatic, the broker (which takes input from user) compares the user input file with existing provenance data. If the comparison of input file results in an exact match then instead of starting a new resource from scratch, the existing resource volume and snapshot are deployed.
Actors: End-user and Cloud provider. A user benefits from

 ISBN: 978-1-61208-216-5

Table II
UNDERLYING ARCHITECTURAL COMPONENTS

Cloud provider	Operating system	Cloud services engine	Languages	Storage unit	Virtualization	Service tested
Eucalyptus 1.6.2	Linux Ubuntu 10.04 Server	Axis2/C 1.6.0	C,C++	File system (XML)	KVM/XEN	Cluster controller

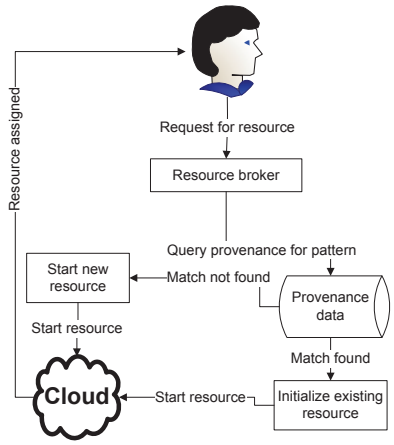


Figure 6. Resource Initialization Using Provenance

this scheme by saving his time and effort to build a resource from scratch. On the other hand, the Cloud provider utilizes existing deployed resources and saves energy.

Advantages:

- Faster resource initialization in case a match is found.
- Utilization of existing deployed resource (volumes, snapshots) to save energy, cost and time.
- Overall Cloud performance will increase.

Figure 6 describes the process of using provenance data and making Clouds more efficient and proactive.

V. EVALUATION

Different approaches are proposed in literature for collecting and storing provenance data to reduce the computation and storage overhead [17]. Mainly, there are two methods. The first method proposes to collect provenance data and store a copy of the parent object. The disadvantage of this method is a huge storage overheard. The second method proposes to store links of the parent object. This method is faster and storage overhead is very low. Disadvantage of this method is consistency in case a parent object is deleted or moved.

To store provenance data we followed the second approach and the proposed framework stores only the link information about the activity of users and Cloud components. The provenance data consists of information like: Cloud images, snapshots, volumes, instance types and user data etc. Real data is already stored in the Cloud storage unit and we do not make a copy of this data. Since links are lightweight, therefore computation and storage overhead for the provenance data is negligible.

We evaluated the cluster controller service and results were surprising for collection and storage module. To get physical evidence, timestamps were calculated at the beginning of provenance module invocation and later on when the data is parsed and saved into XML file. Time overhead including the provenance module for Inflow and Outflow phases of Apache were less than milliseconds. To find the storage overhead we calculated file size of provenance data for individual methods. We chose a worst case scenario where all the incoming and outgoing data was stored. This process was performed for every method in Eucalyptus cluster service and the average file size of stored provenance data is about 5 KB for each method. Evaluation was performed by using the underlying architecture detailed in table II. Physical machine details for running IaaS Cloud are the following:

Number of PCs: 2 (PC1 with Cloud, Cluster and Storage Service, PC2 with Node service), Processor: Intel Core (TM) 2: CPU 2.13 GHz, Memory: 2GB, Disk Space: 250 GB

It is essential to note that the low computation and storage overhead of the provenance frameworks is because of two reasons. First, we used an approach where the extension of the middleware is achieved by built in features. This approach does not add any extra burden except the collection of provenance data. Second, we store the provenance data by using a link based approach. This approach saves on duplicating the storage of huge amounts which already exists in Cloud database.

VI. CONCLUSION AND FUTURE WORK

With the evolution of technology and IaaS, complex applications are target environment for Clouds. Clouds offers "on demand" computing and "pay as you go" model, where applications discover resources at run time. The focus of this paper is provenance data for Cloud IaaS scheme. A client application hires resources from IaaS and populates them according to the requirements. These populated resources are saved in Cloud storage unit and can be used by other applications having the same requirements. This process requires storage of users or application activity. First, we discussed general approaches to collect activity information performed on Cloud IaaS with their pros and cons. By using those approaches as the basis of our study, we developed a framework which is not dependent on Cloud services or underlying architecture. We divided our framework into different components and proposed a use case scenario where the collected provenance data can be used to utilize Cloud resources and to save cost, energy and time. Collecting and

 ISBN: 978-1-61208-216-5

storage overhead of the proposed framework is very low. In the future we will extend the framework for resource utilization in order to save cost, time and energy using provenance.

REFERENCES

[1] C. N. Hoefer and G. Karagiannis, "Taxonomy of cloud computing services," in *Proceedings of the 4th IEEE Workshop on Enabling the Future Service-Oriented Internet (EFSOI'10), Workshop of IEEE GLOBECOM 2010, Miami, USA*, ser. 2010 IEEE GLOBECOM Workshops. USA: IEEE Communications Society, December 2010, pp. 1345–1350.

[2] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the use of cloud computing for scientific workflows," in *Proceedings of the 2008 Fourth IEEE International Conference on eScience*, ser. ESCIENCE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 640–645.

[3] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, no. 3, pp. 200–222, Aug. 2001.

[4] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, *Workflows for e-Science: Scientific Workflows for Grids*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[5] R. S. Barga, Y. L. Simmhan, E. Chinthaka, S. S. Sahoo, J. Jackson, and N. Araujo, "Provenance for scientific workflows towards reproducible research." *IEEE Data Eng. Bull.*, vol. 33, no. 3, pp. 50–58, 2010.

[6] E. Caron, F. Desprez, and A. Muresan, "Forecasting for grid and cloud computing on-demand resources based on pattern matching," in *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, ser. CLOUDCOM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 456–463.

[7] Eucalyptus. [retrieved: may, 2012]. [Online]. Available: http://open.eucalyptus.com/

[8] Nimbus. [retrieved: may, 2012]. [Online]. Available: http://www.nimbusproject.org/

[9] Opennebula. [retrieved: may, 2012]. [Online]. Available: http://opennebula.org/

[10] M. Szomszor and L. Moreau, "Recording and reasoning over data provenance in web and grid services." ser. Lecture Notes in Computer Science, R. Meersman, Z. Tari, and D. C. Schmidt, Eds., vol. 2888. Springer, 2003, pp. 603–620.

[11] Y. Cui and J. Widom, "Lineage tracing for general data warehouse transformations," in *Proceedings of the 27th International Conference on Very Large Data Bases*, ser. VLDB '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 471–480.

[12] P. Buneman, S. Khanna, and W. chiew Tan, "Why and where: A characterization of data provenance," in *ICDT '01: Proceedings of the 8th International Conference on Database Theory*. Springer, 2001, pp. 316–330.

[13] Y. L. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance Techniques," Computer Science Department, Indiana University, Bloomington IN, Tech. Rep., 2005.

[14] Y. L. Simmhan, B. Plale, D. Gannon, and S. Marru, "Performance evaluation of the karma provenance framework for scientific workflows," in *in: International Provenance and Annotation Workshop (IPAW)*. Springer, 2006, pp. 222–236.

[15] Mule esb. [retrieved: may, 2012]. [Online]. Available: http://www.mulesoft.org/what-mule-esb

[16] A. S. Foundation, "Apache axis2/java - next generation web services," Website http://ws.apache.org/axis2/, Jul. 2009.

[17] D. Koop, E. Santos, B. Bauer, M. Troyer, J. Freire, and C. T. Silva, "Bridging workflow and data provenance using strong links," in *Proceedings of the 22nd international conference on Scientific and statistical database management*, ser. SSDBM'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 397–415.

 ISBN: 978-1-61208-216-5

Enhancing Mobile Device Security by Security Level Integration in a Cloud Proxy

Thomas Ruebsamen, Christoph Reich
Hochschule Furtwangen University
Faculty of Computer Science
Furtwangen, Germany
{*Thomas.Ruebsamen, Christoph.Reich*}*@hs-furtwangen.de*

***Abstract*—Smartphones, tablets, laptops and other mobile devices dominate our every day life and became indispensable for many businessmen. But at the same time the number of security vulnerabilities have been increasing. To increase the security of such devices the paper proposes a proxy running in a cloud environment that controls the access for mobile device to applications, enterprise services or Internet services. The developed access management system based on the Role Based Access Control (RBAC) model has been extended by 5 security levels. These security levels are determined by a classification of the user, the communication channel, and the device itself.**

Keywords-security; cloud computing; mobile device; mobile security

I. INTRODUCTION

Since 2009 a drastic increase of reported security vulnerabilities and exploits in operating systems for mobile devices (e.g., Android, iOS, Windows Mobile, Symbian) can be observed. Attacks, especially those using viruses, worms and similar malware, have been relatively confined to desktop PCs, laptops and servers but are now more and more spreading into mobile platforms [1]. The main reason for this trend is their widely adopted usage and the fact that mobile devices are starting to become more and more similar to classic PC-like computers in terms of performance as well as field of application. A couple of years ago, mobile devices had a limited range of applications. Nowadays, expanding application stores and apps available for download, drastically have changed this. Mobile devices can easily be expanded in their functionality simply by installing new apps. A side-effect is the increased probability of being exposed by malware. With every new generation of mobile devices, especially in the smartphone and tablet sector, the performance regarding CPU, memory and network bandwidth is increased. This makes mobile devices an attractive target for attackers.

Another problem is the lack of security fixes for mobile system software. Manufacturers of mobile devices often fail to provide decent software-related support for their products. This is for example shown by the apparent version fragmentation which can be observed in the Android environment [2]. If the manufacturer does not provide its customers with software patches in time, devices become more vulnerable to exploits. Keeping the operating system and crucial software packages up to date is a well known best practice for securing PCs, yet regarding mobile devices this is often not possible due to lack of support. Using firewalls, anti virus scanners, spyware scanners, rootkit detectors and intrusion detection systems (IDS) on non mobile devices is not a common practice. Adopting such tools to mobile devices proves to be difficult, mainly because of lack of resources like battery longevity, computing power and storage.

Securing mobile devices has become one of the main concerns for companies, because they are adopting mobile devices for improving productivity of their employees. Their major concern is how to prevent attacks originating from compromised devices targeted on their corporate networks and their sensitive data.

To solve the problem of lacking resources on mobile devices, offloading resource intensive tasks to the cloud is one solution [3], [4]. Cloud Computing describes a technique where resources like computation power and storage are provided transparently over a network (usually the Internet). One major advantage of cloud computing is the relatively easy scaling of services. The results presented in this paper rely heavily on leveraging cloud computing especially for enabling scalability and providing sufficient resources to effectively enhance security of mobile devices. Such security mechanisms include but are not limited to anti virus, intrusion detection and application analysis in the cloud [5], [6].

In this paper a proxy, that controls the access of mobile devices to applications and services is proposed. The proxy is operated in the cloud which enables it to perform resource intensive analysis tasks. Also, the proxy is the central control component for evaluating the security as wells as the trustability of users, devices and communication channels. This results in the assignment of security levels which themselves are used to enable a more fine grained access control.

This paper is structured as follows: In this section, we gave an introduction to the security problems which occur with current mobile devices, such as smartphones and tablets in today's enterprise environments. In the next section, a security classification framework for mobile devices will be described. Based on this classification, we propose a security level model. In section IV, we will propose two different approaches for security level integration into the

 ISBN: 978-1-61208-216-5

RBAC model. Section V will highlight evaluation results of the proposed security level model and the classification framework using use cases. The last section includes the conclusion of this paper as well as future work.

II. Related Work

Portokalidis et al. [6] describe a system that implements an intrusion detection for Android based systems called *Paranoid Android*. Paranoid Android is based on a cloud deployment model where intrusion detection is offered as a service. By emulating whole devices in virtual machines in the cloud, it is possible to apply resource intensive anomaly detection mechanisms. This would not be possible to do on mobile devices because of the very limited available resources. The clone is kept in sync with the mobile device. Actions performed on the device are replayed by the emulator. They also show that it is imperative to optimize the synchronization and tracing processes because having to collect and transmit data can very easily lead to disproportionate exhaustion of the battery.

A very similar approach is taken by Zonouz et al. In their framework [7], [8] a lightweight agent is deployed on the mobile device, which collects user and sensor information. Additionally a proxy server is used to duplicate all traffic flowing between device and the internet. The collected traffic gets sent to an emulator in the cloud. Using the collected data from the agent and the proxy an analysis component scans for anomalies. In case of an ongoing attack the system informs the agent about countermeasures which need to be taken.

Andromaly [9] is a framework for detecting malware on mobile devices. Their approach is similar to those of classic host based intrusion detection systems. Android based devices are continuously monitored and attacks are detected using machine learning anomaly detectors. One of the main problems this approach are the limited resources on mobile devices which prevents the use of more sophisticated algorithms.

Schmidt et al. [10], [11] suggest using static analysis of executables as well as the integration of a collaborative system for detecting malware on Android based systems. By inspecting files on the function call level and comparing this data to already known malware files can be classified as harmful or harmless. The analysis can either be performed locally on the device or offloaded to a remote detection server. Additionally, devices can exchange analysis results with each other using the server. This leads to an improved detection rate.

Another approach, specifically targeted on the Symbian platform, is described by Bose et al. [12]. They are relying on behavioral analysis for detecting malware on mobile devices. Their idea is based on the assumption that a single action performed by an application can be classified as harmless, but in relation to other actions, which are performed in the same context, malware behavior can be exposed. Based on this assumption Bose et al. developed a database of behavioral signatures for malware. By training a support vector machine with normal behavior of applications, anomalies such as malware can be detected.

Kim et al. [13] analyze a very specific kind of malware causing battery exhaustion. These kind of attacks have already been described generally by Martin et al. in [14] and more specific by analyzing a security vulnerability in the MMS service by Radic et al. [15]. The core component of Kim's framework is a power monitor which monitors energy consumption and generates a power consumption profile. Using this profile it is possible to extract, analyze and detect attacks.

A very similar framework has been developed by Nash et al. [16]. Their system monitors mobile device parameters like CPU utilization and accesses to local storage to measure the used energy on a per process basis. Using this information they try to detect malware which tries to perform battery exhaustion attacks. This monitoring system is designed to be very lightweight. As an extension they suggest to start a fully-fledged intrusion detection system once an energy depletion attack has been detected.

Another kind of intrusion, especially theft, is the core concern of the work of Gupta et al. [17]. Basis for theft detection are profiles consisting of typing patterns and historic information like the history of made and received calls. Using this information the probability of the device being stolen or accessed by an unauthorized person is calculated. Unless there is no sufficient authentication of the user, data stored on the device remains encrypted. Additionally, if a theft has been detected a central management instance is notified.

The main differences of these approaches are whether the system is deployed on device or offloaded on a separate, dedicated system for analysis and detection. The system described in this paper uses a proxy server in the cloud for offloading most of the security related tasks. Also, the security level concept shares the same goals with the aforementioned projects, to enhance security and data protection in mobile enterprise environments.

III. Enhancing Mobile Security

To increase the security of mobile devices the access to services and data is controlled by a proxy running in a cloud (see Figure 1).

This architecture allows leveraging the advantages of cloud computing having almost unlimited computing power for analyzing the security status of the mobile system. The mobile security of the entire system depends basically on the security and trust level of $a)$ the user, $b)$ the mobile device, $c)$ the communication channel, and $d)$ the backend, the cloud. The proxy, which is under company control, is used to collect as much security related information

 ISBN: 978-1-61208-216-5

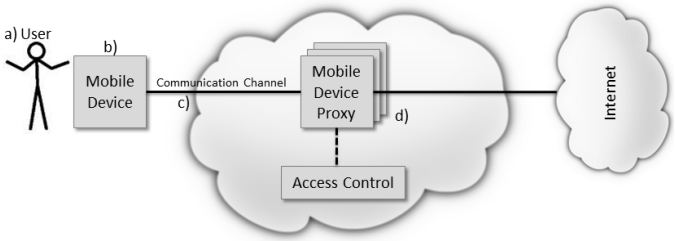


Figure 1. Mobile System Architecture with Cloud Proxy

about the aforementioned components as possible (e.g., by analyzing network traffic, querying company databases for organizational information et cetera). Additionally, the proxy requests information directly from the mobile device, which also monitors the aspects described in the following taxonomy. How this information is to be trusted (e.g., the device may send compromised data) is not in the scope of this paper, but will be part of our future work.

Before a detailed description on security levels will be made (see Section III-B), a mobile security taxonomy classifies the security domain.

A. Mobile Security Taxonomy

Figure 2 illustrates which properties have to be considered for mobile security devices into the categories: user, mobile device, communication channel, and backend.

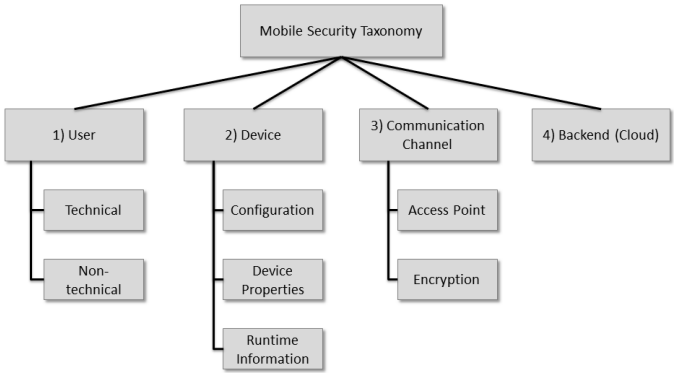


Figure 2. Mobile Security Taxonomy Overview

1) User: The user classification is targeted at the user of a mobile device. The primary function is to determine whether or not an authorized user is using the device. Mobile devices are inherently more prone to theft or unauthorized access because of their portability. Knowing that the owner or at least an authorized user is using the device is therefore crucial, when allowing access to sensitive data.

The second function is to evaluate used authentication mechanisms. For example, having no other authentication mechanisms in place apart from entering a PIN at device startup is very bad. There is no way to distinguish between an authorized user and for instance a thief. If there are other supported mechanisms like biometric identification in use, the user is more trustable, because of the stronger authentication.

Information which is used to evaluate the trustworthiness of a user and therefore assign a security level can be technical and non-technical:

Technical Information: These kinds of classification characteristics are strongly related to device and mobile operating software properties, especially supported authentication mechanisms.

The most simple is the support and usage of username/password combinations for additional user authentication. Most current mobile operating systems support at least authentication via an user-defined password.

Another way to enhance authentication is to use one-time passwords, which are generated on demand. These passwords are usually generated using special devices which are synchronized (based on the current time) with an authentication service. Requiring the user to be in possession of such a device reduces the risk the mobile device being used unauthorized. Of course, it can happen that both devices get stolen.

Similar to one-time password generators are dongles. Dongles are special devices linked to the mobile phone. By monitoring the proximity of the dongle, a mobile device can be locked and access denied until proximity is re-established.

A more sophisticated technical information is the support of biometric authentication by the mobile device. Devices which posses biometric scanners can achieve a better rating in user classification, assuming the biometric scanner and related software is tamper-proof.

Another way to gather information for user classification characteristics is to monitor location-related information. Many mobile devices have integrated GPS sensors. By tracking the location of a device and comparing it to a database such as an employee's schedule it could be detected whether it got stolen or not. Of course, GPS location and SSID are not 100% accurate, and further information is required.

Another way to identify a user is to make use of implicit authentication. Implicit authentication uses keystroke analysis and user action analysis to identify a user. In [17] a system is described which uses the analysis of typing patterns for theft detection.

Non-technical Information: Non-technical information is collected from internal company sources and usually contains information about the organizational structure. Hierarchical information can be used to classify users. For example temporary employees are usually less trustworthy than permanent ones. Management personnel might be more trustworthy than others and thus are allowed to access more sensitive data and services. Information about employees like the length of the affiliation with the company, profes-

 ISBN: 978-1-61208-216-5

sional trainings (e.g., mobile security awareness trainings) taken, can also be used to classify users. These kinds of information can be used to classify users as well as in downstream access control systems.

The combination of different characteristics, technical as well as non-technical ones, enables a more accurate picture of the person using the device.

2) Mobile Device: Security classification of devices is used to evaluate the security and trustworthiness of mobile devices from a technical point of view.

Configuration Monitoring: The configuration of mobile devices includes operating system versions, variants and patch levels as well as information about installed 3rd party apps. Using this information, which is usually supplied by mobile device management systems, it is possible to identify security risks, e.g. non-up-to-date software. An up-to-date system reduces the risks of security vulnerabilities. If there are serious security issues in older software versions of a mobile device a classification in higher security levels could be prohibited.

Device Properties: Mobile devices differ in their hardware configuration. Those features can make a difference in the security of a device, therefore the support and use of such device capabilities is also a factor in device classification. Such device properties include smart card support, which can be used to store digital certificates for authentication purposes, hardware implemented kill pills, for remotely wiping mobile devices, hardware supported encryption, which allows secure storage on mobile devices without putting too much of a burden on the CPU and biometric sensors, which can be used to realize secure and trustworthy authentication.

In the future, virtualization support on mobile devices will be a hot topic in terms of security. With virtualization building distinctly separated environments for parallel personal and business usage of the same device will be possible. This will improve security while handling corporate data and services on mobile devices.

Another characteristic is the mobile device operating systems. iOS and Android, for example, each support different security features and implement them differently. For example the implementation of process isolation or data encryption is done differently on those platforms.

Runtime Information: Runtime information includes collected information about current and historical resource utilization, like CPU load, memory utilization or battery utilization. Using this information, malware could be detected. Additionally, currently running processes and background services should be monitored. This information is sent to the proxy in regular intervals and is accounted for in device security evaluation.

The proxy can be used to collect additional information for security analysis. Network traffic, regardless if it is internal traffic to the corporate intranet or public traffic to the Internet, flows through the cloud-based proxy. This allows for traffic analysis tools to be used. By leveraging deep packet inspection for example, suspicious traffic generated by bots or trojans which communicate with their control instances, can be detected.

The proxy can also be used to create profiles of which network protocols are commonly used and how they are used (e.g., which service is usually used). Deviation from those profiles can be a sign of malware infection.

The proxy is the primary interface of the mobile device to security services like anti virus engines in the cloud. In case those services detect a potential threat, the proxy is informed and uses this information during device security evaluation. One special case is proxy connectivity itself and how it affects device security. It is very likely that there are periods of time where there is no connectivity between proxy and mobile device. This can be because of a GSM/UMTS dead zone or a lengthy stay abroad without data roaming. In these cases the duration between the last connection and the first one after that, must be considered during evaluation, mainly because the mobile device could have been tampered with. Usually, after such a period a mobile device should be regarded as untrusted until a full security check has been performed (either manually or by an automated process).

3) Communication Channel: The communication channel is a critical part of the security evaluation and the resulting security level classification. It is usually not under control of the company but the mobile network operator (MNO). The MNO's data services are used to connect to company intranets and the Internet. But there are also other communication channels (e.g., public access points) which need to be considered in a security evaluation, when accessing company data and services over such channels. The following characteristics have to be paid attention to:

1) *MNO data services* are generally not under the company's control, thus they are to be regarded as insecure. Connectivity to the proxy is established via the Internet using GSM or UMTS. The actual technical details of the network are hidden and usually there is no detailed technical information about the infrastructure and used technologies (e.g., whether and how NAT is used to connect mobile devices to the Internet) available to the MNO's customer.
2) *Public access points* like WLAN in public facilities are also not under the control of the company. Therefore, this communication channel must also be considered as insecure and untrusted.
3) *Known access points* include access points where there is technical information available and transparency is better than in public access points (e.g., the corporate WLAN infrastructure of a partner company). Depending on the actually available information, a better communication security classification is possible when using such access points.
4) *Internal access points* are under full control by the

 ISBN: 978-1-61208-216-5

company. There is full transparency about the technical infrastructure, technologies in use and implemented security measures. Using such access points allows maximum security.

Examining the access point alone is not a sufficient means of security measurement of the communication channel. In fact, the whole channel between mobile device, its access point, stations in between and the proxy has to be taken into consideration. This is especially the case, if a direct connection to proxy is not possible and the connection has to be established via the internet. Analyzing this problem further is out of the scope of this paper. Therefore, an end-to-end encrypted communication channel between mobile device and the proxy is assumed. Examining the security properties of stations in between becomes unnecessary in this case, if the end-to-end encryption is secure and reliable. End-to-end Encryption can be implemented in two ways:

1) *VPNs* are used to encrypt communication between communication partners. Using VPNs to encrypt traffic between mobile devices and their proxies provides maximum security, even if inherently or possibly insecure access points are used (see access points 1-3). In this case, communication security is depending on the security of the deployed VPN technology.
2) *Message encryption* is an alternative to VPNs. In this case not the whole communication is encrypted, but only relevant messages.

A special case is unencrypted communication between mobile device and proxy using an internal access point. This is the only case where it is possible to pass on using VPN or message encryption and still get a high communication security classification. Nevertheless, this is only possible if the communication channel between proxy and mobile device is fully transparent to the company and secure.

4) Backend (cloud): The backend security of the infrastructure, the cloud, with the proxy and the access control module, must be considered as well, but are traditional data center security issues and will not be considered in this paper.

Continuous evaluation of the mobile system based on the taxonomy, security levels can be assigned to each category which is later used for access control.

B. Security Levels

The aforementioned taxonomy influences the access control on company data and services in the cloud or the Internet. Based on continuous evaluation of the particular mobile system parts, security levels (see Figure 3) are assigned and integrated with the classic access control systems (e.g., RBAC, see section IV for further details) to allow fine grained protection of services and data. The overall security level of the mobile system is determined as following: For each part of the mobile system a security Level $Ln_{system\ part}$ where $n = 0, 1, 2, 3, 4$ is identified. The total mobile system security Level (Ln_{system}) is calculated by the minimum of all three single security levels, as stated in the following formula:

$$Ln_{system} = min(Ln_{user}, Ln_{device}, Ln_{communication})$$

with $n = 0, 1, 2, 3, 4$ levels of security.

The security levels either grant broader access rights or deny them. The main decisions, which need to be made are:

- Is the user of the mobile device authentic (has he been sufficiently authenticated)?
- Does the user have access to the requested data and services?
- Does the used mobile device pose a security risk?
- Is the communication channel between device and proxy or rather the requested data and services sufficiently secured?

In the following, the security level state diagram, transitions between security levels as well as mechanisms for applying security levels on access control decisions are described.

1) Security Level Definition: The following section describes the five identified security levels (see Figure 3), ordered by ascending security and trustability.

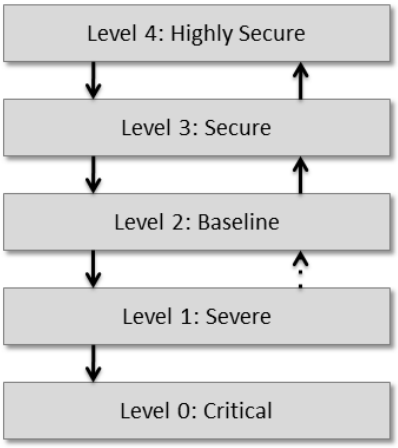


Figure 3. Security Levels

Level 0 (Critical):
The Level 0 security level is the lowest, which can be assigned by the classification process. In this case a highly critical security incident has occurred. If the user classification signals theft or loss of a device, it automatically gets assigned security Level 0. Access to the company's network, services and data is immediately and completely blocked. Further, the removal of all data on the device gets initiated (via a remote wipe of the device), as long there is still connectivity between the proxy and the device. Depending on the company's security policy for lost devices an agent on the device can either be directed to make the device

 ISBN: 978-1-61208-216-5

useless (blocking all communication channels) or to switch into surveillance mode, where GPS location, camera and video information is transmitted to the company for further investigation. This information can be used to to try to re-obtain the device or to initiate legal countermeasures. What has to be done, has to be defined in the company's security policy.

Level 1 (Severe):
The Level 1 security level presumes that the device is in possession of the legitimate owner. Theft or loss can be counted out. Anyhow, there still is a critical security incident. Usually, such incidents will be signaled by security services in the cloud. For example the anti virus system flags the device as compromised because of a detected malware infection or the intrusion detection system throws an alert because of an attempted or succeeded intrusion. Thus, the security classification of the device has failed. A failed security classification means that there is an incident, which is clearly critical. A user installing an unknown app on his device does not per se qualify for Level 1 assignment. Not until the app has been identified as a threat. Just like in Level 0 connectivity to data and services is severely limited and the device is cleaned.

Level 2 (Baseline):
The security level 2 is also known as baseline. User, device and communication classification have not detected a critical problem. All basic services are available and there is connectivity between the mobile device and its proxy in the cloud, but access to data and services is limited, because the full security and trustability of the device cannot be warranted. This can be because the user did not use a sufficiently strong authentication mechanism or there are additional unknown apps installed, which could potentially be dangerous. Another reason for Level 2 assignment is connecting via a public access point without sufficient additional security enhancements, like using a VPN. In Level 2 baseline services like e-mail, calendar and access to non-classified documents are enabled.

Level 3 (Secure):
For accessing confidential services and documents, an elevated security level is required. Level 3 builds upon the properties of Level 2, but requires additional security requirements. This comprises the usage of a VPN, the policy conform configuration of a device (e.g., only explicitly approved apps installed). Of course, the user has to be authenticated using a sufficiently strong mechanism (e.g., user and password combination).

Level 4 (Highly Secure):
The most restrictive security level is Level 4. It can only be assigned if classification attests full compliance to the security policy and additional security mechanisms are used. Such additional mechanisms can be the authentication of the user using biometric information, hardware supported full device encryption and connecting to the network using an internal access point. Only Level 4 allows access to highly confidential internal services and data.

2) Security Level Transitions: The assignment of a security level does not happen linearly. In the following, the transitions between security levels are described:

- L0 $\rightarrow$ L2 and upwards
  This transition describes the case, where a stolen or otherwise lost device gets regained. In this case, the device is not to be trusted and therefore, has to be classified as insecure and compromised. A full manual audit or a full reset by an administrator is needed for it to be assigned Level 2 or above. This evaluation process must not be automated, but be conducted by a qualified administrator.
- L1 $\rightarrow$ L2
  A compromised device has to be audited manually. Alternatively a full reset is also possible to reenter a secure state. This process must also be conducted manually by a qualified administrator.
- L2 $\leftrightarrow$ L3 $\leftrightarrow$ L4
  Transitions between these three security levels can happen automatically. For an assignment to the next higher security level, its security requirements must be fulfilled. For example deinstalling any not explicitly approved apps and connecting to the company's VPN can lead to the automatic upgrade from security Level 2 to 3.
- L2, L3, L4 $\rightarrow$ L1
  This downgrade usually happens when security services detect critical problem like a virus infection or an intrusion attempt. In this case, the user is informed about the incident and the Level 1 is immediately assigned.
- L* $\rightarrow$ L0
  Level 0 is assigned if a theft or loss of a device is detected.

IV. Access Control

The aforementioned classification of user, mobile device and communication channels, resulting in a security level of the overall system, has to be integrated into access control systems for services and data. This way classic access control can be enhanced with secure access control for mobile devices. The following section describes two approaches for integrating the security levels into the widely used role based access control model (RBAC) [18].

A. Role-based Access Control

Access control models serve the purpose of limiting access rights of authenticated subjects on certain objects. Subjects can be users, or programs which act on behalf of a user. All access attempts in a system are monitored and evaluated against a rule set of the access control model. This rule set describes which subjects are allowed to perform

which actions on objects protected by the system. One important aspect of classic access control models, like RBAC, is the distinction between authentication of a subject and the actual access control. Access control systems assume that a subject is properly authenticated before a decision about the authorization of an action is made [19]. Classic access control models are discretionary access control (DAC), mandatory access control (MAC) and role-based access control (RBAC). While DAC and MAC are important models, RBAC seems to be the more interesting model for the further discussion of integrating security levels. More modern approaches like the usage control model (UCON) have yet to prove their importance in real world systems. The fact that RBAC provides an abstraction of real world organizational structures, its wide adoption in software systems and the possibility to implement DAC as wells as MAC models simply by adjusting the RBAC model [20], made it the candidate of choice for further discussion.

The role-based access control model has been unified in 2000, based on the works of Ferraiolo, Kuhn and Sandhu and formally adopted as an ANSI Standard in 2004 [21]. This ANSI standard serves as a basis for further analysis. The basic elements of RBAC are users, roles, sessions and permissions. Users of a system are assigned one or more roles which they can assume. Depending on the role, access to subjects is either granted or denied. Users are assigned to roles using user assignments. Roles describe a function within an organization and the rights and obligations associated with it. Permissions describe operations which can be executed on RBAC-protected objects [21]. This is the foundation of the core RBAC model. Furthermore, there are some extensions to this core model which make it more flexible. One of these extensions is the RBAC 2 model, also called constrained RBAC. With this model it is possible to implement separation of duty concepts into the RBAC model. So called constraints allow a more fine-grained control over the RBAC model.

Despite of the RBAC model already being released as an ANSI standard, there is still research being conducted. Neumann and Strembeck [22] describe an extension to the RBAC 2 model, called context constraints. This type of constraints is used to evaluate predefined conditions at access control decision time. They allow the integration of RBAC model external conditions into the system. Thus, a context condition must be met, before an operation to which it is linked can be performed. One or more context conditions, which evaluate values of context attributes, form a context constraint. Apart from the roles and operations defined in the RBAC model, an unmet context constraint can prohibit the execution of an operation, which would otherwise be perfectly valid without context constraints. In comparison to the constrained RBAC model, Neumann and Strembeck enhance the concept of constraints in a way that makes them more generally applicable, especially the possibility of evaluating information from external databases (e.g., literally an external company database which contains employee records). An example for a context constraint is that users are allowed to access a certain document only in between 8am and 6pm, regardless of them assuming a role which has enough rights to do so or not.

1) RBAC Security Level Integration: Following the basics of the classic RBAC model this section will describe two ways of integrating security levels for mobile devices in RBAC. The first approach is based on extending the RBAC model with previously described context constraints while the second approach uses a two phase flow of access control.

Integration by RBAC Extension: This approach integrates security level requirements into RBAC using context constraints (see Figure 4 (S)ecurity (L)evel Check = Context Constraints). This means in addition to needing a specific role for accessing certain objects, a certain minimum security level is also required. The minimum security level is a context condition and the currently applied security level for a device is a context attribute. Together they form a context constraint which is bound to an operation. Before an actual access decision, based on the user, his role and the object, is made the context constraint is evaluated. If and only if the context constraint is met, the access control decision is made. During evaluation of the context condition, the current security level is pulled from the proxy (see Figure 4 step 3). The proxy is always informed about the currently applied security level. Is the current security level (context attribute) equal or higher than that defined in the context constraint of the object, the evaluation of the access control decision may continue. If the current security level is less than required, no further evaluation takes place and access is blocked.

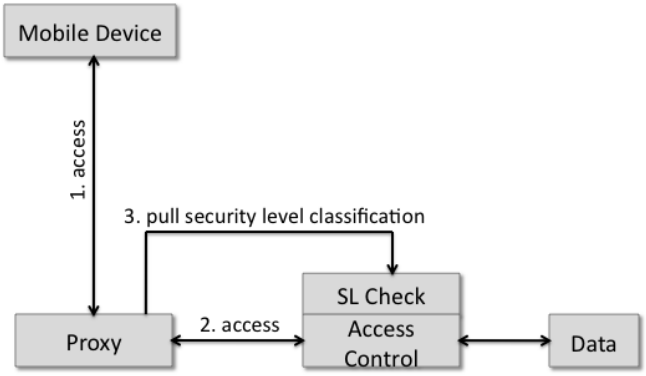


Figure 4. RBAC Security Level Integration - RBAC Extension

The main advantage of this approach is the tight integration of the security level concept into the RBAC model. But, there are also difficulties like the reduced flexibility by having to always integrate security levels into all access control systems which are in use. The pull mechanism during

the context constraint evaluation can also be a problem. As previously described, the assignment of a security level can change abruptly because of the continuous security evaluation of the user, device and communication channel. Therefore, with each access control decision, the current security level needs to be determined. This can happen quite often and thus degrade performance significantly depending on the deployment model (e.g., access control decision point needs to communicate with the proxy via a network).

Integration by Two Phase Flow: The two phase flow splits security level evaluation and actual access control decision into two phases. Figure 5 illustrates this concept. The proxy is the central component for accessing any services on the intranet and the internet and also stores the current security level assignment. Because of this it can easily be used to control security level evaluation. Every object (e.g., data object in Figure 5) possesses a minimum security level, which needs to be matched to gain access via a mobile device. This information is stored in the access control system. Usually, this information is very static and does not change too often. The proxy stores a copy of these object/minimum level mappings. If the required minimum level is changed, the updated mapping is pushed to the proxy. Now, if a mobile device is requesting access to access control protected data or services, the proxy first evaluates whether the minimum security level requirement is met or not. If it is, the request passes for further evaluation by the access control system, if not the request is refused by the proxy.

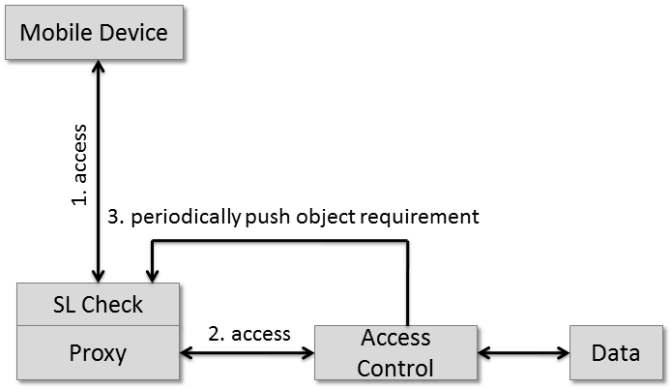


Figure 5. RBAC Security Level Integration - Two Phase Flow

This approach has the advantage of being access control model agnostic. It actually does not matter which system is used for access control, as long as object's requirements are available to the proxy. Also, the proxy can be used to terminate requests even before they reach the access control system located behind it. Pushing object requirements on update or in regular intervals also greatly reduces round trips during fetching of the current security level assignment. A problem of this approach is having to keep the same information (minimum requirements for objects) synchronized in two separate locations.

Both approaches have advantages and disadvantages. Higher flexibility, compatibility and a better communication flow are advantages of the two phase approach.

V. Evaluation by Use Case

In this section an evaluation of the proposed security level and classification concept is performed by using use cases for a better illustration. The general context of these use cases is the usage of mobile devices in a company. Sensitive documents may be stored on mobile devices. There is also an IT security policy in place, which sets the basic rules for using mobile devices (e.g., VPN, user authentication mechanisms, trusted software packages et cetera). Table I presents a selected overview of the most interesting use cases. The first column is used to describe preconditions (the state of security classification *before* a specific incident happens). The second column does the same for postconditions (the state of security classification *after* a specific incident happened). The overall security level classification is evaluated by choosing the current minimum security level of user, device or communication. Certain requirements for reaching a specific security level, e.g. having installed only known apps for reaching level 3 in device classification, is subject to concrete company security policies. Use cases 1 to 3 describe typical scenarios where the security level is lowered because of a security incident detected by the described system, whereas use cases 4 to 6 show how security level upgrades work.

The use cases show, how the security level concept for mobile devices allows to dynamically and continuously adjust their security classification. This allows a more controlled and more secure access of protected data as well as the overall improvement of the security of mobile devices in an enterprise environment.

VI. Conclusion and Future Work

In this paper, we demonstrated why security of mobile devices, like smartphones and tablets, in enterprise environments will be an important issue in the next couple of years. We also proposed a framework based on security levels and classification of user, device and communication which could improve security when handling confidential company data on such devices. Based on an ongoing classification security levels are applied and are evaluated during access on protected data. The integration of these two concepts into the well known RBAC model was also an important issue, discussed in this paper. We provide two possible solutions: one, which integrates tightly with the RBAC model using an extension called context constraints and another approach based on two-phase evaluation. Two-phase evaluation allows the decoupling of classic access control system and additional access control for mobile devices. At

 ISBN: 978-1-61208-216-5

Table I
EVALUATION OF THE SECURITY LEVEL CONCEPT FOR MOBILE DEVICES USING USE CASES

No.	Pre-Incident Security Level	Post-Incident Security Level	Use Case Description
1	min(L3, L3, L3) = L3	min(L0, L3, L3) = L0	The owner of the device is authenticated using PIN and additional username/password. His device adheres to general policies but has a game app installed, which is known to not contain malware but could lead to privacy problems. The connection to the proxy is established via UMTS using the company's VPN. Now, the device is stolen, while the owner is distracted. The thief fails three times to enter username/password correctly upon unlocking the screen. The mobile device agent reports this incident to the proxy which starts countermeasures according to the security policy for stolen devices.
2	min(L3, L4, L3) = L3	min(L3, L1, L3) = L1	The owner of the device is authenticated using PIN and additional username/password. His device configuration adheres strictly to the policies in place. Now, the user installs a new app from the app store. This app is scanned for malware in the cloud. The scanning engine detects a trojan inside the app and reports this incident to the proxy. The proxy starts countermeasures to protect the company's network.
3	min(L3, L4, L3) = L3	min(L3, L2, L3) = L2	Preconditions are the same as in the previous use case. The user installs an unknown app. The app is checked for malware without a positive result. To protect the company's data and network from a potential 0day-attack the security level is lowered.
4	min(L3, L4, L4) = L3	min(L4, L4, L4) = L4	The user is authenticated using username/password. The device's configuration matches security policy 100%. The connection to the proxy is established using the company's internal WLAN and VPN. Maximum security is guaranteed and there are no restrictions due to mobile access. The user now needs access to documents which are highly confidential an therefore require security level 4. The user now chooses to authenticate himself with additional biometric information using the fingerprint scanner. User classification is now upgraded to level 4, which enables an overall classification of 4, allowing access to the protected documents.
5	min(L3, L2, L3) = L2	min(L3, L3, L3) = L3	The user is authenticated using username/password. The device's configuration adheres to general security policy, but an unknown app is installed. The user now needs access to level 3 protected services. To achieve an upgrade, the user uninstalls the app. The proxy now registers that the unknown was removed and upgrades the device classification to level 3, resulting in an overall level 3 classification.
6	min(L4, L4, L2) = L2	min(L4, L4, L3) = L3	The user is authenticated using additional biometric information. The device's configuration matches security policy 100%. The connection to the proxy however is done using a public access point without using the company's VPN only relying on application based communication encryption (e.g. using IMAPS, HTTPS). The user needs access to internal documents requiring him to be classified as level 3. Therefore he establishes a secure VPN connection, which grants communication classification upgrade to level 3, resulting in an overall classification of 3.

last, we provide an evaluation of our approach which is used to demonstrate the feasibility using use cases.

In our future work we will concentrate on the process of securely collecting data (e.g. with the help of trusted infrastructure) about all the participants in our proposed framework and using that data for security classification. The proxy as well as data collected directly on the devices and the trustworthiness of the data will be in the center of our future examination. Another problem to solve will be data privacy protection, because of the very restrictive laws existing in Germany.

REFERENCES

[1] IBM Corporation, "IBM X-Force 2011 Mid-year Trend and Risk Report," https://www14.software.ibm.com/webapp/iwm/web/signup.do?source=swg-spsm-tiv-sec-wp&S_PKG=IBM-X-Force-2011-Mid-year, 2011, [retrieved: May, 2012].

[2] Android Developers, "Android Platform Versions - Current Distribution," http://developer.android.com/resources/dashboard/platform-versions.html, 2012, [retrieved: May, 2012].

[3] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in *Proceedings of the 12th conference on Hot topics in operating systems*, ser. HotOS'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 8–8.

[4] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 49–62.

[5] J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian, "Virtualized in-cloud security services for mobile devices," in *Proceedings of the First Workshop on Virtualization in Mobile Computing*, ser. MobiVirt '08. New York, NY, USA: ACM, 2008, pp. 31–35.

[6] G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid android: versatile protection for smartphones," in *Proceedings of the 26th Annual Computer Security Applications Conference*, ser. ACSAC '10. New York, NY, USA: ACM, 2010, pp. 347–356.

[7] S. A. Zonouz, K. R. Joshi, and W. H. Sanders, "Cost-aware systemwide intrusion defense via online forensics and on-

 ISBN: 978-1-61208-216-5

demand detector deployment," in *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*, ser. SafeConfig '10. New York, NY, USA: ACM, 2010, pp. 71–74.

[8] A. Houmansadr, S. Zonouz, and R. Berthier, "A cloud-based intrusion detection and response system for mobile phones," in *Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*, june 2011, pp. 31 –32.

[9] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "Andromaly: a behavioral malware detection framework for android devices," *Journal of Intelligent Information Systems*, pp. 1–30, 2011.

[10] A.-D. Schmidt, R. Bye, H.-G. Schmidt, J. Clausen, O. Kiraz, K. Yuksel, S. Camtepe, and S. Albayrak, "Static analysis of executables for collaborative malware detection on android," in *Communications, 2009. ICC '09. IEEE International Conference on*, june 2009, pp. 1 –5.

[11] A.-D. Schmidt, F. Peters, F. Lamour, and S. Albayrak, "Monitoring smartphones for anomaly detection," in *Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, ser. MOBILWARE '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 40:1–40:6.

[12] A. Bose, X. Hu, K. G. Shin, and T. Park, "Behavioral detection of malware on mobile handsets," in *Proceeding of the 6th international conference on Mobile systems, applications, and services*, ser. MobiSys '08. New York, NY, USA: ACM, 2008, pp. 225–238.

[13] H. Kim, J. Smith, and K. G. Shin, "Detecting energy-greedy anomalies and mobile malware variants," in *Proceeding of the 6th international conference on Mobile systems, applications, and services*, ser. MobiSys '08. New York, NY, USA: ACM, 2008, pp. 239–252.

[14] T. Martin, M. Hsiao, D. Ha, and J. Krishnaswami, "Denial-of-service attacks on battery-powered mobile computers," in *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, march 2004, pp. 309 – 318.

[15] R. Racic, D. Ma, and H. Chen, "Exploiting mms vulnerabilities to stealthily exhaust mobile phone's battery," in *Securecomm and Workshops, 2006*, 28 2006-sept. 1 2006, pp. 1 –10.

[16] D. Nash, T. Martin, D. Ha, and M. Hsiao, "Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices," in *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, march 2005, pp. 141 – 145.

[17] A. Gupta, D. Gupta, and N. Gupta, "Infosec-mobcop - framework for theft detection and data security on mobile computing devices," in *Contemporary Computing*, ser. Communications in Computer and Information Science, S. Ranka, S. Aluru, R. Buyya, Y.-C. Chung, S. Dua, A. Grama, S. K. S. Gupta, R. Kumar, and V. V. Phoha, Eds. Springer Berlin Heidelberg, 2009, vol. 40, pp. 637–648.

[18] D. Ferraiolo and R. Kuhn, "Role-based access control," in *In 15th NIST-NCSC National Computer Security Conference*, 1992, pp. 554–563.

[19] R. S. Sandhu and P. Samarati, "Access control: Principles and practice," *IEEE Communications Magazine*, vol. 32, pp. 40–48, 1994.

[20] S. Osborn, R. Sandhu, and Q. Munawer, "Configuring role-based access control to enforce mandatory and discretionary access control policies," *ACM Trans. Inf. Syst. Secur.*, vol. 3, pp. 85–106, May 2000.

[21] American National Standard Institue Inc., "American National Standard for Information Technology - Role Based Access Control," http://profsandhu.com/journals/tissec/ANSI+INCITS+359-2004.pdf, 2004, [retrieved: May, 2012].

[22] G. Neumann and M. Strembeck, "An Approach to Engineer and Enforce Context Constraints in an RBAC Environment," in *In Proc. of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM Press, 2003, pp. 65–79.

 ISBN: 978-1-61208-216-5

Maximizing Utilization in Private IaaS Clouds with Heterogenous Load

Tomáš Vondra, Jan Šedivý
Dept. of Cybernetics, Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27
Prague, Czech Republic
vondrto6@fel.cvut.cz, sedivja2@fel.cvut.cz

***Abstract*—This document presents ongoing work on creating a computing system that can run two types of workloads on a private cloud computing cluster, namely web servers and batch computing jobs, in a way that would maximize utilization of the computing infrastructure. The idea stems from the experience with the Eucalyptus private cloud system, which is used for cloud research at the Dept. of Cybernetics. This cloud lets researchers use spare computing power of lab computers with the help of our in-house queue engine called Cloud Gunther. This application improves upon current practices of running batch computations in the cloud by integrating control of virtual machine provisioning within the job scheduler. In contrast to other similar systems, it was built with the capacity restrictions of private clouds in mind. The Eucalyptus system has also been evaluated for web server use, and the possibility of dynamically changing the number of servers depending on user demand, which changes throughout the day, has been validated. Although there are already tools for running interactive services in the cloud and tools for batch workloads, there is no tool that would be able to efficiently distribute resources between these two in private cloud computing environments. Therefore, it is difficult for the owners of private clouds to fully exploit the potential of running heterogenous load while keeping the utilization of the servers at optimal levels. The Cloud Gunther application will be modified to monitor the resource consumption of interactive traffic in time and use that information to efficiently fill the remaining capacity with its batch jobs, therefore raising the utilization of the cluster without disrupting interactive traffic.**

Keywords-Cloud Computing; Automatic Scaling; Job Scheduling; Real-time Infrastucture.

I. INTRODUCTION

According to Gartner [1], private cloud computing is currently at the top of the technology hype; but, its popularity is bound to fall due to general disillusionment.

Why? While the theoretical advantages of cloud computing are widely known – private clouds build on the foundations of virtualization technology and add automation, which should result in savings on administration while improving availability, they provide elasticity, which means that an application deployed to the cloud can dynamically change the amount of resources it uses, which is connected to agility, meaning that the infrastructure can be used for multiple purposes depending on current needs. Lastly, the cloud should provide self-service, so that the customer can provision his infrastructure at will, and pay-per-use, so he will pay exactly for what he consumed.

The problem is that not all of these features are present in current products that are advertised as private clouds. Specifically, this document will deal with the problem of infrastructure agility.

A private cloud can be used for multiple tasks, which all draw resources from a common pool. This heterogenous load can basically be broken down into two parts, interactive processes and batch processes. An example of the first are web applications, which are probably the major way of interactive remote computer use nowadays, the second could be related to scientific computations or, in the corporate world, data mining.

When building a data center, which of course includes private clouds, the investor will probably want to ensure that it is utilized as much as possible. The private cloud can help achieve that, but not when the entire load is interactive. This is due to the fact that interactive load depends on user activity, which varies throughout the day, as seen in Figure 1.

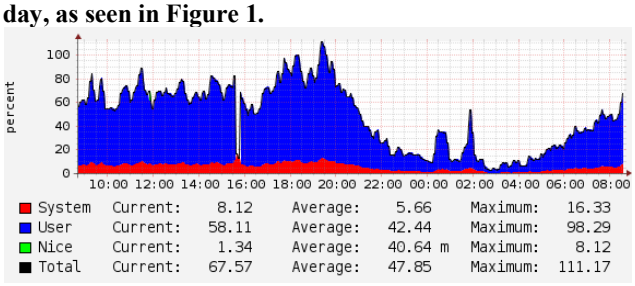


Figure 1. Daily load graph of an e-business website [2]

In our opinion, the only way to increase the utilization of a private cloud is to introduce non-interactive tasks that will fill in the white parts of the graph, i.e., capacity left unused by interactive traffic (which of course needs to have priority over batch jobs).

HPC (High Performance Computing) tasks are traditionally the domain of grid computing. Lately, however, they also began to find their way into the cloud. Examples may be Google's data mining efforts in their private cloud or Amazon's Elastic MapReduce public service [16]. The grid also has the disadvantage that it is only usable for batch and parallel jobs, not interactive use.

 ISBN: 978-1-61208-216-5

Currently, there is not much support for running of batch jobs on private clouds. The well known scheduling engines Condor [17] and SGE (Sun Grid engine) [18] both claim Amazon EC2 (Elastic Compute Cloud) [19] compatibility, they however cannot control the cloud directly, they only use resources provisioned by other means (See Section II.). (SGE seems to be able to control cloud instances in a commercial fork by Univa, though [3].)

That is why the Cloud Gunther project was started. It is a web application that can run batch parallel and pseudoparallel jobs on the Eucalyptus private cloud [4]. The program does not only run tasks from its queue; it can also manage the VM (virtual machine) instances the tasks are to be run on.

What the application currently lacks is support for advanced queuing schemes (only Priority FCFS (First Come First Served) has been implemented). Further work will include integration of a better queuing discipline, which will be capable of maximizing utilization of the cloud computing cluster by reordering the tasks as to reduce the likelihood of one task waiting for others to complete, while there are unused resources in the cluster, effectively creating a workflow of tasks (see Section IV).

The scheduler will be fed with data about the average amount of free resources left on the cluster by interactive processes. This will ensure that the cluster is always fully loaded, but the interactive load is never starved for resources.

This document has five sections. After Section I, Introduction, comes Section II, Related Work, which will present the state of the art in the area of grid schedulers and similar cloud systems. Section III, Completed Work, summarizes progress done in cloud research at the Dept. of Cybernetics, mainly the Cloud Gunther job scheduler. Section IV, Future Work, outlines the plans for expansion of the scheduler, mainly to accommodate heterogenous load on the cloud computing cluster. Section V, Conclusion, ends the paper.

II. RELATED WORK

As already stated, the most notable job control engines in use nowadays are probably SGE [18] and Condor [17]. These were developed for clusters and thus lack the support of dynamic allocation and deallocation of resources in cloud environments.

There are tools that can allocate a complete cluster for these engines, for example StarCluster for SGE [9]. The drawback of this solution is that the management of the cloud is split in two parts – the job scheduler, which manages the instances currently made available to it (in an optimal fashion, due to the experience in the grid computing field), and the tool for provisioning the instances, which is mostly manually controlled.

This is well illustrated in an article on Pandemic Influenza Simulation on Condor [10]. The authors have written a web application which would provision computing resources from the Amazon cloud and add them to the Condor resource pool. The job scheduler could then run tasks on them. The decision on the number of instances was however left to the users.

A similar approach is used in the SciCumulus workflow management engine, which features adaptive cloud-aware scheduling [11]. The scheduler can react to the dynamic environment of the cloud, in which instances can be randomly terminated or started, but does not regulate their count by itself.

The Cloud Gunther does not have this drawback, as it integrates job scheduling with instance provisioning. This should guarantee that there is no unused time between the provisioning of a compute resource and its utilization by a task, and that the instances are terminated immediately when they are no longer needed.

A direct competitor to Cloud Gunther is Cloud Scheduler [13]. From the website, it seems to be a plug-in for Condor which can manage VM provisioning for it. Similarly to Cloud Gunther, it is fairly new and only features FCFS queuing.

An older project of this sort is Nephele [14], which focuses on real-time transfers of data streams between jobs that form a workflow. It provisions different-sized instances for each phase of the workflow. In this system, the number and type of machines in a job are defined upfront and all instances involved in a step must run at once, so there is little space for optimization in the area of resource availability and utilization.

Aside from cluster-oriented tools, desktop grid systems are also reaching into the area of clouds. For example, the Aneka platform [12] can combine resources from statically allocated servers, unused desktop computers and Amazon Spot instances. It can provision the cloud instances when they are needed to satisfy job deadlines. This system certainly seems more mature than Cloud Gunther and has reached commercial availability.

None of these systems deals with the issue of resource availability in private clouds and fully enjoy the benefits of the illusion of infinite supply. To the best of our knowledge, no one has yet dealt with the problem of maximizing utilization of a cloud environment that is not fully dedicated to HPC and where batch jobs would have the status of “filler traffic”.

III. COMPLETED WORK

A. Eucalyptus

Eucalyptus [4] is the cloud platform that is used for experiments at the Dept. of Cybernetics. It is an open-source implementation of the Amazon EC2 industry standard API (Application Programming Interface) [19]. It started as a research project at the University of California and evolved to a commercial product.

It is a distributed system consisting of five components. Those are the Node Controller (NC), which is responsible of running virtual machines from images obtained from the Walrus (Amazon S3 (Simple Storage Service) implementation). Networking for several NCs is managed by a Cluster Controller (CC), and the Cloud

 ISBN: 978-1-61208-216-5

Controller (CLC) exports all external APIs and manages the cloud's operations. The last component is the Storage Controller (SC), which exports network volumes, emulating the Amazon EBS (Elastic Block Store) service. The architecture can be seen in Figure 2.

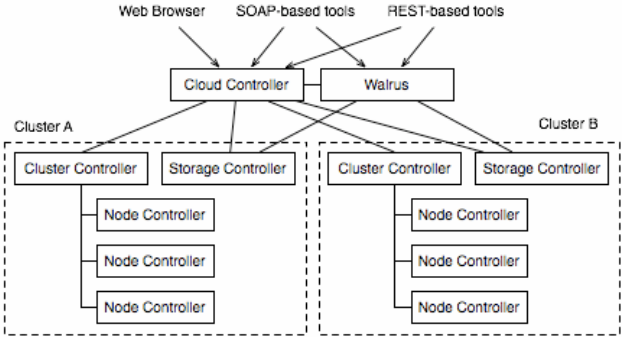


Figure 2. Eucalyptus architecture [4]

Our Eucalyptus setup consists of a server that hosts the CLC, SC and Walrus components and is dedicated to cloud experiments. The server manages 20 8-core Xeon workstations, which are installed in two labs and 1/4 of their capacity can be used for running VM instances through Eucalyptus NCs. A second server, which is primarily used to provide login and file services to students and is physically closer to the labs, is used to host Eucalyptus CC.

The cloud is used for several research projects at the Cloud Computing Center research group [5]. Those are:

- Automatic deployment to PaaS (Platform as a Service), a web application capable of automatic deployment of popular CMS (Content Management Systems) to PaaS.
- Effective scaling in private IaaS (Infrastructure as a Service), a diploma thesis on adding automatic scaling and load balancing support for web applications in private clouds.
- Cloud Gunther, a web application that manages a queue of batch computational jobs and runs them on Amazon EC2 compatible clouds.

Aside from this installation of Eucalyptus, we also have experience deploying the system in a corporate environment. An evaluation has been carried out in cooperation with the Czech company Centrum. The project validated the possibility of deploying one of their production applications as a machine image and scaling the number of instances of this image depending on current demand. A hardware load-balancer appliance from A10 Networks was used in the experiment and the number of instances was controlled manually as private infrastructure clouds generally lack the autoscaling capabilities of public clouds.

B. Cloud Gunther

While the Effective scaling in private IaaS project will also be instrumental for further research, it is only just starting. In contrast, the Master's thesis on Cloud Gunther has already been defended; the possibilities for its further development are the main topic of this article.

The application is written in the Ruby on Rails framework and offers both interactive and REST (Representational State Transfer) access. It depends on Apache with mod_passenger, MySQL and RabbitMQ for operation.

It can control multiple Amazon EC2 [19] compatible clouds. The queuing logic resides outside the MVC (Model, View, Controller) scheme of Rails, but shares database access with it. The communication scheme is on Fig. 3.

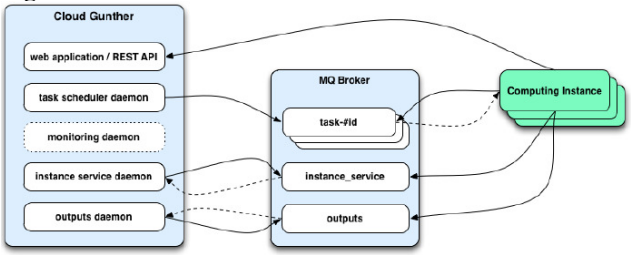


Figure 3. Communication scheme in Cloud Gunther [6]

The Scheduler daemon contains the Priority FCFS queuing discipline and is responsible for launching instances and submitting their job details to the message broker. The Agent on the instance then retrieves these messages and launches the specified user algorithm with the right parameters. It is capable of running multiple jobs from the same user, thus saving the overhead of instance setup and teardown.

The two other daemons are responsible for collecting messages from the queue, which are sent by the instances. The Instance Service serves to terminate instances, which have run out of jobs to execute; the Outputs daemon collects standard and error outputs of user programs captured by the launching Agent. A Monitoring daemon is yet to be implemented.

The web application itself fulfills the requirement of multitenancy by providing standard user login capabilities. The users can also be categorized into groups, which have different priorities in the scheduler.

The cloud engine credentials are shared for each cloud (for simpler cloud access via API and instance management via SSH (Secure Shell)).

Each cloud engine has associated images for different tasks, eg. image for Ruby algorithms, image for Java, etc. The images are available to all users, however when launched, each user will get his own instance.

The users can define their algorithm's requirements, i.e., which image the algorithm runs on and what instance size it needs. There is also support for management of different versions of the same algorithm. They may only differ in command line parameters, or each of them may have a binary program attached to it, which will be uploaded to the instance before execution.

Individual computing tasks are then defined on top of the algorithms. The task consists of input for the algorithm, which is interpolated into its command line

 ISBN: 978-1-61208-216-5

with the use of macros, as well as the instance index and total count of instances requested. These values are used by pseudoparallel algorithms to identify the portion of input data to operate on, and by parallel algorithms for directing communication in message passing systems.

As one can see in Figure 4., the system is ready for private clouds. It can extract the amount of free resources from Eucalyptus and the scheduler takes it into account when launching new instances.

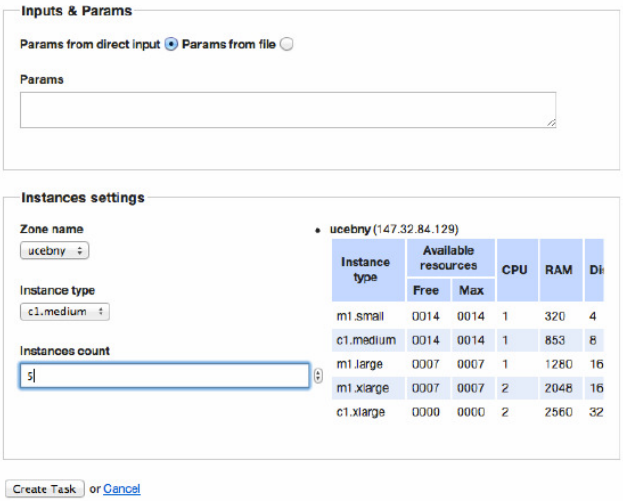


Figure 4. Cloud Gunther – part of the New Task screen [6]

The Cloud Gunther has been tested on several real workloads from other scientists. Those were production planning optimization, recognition of patterns in images and a multiagent simulation. They represented a parameter sweep workflow, a pseudoparallel task and a parallel task, respectively.

VM images for running the tasks were prepared in cooperation with the users. Usability was verified by having the users set up algorithm descriptions in the web interface. The program then successfully provisioned the desired number of VM instances, executed the algorithms on them, collected the results and terminated the instances.

The main drawback, from our point of view, is that when there are jobs in the queue, the program consumes all resources on the cluster.

This is not a problem in the experimental setting, but in a production environment, which would be primarily used for interactive traffic, and would attempt to exploit the agility of cloud infrastructure to run batch jobs as well, this would be unacceptable.

In such a setting, the interactive traffic needs to have absolute priority. For example, if there was a need to increase the number of web servers due to a spike in demand, then in the current state, the capacity would be blocked by Cloud Gunther until some of its tasks finished. It would be possible to terminate them, but that would cause loss of hours of work. A proactive solution to the heterogenous load situation is needed.

IV. FUTURE WORK

Future work planned on the Cloud Gunther can be split into two categories. First and more important is the consideration of interactive load also present on the cluster, see Subsection A. Second is integration of better queuing disciplines to bring it up to par with existing cluster management tools. Two ideas for that are presented in Subsections B and C.

A. Estimation of the amount of interactive load in time

The interactive traffic needs to have priority over the batch jobs. Therefore, once work is completed on the general purpose autoscaler for private IaaS, it will be possible to record the histogram of the number of instances that the autoscaler is managing. From this histogram, data on daily, weekly and monthly usage patterns of the web servers may be extracted and used to set the amount of free resources for Cloud Gunther.

The vision on the extraction method is that it will employ machine learning techniques to approximate the statistical distribution of the number of web server instances at any hour of the year, probably breaking it up to yearly, monthly, weekly and daily curves.

Instead of seeing only the current amount of free resources in the cloud, the batch job scheduler could be able to ask: "May I allocate 10 large instances to a parallel job for the next 4 hours with 90% probability of it not being killed?"

A similar problem exists in desktop grids. Article [15] illustrates the collection of availability data from a cluster of desktop machines and presents a simulation of predictive scheduling using this data. The abstraction of the cloud will shield away the availability of particular machines or their groups, the only measured quantity will be the amount of available VM slots of a certain size.

B. Out-of-order scheduling

This of course assumes a scheduler that will be capable of using this information. Our vision is a queue discipline that internally constructs a workflow out of disparate tasks. The tasks, each with an associated estimate of duration, will be reordered so that the utilization of the cloud is maximized.

For example, when there is a job currently running on 20 out of 40 slots and should finish in 2 hours, and there is a 40 slot job in the queue, it should try to run several smaller 2 hour jobs to fill the free space, but not longer, since that would delay the large job.

These requirements almost exactly match the definition of the Multiprocessor scheduling problem (see [8]). Since this is a NP-hard class problem, solving it for the whole queue would be costly. The most feasible solution seems to come from the world of out-of-order microprocessor architectures, which re-order instructions to fully utilize all execution units, but only do so with the first several instructions of the program. The batch job scheduler will be likewise able to calculate the exact solution with the first several jobs in the queue, which will otherwise remain Priority FCFS.

 ISBN: 978-1-61208-216-5

C. *Dynamic priorities*

The estimation of job duration is a problem all for itself. At first, the estimate could be done by the user. Later, a system of dynamic priorities could be built on top of that.

The priorities would act at the level of users, penalizing them for wrong estimates, or better, suspending allocation of resources to users whose tasks have been running for longer time than the scheduler thought.

Inspiration for this idea is taken from the description of the Multilevel Feedback Queue scheduler used historically in Linux [7]. However, the scheduler will set priorities for users, not processes, and allocate VMs to tasks, not jiffies to threads. It also will not have to be real-time and preemptive, making the design simpler.

The scheduler's estimate of process run time could be based on the user estimates, but also on the previous run time of processes from the same task or generally those submitted by the same user for the same environment. That would lead to another machine learning problem.

V. CONCLUSION

The cloud presents a platform that can join two worlds that were previously separate – web servers and HPC grids. The public cloud, which offers the illusion of infinite supply of computing resources, will accommodate all the average user's needs, however, new resource allocation problems arise in the resource-constrained space of private clouds.

We have experience using private cloud computing clusters both for running web services and batch scientific computations. The challenge now is to join these two into a unified platform.

Currently, Cloud Gunther, although not ready for commercial deployment, already has some state of the art features, like the automatic management of cloud computing instances and a REST-compliant web interface. It also differs from other similar tools by its orientation towards private cloud computing clusters.

In the future, it could become a unique system for managing batch computations in a cloud environment primarily used for web serving, thus allowing to exploit the dynamic nature of private cloud infrastructure and to raise its overall utilization.

ACKNOWLEDGMENTS

Credit for the implementation of Cloud Gunther, mainly the user friendly and cleanly written web application goes to Josef Šín.

We thank the company Centrum for providing material for our experiment and insights on private clouds from the business perspective.

REFERENCES

[1] D. M. Smith, "Hype Cycle for Cloud Computing," Gartner, 27 July 2011, G00214915.

[2] T. Vondra and J. Šedivý, "Od hostingu ke cloudu," Research Report GL 229/11, CTU, Faculty of Electrical Engineering, Gerstner Laboratory, Prague, 2011, ISSN 1213-3000.

[3] T. P. Morgan, "Univa skyhooks grids to clouds: Cloud control freak meets Grid Engine," The Register, 3rd June 2011, <http://www.theregister.co.uk/2011/06/03/univa_grid_engine_cloud/> 19 March 2012.

[4] "Installing Eucalyptus 2.0," Eucalyptus, <http://open.eucalyptus.com/wiki/EucalyptusInstallation_v2.0> 19 March 2012.

[5] J. Šedivý, "3C: Cloud Computing Center," CTU, Faculty of Electrical Engineering, dept. of Cybernetics, Prague, <https://sites.google.com/a/3c.felk.cvut.cz/cloud-computing-center-preview/> 19 March 2012.

[6] J. Šín, "Production Control Optimization in SaaS," Master's Thesis, CTU, Faculty of Electrical Engineering and University in Stavanger, Department of Electrical and Computer Engineering, Supervisors J. Šedivý and C. Rong, Prague, 20 December 2011.

[7] T. Groves, J. Knockel, E. Schulte, "BFS vs. CFS - Scheduler Comparison," 11 December 2011 < http://slimjim.cs.unm.edu/~eschulte/data/bfs-v-cfs_groves-knockel-schulte.pdf > 11 May 2012.

[8] "Multiprocessor scheduling," in Wikipedia: the free encyclopedia, San Francisco (CA): Wikimedia Foundation, 12 March 2012, <http://en.wikipedia.org/wiki/Multiprocessor_scheduling> 19 March 2012.

[9] "StarCluster," Massachusetts Institute of Technology, < http://web.mit.edu/star/cluster/index.html> 11 May 2012.

[10] H. Eriksson, et al., "A Cloud-Based Simulation Architecture for Pandemic Influenza Simulation," AMIA Annu Symp Proc. 2011; 2011: 364–373, pp. 364–373.

[11] D. de Oliveira, E. Ogasawara, K. Ocaña, F. Baião and M. Mattoso, "An adaptive parallel execution strategy for cloud-based scientific workflows," Concurrency Computat.: Pract. Exper. (2011), doi: 10.1002/cpe.1880.

[12] R. N. Calheiros, C. Vecchiola, D. Karunamoorthya and R. Buyya, "The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds," Future Generation Computer Systems 28 (2012), pp. 861-870, doi: 10.1016/j.future.2011.07.005.

[13] "Cloud Scheduler," University of Victoria, <http://cloudscheduler.org/> 11 May 2012.

[14] D. Warneke and O. Kao, "Nephele: efficient parallel data processing in the cloud," MTAGS '09: Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, November 2009, doi: 10.1145/1646468.1646476.

[15] K. Ramachandran, H. Lutfiyya and M. Perry, "Decentralized approach to resource availability prediction using group availability in a P2P desktop grid," Future Generation Computer Systems 28 (2012), pp. 854–860, doi: 10.1109/CCGRID.2010.54.

[16] R. Grossman and Y. Gu, "Data mining using high performance data clouds: experimental studies using sector and sphere," In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08). ACM, New York, NY, USA, 2008, pp. 920-927, doi: 10.1145/1401890.1402000.

[17] M. J. Litzkow, M. Livny and M. W. Mutka, "Condor-a hunter of idle workstations," 8th International Conference on Distributed Computing Systems (1988), pp. 104-111.

[18] W. Gentzsch, "Sun Grid Engine: towards creating a compute power grid," Proceedings of the first IEEE/ACM International Symposium on Cluster Computing and the Grid (2001), pp. 35-36.

[19] "Amazon Elastic Compute Cloud (EC2) Documentation," Amazon, <http://aws.amazon.com/documentation/ec2/> 27 May 2012

 ISBN: 978-1-61208-216-5

Defining Inter-Cloud Architecture for Interoperability and Integration

[1]Yuri Demchenko, [1]Canh Ngo, [1,2]Marc X. Makkes, [1,2]Rudolf Strijkers, [1]Cees de Laat
[1]University of Amsterdam
System and Network Engineering Group
Amsterdam, The Netherlands
e-mail: {y.demchenko, c.t.ngo, delaat}@uva.nl

[2]TNO
Information and Communication Technology
Groningen, The Netherlands
e-mail: {marc.makkes, rudolf.strijkers}@tno.nl

***Abstract*—This paper presents an on-going research to develop the Inter-Cloud Architecture, which addresses the architectural problems in multi-provider multi-domain heterogeneous cloud based applications integration and interoperability, including integration and interoperability with legacy infrastructure services. Cloud technologies are evolving into a common way to virtualize infrastructure services and to offer on-demand service provisioning. In this way, they add physical/hardware platform independency and mobility to the existing distributed computing and networking technologies. The paper uses existing standards in Cloud Computing, in particular the recently published NIST Cloud Computing Reference Architecture (CCRA) as the basis for the Inter-Cloud architecture. The proposed Inter-Cloud Architecture defines three complimentary components addressing Inter-Cloud interoperability and integration: multi-layer Cloud Services Model that combines commonly adopted cloud service models, such as IaaS, PaaS, SaaS, in one multilayer model with corresponding inter-layer interfaces; Inter-Cloud Control and Management Plane that supports cloud based applications interaction; and Inter-Cloud Federation Framework. The paper briefly presents the architectural framework for cloud based infrastructure services provisioning being developed by the authors. The proposed architecture intends to provide a basis for building multilayer cloud services integration framework and to allow optimised provisioning of computing, storage and networking resources. In this way, the proposed Inter-Cloud architecture will facilitate cloud interoperability and integration.**

Keywords-Inter-Cloud Architecture; Cloud Computing Reference Architecture; Architectural framework for cloud infrastructure services provisioned on-demand; Cloud middleware.

I. INTRODUCTION

Cloud computing technologies [1, 2] are emerging as infrastructure services for provisioning computing and storage resources on-demand in a simple and uniform way and may involve multi-provider and multi-domain resources, including integration with the legacy services and infrastructures. Cloud computing represents a new step in evolutional computing and communication technology development by introducing a new abstraction layer for general virtualisation of infrastructure services (similar to utilities) and mobility. Current developments in cloud technologies demonstrate the need to (1) develop an Inter-Cloud architecture that provides a common/interoperable environment and definition for moving existing infrastructures and infrastructure services into cloud environments and (2) integration tools to include existing enterprise and campus infrastructures. More complex use of cloud infrastructure services, such as in multi-domain enterprise environments, require new service provisioning and security models that allow on-demand provisioning of complex project and group-oriented infrastructure services across multiple providers.

Cloud based virtualisation enables easy upgrade and/or migration of enterprise application, including also the whole Information Technology (IT) infrastructure segments with automation or infrastructure management tools This brings significant cost savings compared to traditional infrastructure development and management, which requires lot of manual work. In particular, applications that use modern SOA (Service Oriented Architecture) web services platforms for services and integration benefit from cloud based infrastructure services, such as elastic scaling and on-demand provisioning. However, their composition and integration into distributed cloud based infrastructure will require a number of functionalities and services that can be jointly defined as Inter-Cloud Architecture.

This paper presents an on-going research at the University of Amsterdam to develop the Inter-Cloud Architecture (ICA). The Inter-Cloud architecture addresses the problem of (1) multi-domain heterogeneous cloud based applications integration and interoperability, including integration and interoperability with legacy infrastructure services, and (2) intra-provider infrastructure interoperability and measurability, and (3) cloud federation. The papers refers to the architectural framework for provisioning Cloud Infrastructure Services On-Demand [3] being developed by authors as a result of cooperative efforts in a number of currently running projects such as GEANT3 [4] and GEYSERS [5]. The architectural framework provides a basis for defining the proposed Inter-Cloud architecture. The presented paper significantly extends the research results initially presented as a poster paper at the IEEE CloudCom2011 Conference [6].

 ISBN: 978-1-61208-216-5

The remainder of the paper is organized as follows. Section II provides overview and detailed analysis of the ongoing standardisation activities at NIST and IEEE that have direct relation with and provide a basis for the proposed ICA. Section III describes a basic use case for defining ICA, and section provides motivation and defines the main components of the proposed Inter-Cloud Architecture. In Sections IV the Inter-Cloud definition and requirements are described. Section V describes the abstract model for cloud based infrastructure services provisioning. Section VI describes the Infrastructure Services Modeling Framework that provides a basis for complex infrastructure services composition and management. The paper concludes with future developments in Section VII.

II. CLOUD STANDARDISATION OVERVIEW

Two standardization activities form the basis of ICA and will be analysed in detail. One, the Cloud Computing technology and Cloud Computing Reference Architecture definition by the National Institute of Standards and Technology (NIST) and two, the IEEE standardisation activity to define Intercloud Interoperability and Federation framework. Suggestions are provided for the required extensions in the context of the proposed Inter-Cloud Architecture.

An overview of the standards that define internal cloud management, components design and communications is left out. This category of standards is well presented by DMTF, SNIA and OGF standards that correspondingly define standards for Open Virtual Machine Format (OVF) [7], Cloud Data Management Interface (CDMI) [8], and Open Cloud Computing Interface (OCCI) [9]. These standards are commonly accepted by industry and provide a basis for intra-provider infrastructure operation and services delivery to customers.

A. NIST Cloud Computing related standards

NIST is active in fostering cloud computing practices that support interoperability, portability, and security requirements that are appropriate and achievable for important usage scenarios. Since first publication of the currently commonly accepted NIST Cloud definition in 2008, NIST is leading the internationally recognised activity on defining a conceptual and standardised base in Cloud Computing. The ongoing publications of their activities create a solid base for cloud services development and offering:

NIST SP 800-145, NIST definition of cloud computing [1]
NIST SP 500-292, Cloud Computing Reference Architecture, v1.0 [2]
DRAFT NIST SP 800-146, Cloud Computing Synopsis and Recommendations [10]
NIST SP500-291 NIST Cloud Computing Standards Roadmap [11]
Draft SP 800-144 Guidelines on Security and Privacy in Public Cloud Computing [12]

1) NIST Cloud Computing Reference Architecture (CCRA)

NIST SP 800-145 document defines Cloud Computing in the following way:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics (on-demand self-service, broad network access, resource pooling. rapid elasticity, measured Service), 3 service/provisioning models. (Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS)), 4 deployment models (public, private, community, hybrid clouds)."

The IaaS service model is defined in the following way:

"The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of selecting networking components (e.g., host firewalls)."

Figure 1 presents a high level view of the NIST Cloud Computing Reference Architecture (CCRA), which identifies the major actors (Cloud Consumer, Cloud Service Provider, Cloud Auditor, Cloud Broker, and Cloud Carrier), their activities and functions in cloud computing. A cloud consumer may request cloud services from a cloud provider directly or via a cloud broker. A cloud auditor conducts independent audits and may contact the others to collect necessary information.

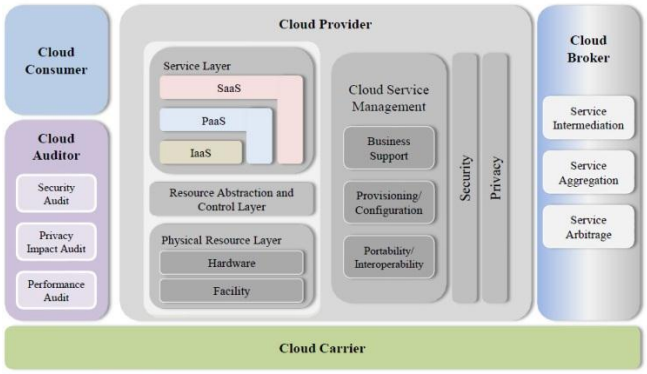


Figure 1. NIST Cloud Computing Reference Architecture (CCRA) [2]

The proposed architecture is suitable for many purposes where network performance is not critical but needs to be extended with explicit network services provisioning and management when the cloud applications are critical to network latency like in case of enterprise applications, business transactions, crisis management, etc.

2) Extending Cloud definition and CCRA for ICA

NIST CCRA and Cloud Computing definition are well suited for describing service, business, or operational relations. However, it has limited applicability for design purposes, i.e. defining basic functional components, interfaces, and layers.

The recently published CCRA includes the Cloud Carrier role, a role typical for telecom operators, which provides network connectivity as a 3rd party service. Despite the introduction of the Cloud Carrier role, there is no well-defined service model how network connectivity as a 3rd parte service be achieved. The IaaS cloud service model does not explicitly include provisioning of network services and infrastructure. One reason is that cloud computing has been developed primarily for provisioning storage and computing resources in the assumption that best-effort Internet connectivity is sufficient. However, this situation presents serious limitations for large scale use of cloud in enterprise applications that require guaranteed network connectivity QoS and low network latency in particular.

Another limitation of the current CCRA is that it is unsuitable for defining a security infrastructure and its integration with infrastructure services, which can be potentially multilayer and multi-domain.

The following extensions and improvements should be made to at least the Cloud IaaS model to meet requirements of a wide range of critical enterprise services (other service models such as PaaS, SaaS should also allow management of network related parameters):

- Define a layered cloud services model suitable for defining inter-layer and inter-service (functional) interfaces,
- Define virtualisation of resources and services as cloud features (in which virtualisation includes resource abstraction, pooling, composition, instantiation, orchestration, and lifecycle management),
- Include QoS provisioning and user / application control over QoS in the network services definition,
- Define an infrastructure service that includes the following attributes/features:
    - Topology description of computing, storage resources and their interconnection in the network infrastructure,
    - Infrastructure/topology description format that allows topology transformation operations for control and optimization (e.g., homomorphic, isomorphic, QoS, energy aware etc.).

In the context of the above definition, cloud infrastructure may include:

- Internal cloud provider infrastructure which is provided as a service, and
- External or Inter-Cloud infrastructure that can be provided by either a cloud operator or a network services provider.

In relation to business/operational aspects, the CCRA should be extended to address the following features:

- Better definition of the Cloud Carrier role, operational model and interaction with other key actors,
- Extend the set of basic roles with roles typical for telecom operators/providers as Cloud/infrastructure Operator, and split Customer role on Customer and User as representing customer organization and end-user.

B. IEEE Intercloud Working Group (IEEE P2302)

IEEE P2302 Working Group recently published a draft Standard on Intercloud Interoperability and Federation (SIIF) [13] proposing an architecture that defines topology, functions, and governance for cloud-to-cloud interoperability and federation.

Topological elements include clouds, roots, exchanges (which mediate governance between clouds), and gateways (which mediate data exchange between clouds). Functional elements include name spaces, presence, messaging, resource ontologies (including standardized units of measurement), and trust infrastructure. Governance elements include registration, geo-independence, trust anchor, and potentially compliance and audit.

However, the proposed approach has very limited scope by attempting to address a hypothetical scenario when all resources and applications will be located and run in multiple clouds and they need to be federated similar to Contend Distribution Network (CDN) [14]. The proposed architecture tries to replicate the CDN approach but doesn't address the generic problems with interoperability and integration of the heterogeneous multi-domain and multi-provider clouds.

The proposed solutions are built around extended use of the XMPP [15] as a base Intercloud protocol and introduce Intercloud Root and Exchange Hosts to support Intercloud communications, trust management and identity federation.

The proposed architecture originated from the position paper published by Cisco in 2009 [16] that tried to leverage the basic routing and messaging Internet protocols such as BGP, OSPF, XMPP to address Inter-Cloud integration and interoperability.

The limitation of the proposed architecture and approach is that it tries to closely imitate Internet approach in building hierarchical interconnected infrastructure for Internet protocol based services to support Inter-Cloud communication. But actually there is no need for such additional Inter-Cloud layer or infrastructure because cloud applications and infrastructure can use all Internet technologies directly to support intra-provider communications and user-customer-provider or inter-provider communications, given the appropriate network virtualisation and address translation technologies. Cloud technologies provide a virtualisation platform for IT and network services and allow entire infrastructure instantiation together with related protocols and core infrastructure services related to control and management functions. An

 ISBN: 978-1-61208-216-5

extreme use case that demonstrates the capabilities of cloud technologies is to create managed virtual Internets [27] using advanced programmable networking concepts [18].

III. GENERAL USE CASES FOR ICA

The two basic use cases for Inter-Cloud architecture can be considered: large project-oriented scientific infrastructure provisioning including dedicated transport network infrastructure, and periodic semester based educational course that requires computer laboratory facilities to setup, operated and suspended till the next semester [19]. Both cases should allow the whole infrastructure of computers, storage, network and other utilities to be provisioned on-demand, physical platform independent and allow integration with local persistent utilities and legacy services and applications.

Figures 2 illustrates the typical e-Science or enterprise infrastructure that includes enterprise proprietary and Cloud based computing and storage resources, instruments, control and monitoring system, visualization system, and users represented by user clients.

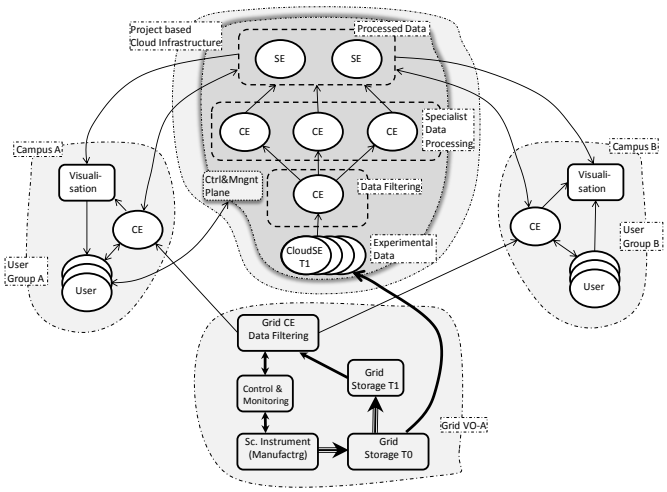


Figure 2. Project oriented collaborative infrastructure containing Grid based Scientific Instrument managed by Grid VO-A, 2 campuses A and B, and Cloud based infrastructure provisioned on-demand.

Figure 2 also illustrates a typical use case when two or more cooperative users/researcher groups in different locations want to use high performance infrastructure. In order to fulfill their task (e.g. cooperative image processing and analysis) they require a number of resources and services to process raw data on distributed Grid or Cloud data centers, analyse intermediate data on specialist applications and finally deliver the result data to the users/scientists. This use case includes all basic components of the typical e-Science research process: data collection, data mining, filtering, analysis (with special scientific applications), visualisation, and finally presentation to the users.

IV. ICA DEFINITION AND REQUIREMENTS

The developed Inter-Cloud Architecture should address the interoperability and integration issues in the current and emerging heterogeneous multi-domain and multi-provider clouds that could host modern and future critical enterprise infrastructures and applications.

The proposed ICA should address the following goals, challenges and requirements:

- ICA should support communication between cloud applications and services belonging to different service layers (vertical integration), between cloud domains and heterogeneous platforms (horizontal integration).
- ICA should provide a possibility that applications could control infrastructure and related supporting services at different service layers to achieve run-time optimization (Inter-Cloud control and management functions).
- ICA should support cloud services/infrastructures provisioning on-demand and their lifecycle management, including composition, deployment, operation, and monitoring, involving resources and services from multiple providers.

Following the above requirements, we define the subsequent complimentary components of the proposed Inter-Cloud Architecture:

(1) Multilayer Cloud Services Model (CSM) for vertical cloud services interaction, integration and compatibility;

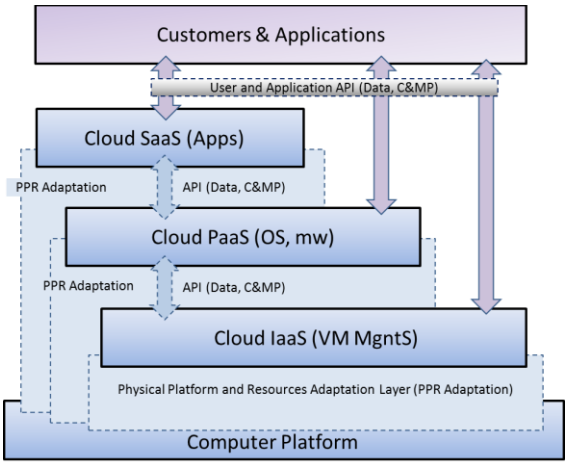
(2) Inter-Cloud Control and Management Plane (ICCMP) for Inter-Cloud applications/infrastructure control and management, including inter-applications signaling, synchronization and session management, configuration, monitoring, run time infrastructure optimization including VM migration, resources scaling, and jobs/objects routing;

(3) Inter-Cloud Federation Framework (ICFF) to allow independent clouds and related infrastructure components federation of independently managed cloud based infrastructure components belonging to different cloud providers and/or administrative domains; this should support federation at the level of services, business applications, semantics, and namespaces, assuming special gateway or federation services.

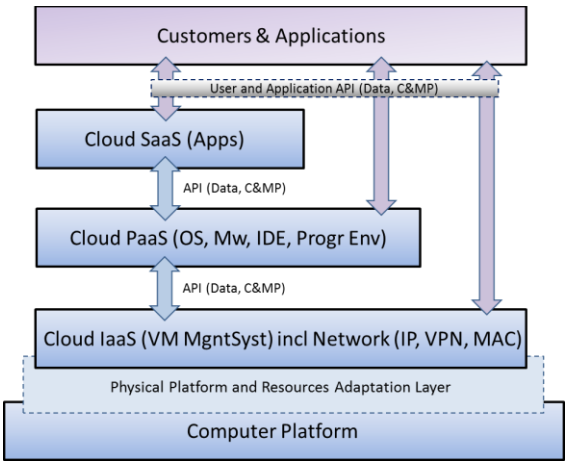
At this stage of research, we define only multi-layer Cloud Services Architecture that can be built using modern SOA technologies re-factored to support basic cloud service models as discussed below and in the following section. Future research on ICCMP will leverage User Programmable Virtualised Networks (UPVN) [20], and Internet technologies such as provided by CDN and Generalized Multi-Protocol Label Switching (GMPLS) [21]. The ICFF can be built using existing platforms for federated network access and federated identity management widely used for multi-domain and multi-provider infrastructure integration.

Figure 3 illustrates the current relation between basic Cloud service models IaaS, PaaS, SaaS that expose standards based interfaces to users, services, and applications but use proprietary interfaces to the physical provider platform. In case the application or service spans multiple heterogeneous cloud service providers, cloud services from different service models and layers will need to interact. This motivates definition of the Inter-Cloud Architecture that is depicted on Figure 3b as multilayer architecture with interlayer interfaces.

 ISBN: 978-1-61208-216-5



(a) Current relation between Cloud service models



(b) current relation between Cloud service models

Figure 3. Inter-Cloud Architecture for Cloud interoperability and integration.

In the proposed Inter-Cloud layered service model, the following layers can be defined (numbering from bottom up):
(6) Customers and applications,
(5) SaaS (or cloud applications) as a top cloud layer that represents cloud applications,
(4) PaaS provides middleware services to customers and applications (6) or used as a platform for (5),
(3) IaaS provides infrastructure services to (6) or used for hosting cloud platforms (4),
(2) Cloud virtualisation and management layer (e.g. represented by VMWare as virtualisation platform, and OpenNebula, OpenStack as cloud management software),
(1) Physical hardware (e.g. physical servers, network devices).

V. ABSTRACT MODEL FOR CLOUD BASED INFRASTRUCTURE SERVICES PROVISIONING

Figure 4 below illustrates the abstraction of the typical project or group-oriented Virtual Infrastructure (VI) provisioning process that includes both computing resources and supporting network that commonly referred as infrastructure services. The figure also shows the main actors involved into this process, such as Physical Infrastructure Provider (PIP), Virtual Infrastructure Provider (VIP), Virtual Infrastructure Operator (VIO).

The required supporting infrastructure services are pictured on the left side of the picture and includes functional components and services used to support normal operation of all mentioned actors. The Virtual Infrastructure Composition and Management (VICM) layer includes the Logical Abstraction Layer and the VI/VR Adaptation Layer facing correspondingly lower PIP and upper application layer. VICM related functionality is described below as related to the proposed Composable Services Architecture (CSA).

The proposed architecture is SOA based and uses the same basic operational principles as known and widely used by SOA frameworks. Consequently, the proposed architecture also provides a direct mapping to the possible VICM implementation platforms such as Enterprise Services Bus (ESB) [22] or OSGi framework [23].

The infrastructure provisioning process, also referred to as Service Delivery Framework (SDF), is adopted from the TeleManagement Forum SDF [24] with necessary extensions to allow dynamic services provisioning. It includes the following main stages: (1) infrastructure creation request sent to VIO or VIP that may include both required resources and network infrastructure to support distributed target user groups and/or consuming applications; (2) infrastructure planning and advance reservation; (3) infrastructure deployment including services synchronization and initiation; (4) operation stage, and (5) infrastructure decommissioning. The SDF combines in one provisioning workflow all processes that are run by different supporting systems and executed by different actors.

Physical Resources (PR), including IT resources and network, are provided by Physical Infrastructure Providers (PIP). In order to be included into VI composition and provisioning by the VIP they need to be abstracted to Logical Resource (LR) that will undergo a number of abstract transformations including possibly interactive negotiation with the PIP. The composed VI need to be deployed to the PIP which will create virtualised physical resources (VPR) that may be a part, a pool, or a combination of the resources provided by PIP.

The deployment process includes distribution of common VI context, configuration of VPR at PIP, advance reservation and scheduling, and virtualised infrastructure services synchronization and initiation, to make them available to Application layer consumers.

The proposed abstract model provides a basis for ICA definition and allows outsourcing the provisioned VI operation to the VI Operator (VIO) who is from the user/consumer point of view provides valuable services of the required resources consolidation - both IT and networks, and takes a burden of managing the provisioned services.

 ISBN: 978-1-61208-216-5

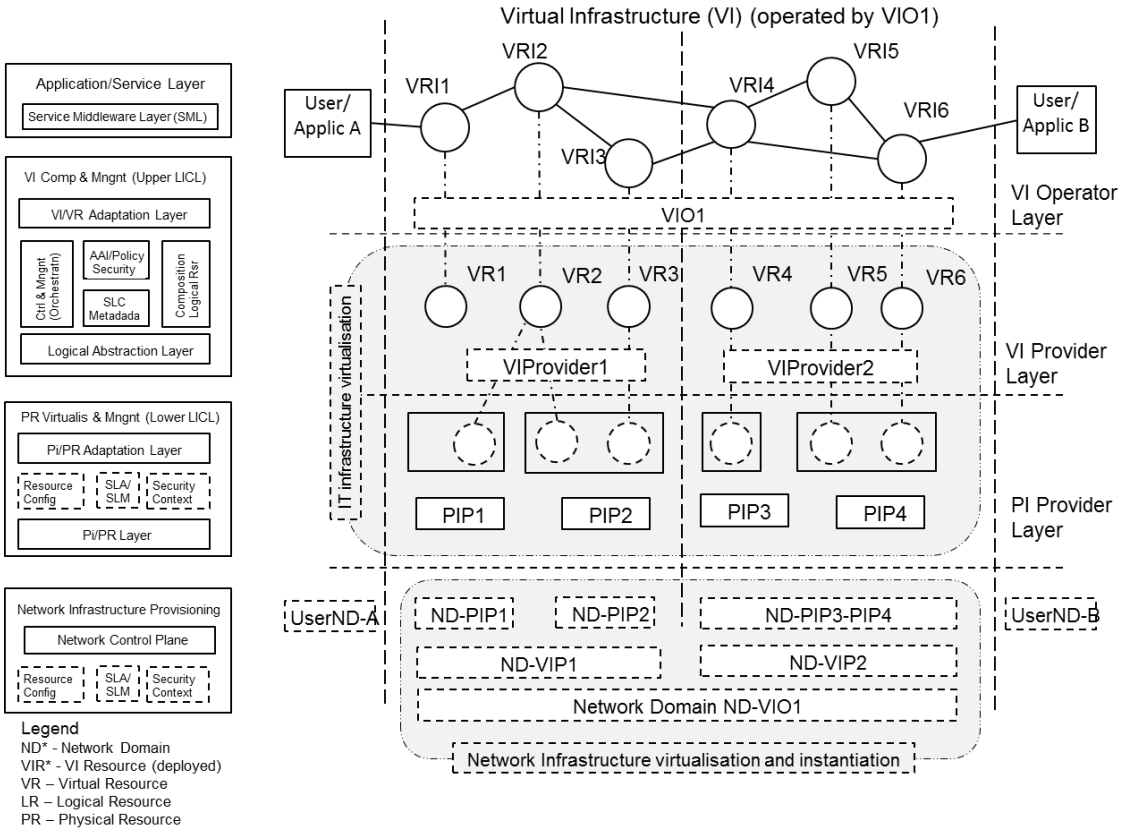


Figure 4. Main actors, functional layers and processes in on-demand infrastructure services provisioning

It is important to mention that physical and virtual resources discussed here are in fact complex software enabled systems with their own operating systems and security services. The VI provisioning process should support the smooth integration into the common federated VI security infrastructure by allowing the definition of a common access control policy. Access decisions made at the VI level should be trusted and validated at the PIP level. This can be achieved by creating dynamic security associations during the provisioning process.

VI. INFRASTRUCTURE SERVICES MODELING FRAMEWORK

The Infrastructure Services Modeling Framework (ISMF) provides a basis for virtualization and management of infrastructure services, including description, discovery, modeling, composition, and monitoring. In this paper we mainly focus on the description of resources and the lifecycle of these resources. The described model in this section is being developed in the GEYSERS project [5].

A. Resource Modeling

The two main descriptive elements of the ISMF are the infrastructure topology and descriptions of resources in that topology. Besides these main ingredients, the ISMF also allows for describing QoS attributes of resources, energy related attributes, and attributes needed for access control.

The main requirements for the ISMF are, that it should allow for describing Physical Resources (PR) as well as Virtual Resources (VR). Describing physical aspects of a resource means that a great level of detail in the description is required while describing a virtual resource may require a more abstract view. Furthermore, the ISMF should allow for manipulation of resource descriptions such as partitioning and aggregation. Resources on which manipulation takes place, and resources that are the outcome of manipulation are called Logical Resources (LR).

The ISMF is based on semantic web technology. This means that the description format will be based on the Web Ontology Language (OWL) [25]. This approach ensures the ISMF is extensible and allows for easy abstraction of resources by adding or omitting resource description elements. Furthermore, this approach has enabled us to re-use the Network Description Language [26] to describe infrastructure topologies.

B. Virtual Resource Lifecycle

Figure 5 illustrates relations between different resource presentations along the provisioning process that can also be defined as the Virtual Resource lifecycle.

The Physical Resource information is published by a PIP to the Registry service serving VICM and VIP. This published information describes a PR. The published LR information presented in the commonly adopted form (using

common data or semantic model) is then used by VICM/VIP composition service to create the requested infrastructure using a combination of (instantiated) Virtual Resources and interconnecting them with a network infrastructure. In its own turn the network can be composed of a few network segments run by different network providers.

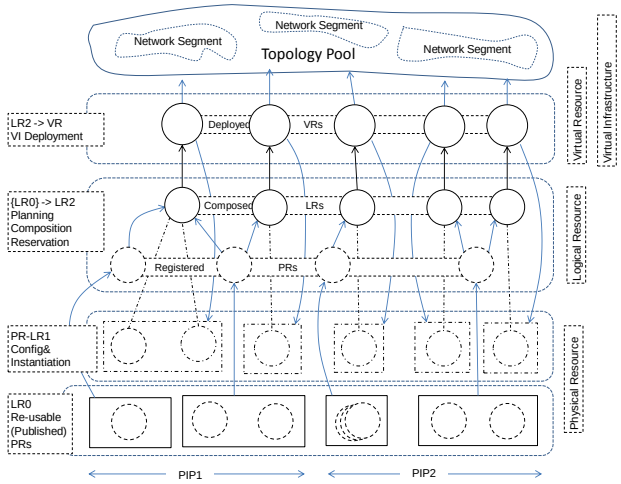


Figure 5. Relation between different resource presentations in relation to different provisioning stages.

VII. FUTURE DEVELOPMENT

The paper presents an on-going research at the University of Amsterdam to develop the Inter-Cloud Architecture (ICA) addresses the problem of multi-domain heterogeneous Cloud based applications integration and inter-provider and inter-platform interoperability.

The presented research is planned to be contributed to the Open Grid Forum Research Group on Infrastructure Services On-Demand provisioning (ISOD-RG) [27], where the authors play active role.

ACKNOWLEDGEMENTS

This work is supported by the FP7 EU funded Integrated project The Generalized Architecture for Dynamic Infrastructure Services (GEYSERS, FP7-ICT-248657) and by the Dutch national program COMMIT.

REFERENCES

[1] NIST SP 800-145, “A NIST definition of cloud computing”, [online] Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] NIST SP 500-292, Cloud Computing Reference Architecture, v1.0. [Online] http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf

[3] Generic Architecture for Cloud Infrastructure as a Service (IaaS) Provisioning Model, Release 1. SNE Techn. Report SNE-UVA-2011-03, 15 April 2011. [Online] http://staff.science.uva.nl/~demch/worksinprogress/sne2011-techreport-2011-03-clouds-iaas-architecture-release1.pdf

[4] GEANT Project. [Online] http://www.geant.net/pages/home.aspx

[5] Generalised Architecture for Dynamic Infrastructure Services (GEYSERS Project). [Online] http://www.geysers.eu/

[6] Demchenko, Y., R.Strijkers, C.Ngo, M.Cristea, M.Ghijsen, C. de Laat, Defining Inter-Cloud Architecture. Poster paper. Proc. 3rd IEEE Conf. on Cloud Computing Technologies and Science (CloudCom2011), 29 November - 1 December 2011, Athens, Greece. ISBN: 978-960-93-3482-2

[7] Open Virtualization Format (OVF), DMTF. [online] http://www.dmtf.org/standards/ovf

[8] Cloud Data Management Interface, SNIA. [online] http://www.snia.org/cdmi

[9] GFD.183 Open Cloud Computing Interface - Core [online] http://www.ogf.org/documents/GFD.183.pdf DRAFT NIST SP 800-146, Cloud Computing Synopsis and Recommendations. [online] Available: http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf

[10] NIST SP 800-146, Cloud Computing Synopsis and Recommendations. [online] Available: http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf

[11] NIST SP500-291 NIST Cloud Computing Standards Roadmap. [online] Available: http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/StandardsRoadmap/NIST_SP_500-291_Jul5A.pdf

[12] Draft SP 800-144 Guidelines on Security and Privacy in Public Cloud Computing. [online] Available: http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf

[13] IEEE P2302 - Standard for Intercloud Interoperability and Federation (SIIF). [online] http://standards.ieee.org/develop/project/2302.html

[14] Leung, K. and Lee, Y. (2011). Content Distribution Network Interconnection (CDNI) Requirements. IETF draft, work in progress, draft-ietf-cdni-requirement-00.

[15] RFC3920 Extensible Messaging and Presence Protocol (XMPP): Core. [online] http://www.ietf.org/rfc/rfc3920.txt

[16] Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., Morrow, M., Blueprint for the Intercloud - Protocols and Formats for Cloud Computing Interoperability. In Internet and Web Applications and Services, 2009. ICIW '09. Fourth International Conference on, 24-28 May 2009, Venice, Italy.

[17] J. D. Touch, Y.-S. Wang, L. Eggert, and G. G. Finn, “A virtual internet architecture,” *ISI Technical Report*, Mar. 2003.

[18] R. Strijkers, M. Cristea, C. de Laat, and R. Meijer, “Application framework for programmable network control,” Advances in Network-Embedded Management and Applications, pp. 37–52, 2011.

[19] Demchenko, Y., J. van der Ham, M. Ghijsen, M. Cristea, V. Yakovenko, C. de Laat, "On-Demand Provisioning of Cloud and Grid based Infrastructure Services for Collaborative Projects and Groups", The 2011 Intern. Conf. on Collaboration Technologies and Systems (CTS 2011), May 23-27, 2011, Philadelphia, Pennsylvania, USA

[20] Meijer, R. J., Strijkers, R. J., Gommans, L., and de Laat, C. (2006). User Programmable Virtualized Networks. In e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on (p. 43).

[21] RFC 3945. Generalized Multi-Protocol Label Switching (GMPLS) Architecture. [online] http://www.ietf.org/rfc/rfc3945.txt

[22] D. Chappell, ENTERPRISE SERVICE BUS, O’Reilly, June 2004.

[23] OSGi Service Platform Release 4, Version 4.2. [online] Available: http://www.osgi.org/Download/Release4V42

[24] TMF Service Delivery Framework. [Online] http://www.tmforum.org/servicedeliveryframework/4664/home.html

[25] OWL 2 Web Ontology Language [online] Available: http://www.w3.org/TR/owl2-overview/

[26] J. van der Ham, F.Dijkstra, P.Grosso, R. van der Pol, A.Toonk, C. de Laat, "A distributed topology information system for optical networks based on the semantic web", Elsevier Journal on Optical Switching and Networking, Volume 5, Issues 2-3, June 2008, pp. 85-93

[27] Open Grid Forum Research Group on Infrastructure Services On-Demand provisioning (ISOD-RG). [Online]. http://www.gridforum.org/gf/group_info/view.php?group=ISOD-RG

 ISBN: 978-1-61208-216-5

Cloud Network Security Monitoring and Response System

Murat Mukhtarov
Information Security Faculty
National Research Nuclear
University MEPhI
Moscow, Russia
Muhtarov.mr@gmail.com

Natalia Miloslavskaya
Information Security Faculty
National Research Nuclear
University MEPhI
Moscow, Russia
NGMiloslavskaya@mephi.ru

Alexander Tolstoy
Information Security Faculty
National Research Nuclear
University MEPhI
Moscow, Russia
AITolstoj@mephi.ru

***Abstract* — The public clouds network monitoring and response system, based on flow measurements, open source tools and CSMS (Cloud Security Monitoring System) module, is to be introduced in this paper. The main goal of the research is to develop an algorithm and to implement a system, which automatically detects and makes a response to network anomalies, occurring inside a Cloud infrastructure. In this research is proposed approach of anomaly detection inside the Cloud infrastructure which is based on a profiling method of IPFIX (IP Flow Information Export) protocol data and idea of negative selection principle is used for generating signatures of network anomalies, which are named detectors. The automatic response module makes a decision about network anomalies origin, based on several iterative checks and creates a record on the firewall rules table. The network traffic profiling process automatically generates the firewall rules set for all traffic classes, obtained during the learning process. Main results of the research are development of the algorithms and the way of the monitoring network attacks inside the Cloud. Implementation of the algorithms is python-based script and currently stays under hard-testing phase.**

Keywords - Cloud computing; Cloud infrastructure; Virtual Infrastructure; Application Hosting; Network Security.

I. INTRODUCTION

Cloud computing is a novel way to provide customers with Information Technology services, but with virtualization technologies in the background. Cloud computing uses networked infrastructure; software and computing power to provide resources to customers in an on-demand environment. With cloud computing, information is stored remotely in a centralized server farm and is accessed by the hardware or software thin clients that can include desktop computers, notebooks, handhelds and other devices. Typically, Clouds utilize a set of virtualized computers that enable users to start and stop servers or use compute cycles only when needed (also referred to as utility computing) [1]. In terms of information security, the cloud computing threat model consists of three fundamental issues: availability, integrity and confidentiality violations. Availability is terminated via Denial of Service -attacks. The likelihood and easiness of these attacks will increase as the volume of information exchanged between a user and a cloud provider increases. Integrity issues arise due to the fact that users must be sure that the information they retrieve is the same as that they store. This is a difficult task for one reason: information changes over time as do users themselves. But, also, it is important to separate users' information from the production information (for example configuration files, system files integrity, and so on). Finally, confidentiality issues may take place, for example over (accidental) disclosure of information to third parties or because of aggregation. Most computer compromises result in information leakage, so this is also an important issue [2].

In this research, we focus on availability as a main issue and the other issues that arise from it, so they are subsidiary risks for us. One possible way for Cloud networks to be monitored is to use a network telemetry principal with such protocols as Cisco Netflow [3] or IPFIX (Internet Protocol Information Export) [4]. Design of the open source virtualization technologies provides an opportunity to use Netflow/IPFIX probes on a hypervisor without performance reduction. IPFIX protocol has some advantages while being compared with the Netflow; it is not proprietary, it is open-standard and has improvements [5] that can be used in open source systems such as Linux or BSD (Berkley Source Distribution) -derivate systems. IPFIX is a lightweight network monitoring protocol for the connection control and volume-based traffic estimation [6]. Here we propose an approach to profile IPFIX data in such environment as a Cloud infrastructure and also suggest ways to make an automatic response to the detected anomalies inside a network. The way described in the paper is applicable to the Cloud solutions that provide their customers with such services as Infrastructure as a Service, Platform as a Service. In other words a Cloud Infrastructure consists of large amount of virtual machines running inside virtual infrastructure based on physical servers and network equipment.

II. STATE OF ART AND RELATED WORKS

The main focus of the paper is a network security monitoring approach in a Cloud infrastructure. We discuss some network security threats and issues that may occur in the virtual infrastructure clouds. All of them use shared hardware, network [1] and hypervisor's resources [2].

Security threats related to hosting application in a Cloud Infrastructure are covered by Molnar and Schechter [7]. The

 ISBN: 978-1-61208-216-5

researches compare traditional and cloud hosting focused on information security threats.

The Virtual Local Area Network (VLAN) separation technique on a Cloud Infrastructure is mentioned by Berger's et al. [8]. They suggest a way of increasing virtual infrastructure security by using a strong security policy inside a cloud infrastructure – Trusted virtual data center (TVDC). Their idea is based on the research of Bussani, et al., Trusted Virtual Domains (TVD): Secure Foundations for Business and Information Technology Services [9]. The main idea of TVDC is a strong isolation and integrity guarantee in virtualized, cloud computing environments [8]. To achieve this isolation researchers use network separation techniques based on IEEE 802.1q [10], memory control techniques and "colorizing" each data flow inside a cloud. Another approach to a Cloud infrastructure monitoring called "Private Clouds MONitoring Systems" (PCMONS) was created by Chaves, et al. [11]. Their main goal was to develop a modular and extensible monitoring system for the private Clouds. PCMONS is implemented as a module for the open source monitoring system Nagios and is compatible with the open source IaaS platform Eucalyptus [11]. But, it has several disadvantages: as PCMONS is a Nagios module, it inherited Nagios performance and scalability issues that eliminate applicability to the huge Cloud infrastructure; also it is compatible only with one solution. The described system monitoring approach is focused on network security monitoring and response actions inside a Cloud. The main advantages of the CSMS approach are compatibility with the majority of operating systems and network equipment due to IPFIX protocol, ability of an automatic response to a network attack and ability of identifying unknown network anomalies in some cases.

III. PROFILING NETWORK TRAFFIC DATA

To monitor the network traffic anomalies, that in fact are the result of DDoS-attacks or abuse traffic, we have to find a way that will be applicable to the implementation inside a network of a Cloud Infrastructure.

We worked out several requirements to this approach:

1) To be informative enough to analyze network traffic volumes by traffic types;

2) To be lightweight;

3) To be easy to spread through a Cloud infrastructure network and

4) To not impact production network performance.

The best way that satisfies all these requirements is to use flow-based measurement protocols like Netflow or IPFIX [6]. Here, we use IPFIX, because it is an open standard protocol.

To profile IPFIX data, we use a maximum entropy estimation approach, introduced in [12] and [13]. We have to modify and improve an algorithm of profile estimation to make it applicable to IPFIX data analysis (Fig. 1). For designing an algorithm we have to classify a given pattern of network traffic. Network traffic classification process is needed because traffic patterns usually consist of large amount of the different traffic packets and storing profile of raw traffic data will require large amount of disk space. Therefore, large volumes of data will require more processor time for processing. So, we propose to use preprocessing classification algorithm, which allows us to work with volume-based estimation of network traffic data, which is divided by classes. Result of preprocessing is a significant reduction of the size of data which should be processed by monitoring system. Amount of traffic classes should be selected by user. Also, an expert should exclude "anomalies" if they are present in a given pattern.

This algorithm checks in a cycle each traffic class with maximum entropy approach and estimates weights of each traffic class in a model. The algorithm's result is the network traffic profile in which only the most significant traffic classes in a given pattern are stored.

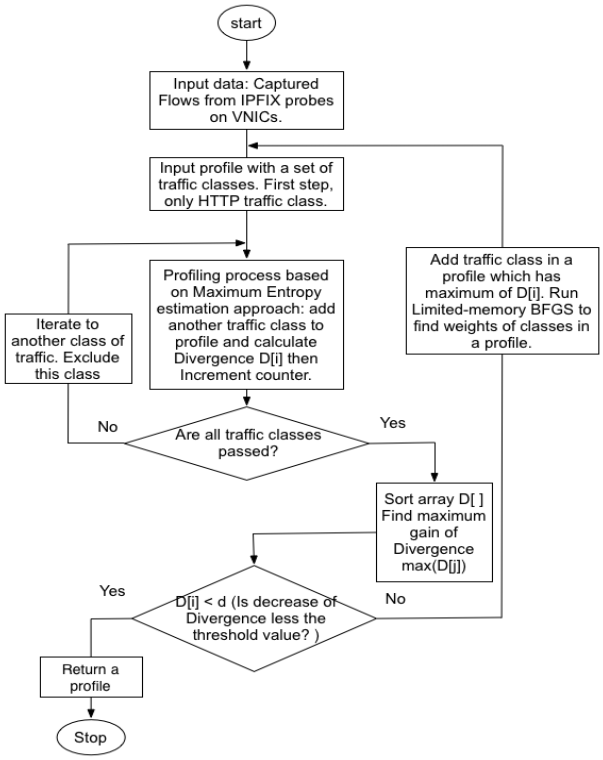


Figure 1. Block diagram of IPFIX data profiling.

The algorithm stops when the next traffic class does not improve the profile enough, in other words the decrease of divergence should be less then threshold value.

To adapt the algorithm to a Cloud Infrastructure network, we propose to make some special traffic classes that are inherent to the network of a Cloud infrastructure. We placed HTTP, HTTPS, DNS, SMTP, POP3, IMAP, POP3S, IMAPS, RDP, VNC, SQL ports in the separate classes. Also, we modified greedy algorithm to make it easier for implementation. We propose to exclude network traffic class from sampling process, after cycle pre-check with a

 ISBN: 978-1-61208-216-5

given pattern. This improvement allows avoiding additional checks of the network traffic patterns due to IPFIX protocol input data format.

IV. ANOMALY DETECTORS GENERATION ALGORITHM

Another approach proposed in this paper is a special way of generating a network anomaly detector. The idea of this approach lays in a negative selection algorithm, introduced by Forrest et al. [14]. According to the negative selection, a network traffic profile, which is returned by the IPFIX data-profiling algorithm (normal behavior profile), could be modified in the manner proposed below. To create a set of potential anomalies detectors, we increased the volumes of the traffic classes in a normal behavior profile with the random values in the range of Lower_Tr and Upper_Tr variables. Also there are several settings for the detector generating process: the amount of detectors needed, the amount of affected positions in a profile and a threshold value of divergence. The block diagram of the algorithm is shown on the Fig. 2.

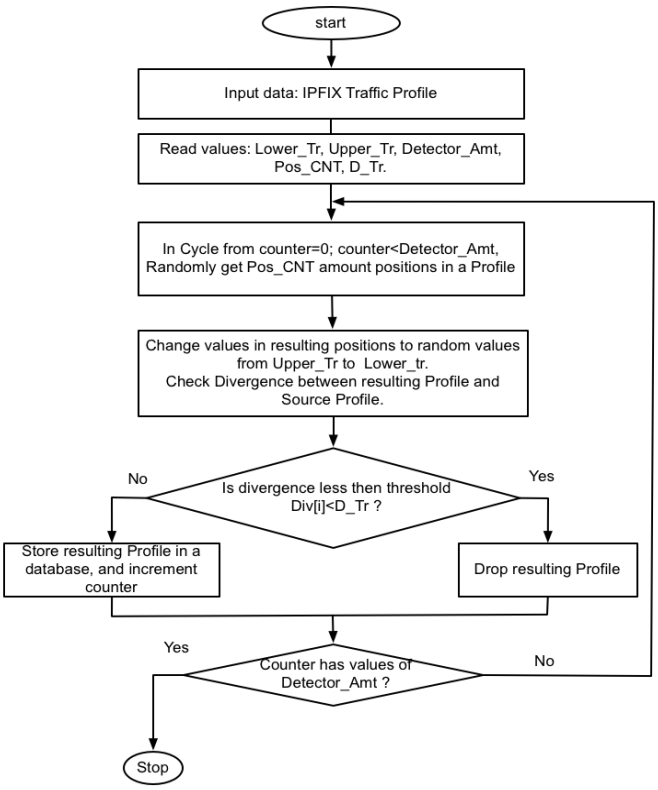


Figure 2. Anomaly detectors generation algorithm.

The algorithm randomly changes values of positions of the network traffic classes inside a profile according to the values of the Lower_Tr and Upper_Tr. The stopping criteria is an achievement of the required number of anomaly detectors. Detectors that are similar to the normal behavior profile should be dropped. All other detectors should be stored inside the database.

V. ANOMALY DETECTION AND RESPONSE INSIDE CLOUD INFRASTRUCTURE

Anomaly detection is based on the set of detectors, recorded in the database. Fig. 3 shows a detector life cycle, called "maturing" while comparing it with an immune system.

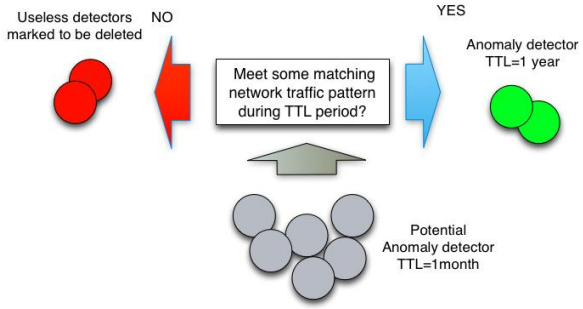


Figure 3. Anomaly detectors maturing process.

The probes are the exporting flows of IPFIX data to the collectors. A collector consists of two parts: the first one normalizes incoming data from the probes, and the second one performs a comparison of a captured traffic pattern with an anomaly detector. Each anomaly detector has "time-to-live" (TTL) attribute. Normally we set its values as one month. If a detector never matches any of the captured traffic patterns within one month it will be marked for deletion and it would be dropped at the end of the next month. One month period seems reasonable to reduce impact on the monitoring system performance and limit number of detector, but depends on user settings. Deletion of the detector does not mean that network anomaly which should be covered with deleted detector would not be handled properly. Generating of the anomaly detectors is a pseudorandom process which allows the possibility of the collisions. So such kind of anomaly possible could be found with detectors from another generation with some probability.

Another case is when a detector matched some of the network traffic pattern. This detector changes its TTL ("time-to-live") attribute to one year and spreads it across all probes. Hence, we could clean our detectors database from the patterns that we will never observe in the network traffic and collect patterns that are really useful for anomaly detection.

Fig. 4 introduces our algorithm of anomaly detection and response actions. IPFIX information has several attributes referred to the IP packet header data. When a network anomaly is detected, a Cloud monitoring system could tell us what kind of traffic causes an anomaly. In this case, we could find out a source IP address of anomaly traffic and block it inside a firewall.

 ISBN: 978-1-61208-216-5

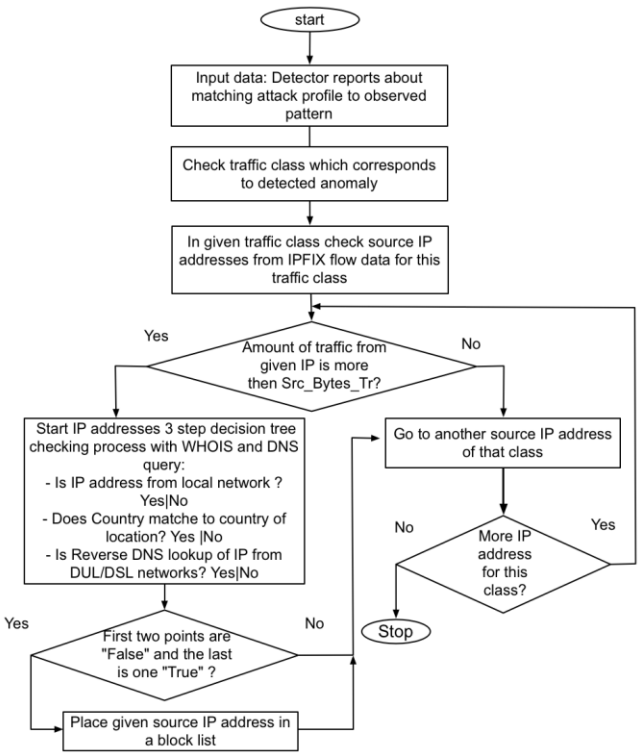


Figure 4. Block diagram of monitoring and response process.

To increase accuracy of the blocking system and to preserve a normal traffic we use "Whois" Database queries to learn an origin of IP address, location and reverse DNS queries to estimate the purpose of IP address usage. We perform several checks: "*Is the IP address from our network?", "Is the IP address from our Country?", "Is the IP address from Digital Subscriber Line (DSL)/Dial Up Line (DUL) network?".* So, if the IP address is outside of a Cloud network and region, we propose that it probably could be the reason for an anomaly and we block it for an hour. We perform the same action if the IP address is from DSL/DUL networks.

VI. INTEGRATING CSMS IN EXISTING CLOUD INFRASTRUCTURE

To show integration process of CSMS module and IPFIX sensors inside typical cloud datacenters, in this section we demonstrate deployment example. For example, we have already deployed cloud infrastructure based on open source private Cloud Eucalyptus as shown on Fig 5.

Eucalyptus Cloud Controller usually runs on a Linux-based computer with two network interface cards (NIC). Cloud Controller is a front end of the Cloud Infrastructure and it divides network on two parts: public local area network (on Fig 5. Public switch) and private local area network (on Fig 5. Private switch). We suggest to deploy CSMS module on Cloud Controller as it is central part of the Cloud Infrastructure and it is connected both private and public networks. Also, we suggest deploying firewall equipment, which is connected to the Public switch and able to block outside IP addresses in case of receiving command from CSMS.

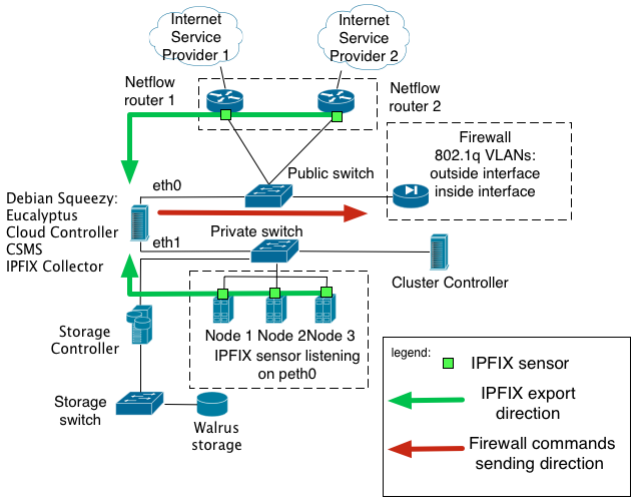


Figure 5. CSMS deployment inside Eucalyptus Cloud infrastructure.

IPFIX sensors deployed in the Cloud Infrastructure Nodes (component of the Cloud where runs Virtual Machines of the End User) and send information to the IPFIX collector, which is also deployed on the Cloud Controller. In addition, IPFIX data is exporting from the border routers. CSMS module analyzes incoming data from the several sources (Nodes and Routers) and performs anomaly recognition actions – compare anomaly detectors patterns against observing network traffic data. In the case when anomaly discovered, CSMS performing IP address check process to be sure that traffic not from own or trusted networks and then sends command to Firewall equipment in order to block malicious IP address. Advantage of this approach lies in possibility to deploy IPFIX sensors in every operating system which supports traffic capturing. It means that no matter which kind of Cloud or Virtualization technology going to be used, the most important is the ability to export IPFIX data from network equipment or from virtual network interfaces of the Cloud nodes.

VII. CONCLUSION AND FUTURE WORKS

Flow-based measurement protocols such as IPFIX are an appropriate source of network traffic information, which allows us to analyze traffic with statistical frameworks and approaches. In the paper we use the maximum entropy estimation approach to obtain the normal behavior network traffic profile based on IPFIX data. This way of monitoring network security is more productive and easy to implement in existing Clouds due to design and implementation of open source-based virtualizing software. The suggested approach of anomaly detection based on negative selection algorithm seems to be an appropriate way of monitoring in distributed environments such as a Cloud infrastructure network. It is ready to detect DDoS-attacks and other abuse

 ISBN: 978-1-61208-216-5

traffic attacks, having an availability issue for Cloud computing as a main concern. Automatic response ability of the CSMS with the "Whois" and reverse DNS information, based on source IP address filtering, is a useful way to preserve customers from false-positive errors.

The future developments of this research are testing and implementing of proposed algorithms and approaches to find a suitable way of integrating them inside the existing open source Cloud infrastructures. Also an applicability of the described proposal to the network attacks should be analyzed.

REFERENCES

[1] Securing the Cloud: A review of Cloud Computing, Security Implications and Best Practicies http://www.savvis.com/en-us/info_center/documents/savvis_vmw_whitepaper_0809.pdf. VMware Inc.(2009) (last access date 13/03/2012).

[2] Schoo P., Fusenig V., Souza V. at el. Chanlanges of Cloud Networking Security. 2nd International ICST Conference on Mobile Networks and Management, September 2010 Santandar Spain (October 2010). HP Laboratories, HPL-2010-137 (2010).

[3] Claise B. RFC 3954 Cisco Systems Netflow Services Export Version 9 (2004).

[4] Claise B. RFC 5101 Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information (2008).

[5] Boschi E. and Trammell B. Bidirectional Flow Measurement, IPFIX, and Security Analysis. pp. 8-10 (2006).

[6] Mukhtarov M., Miloslavskaya N., Tolstoy A. Netowrk security Threats and Cloud Services Monitoring. ICNS 2011 May 22, 2011 Venice/Mestre Italy. pp. 141-145 (2011).

[7] Molnar D., Schechter S. Self Hosting vs. Cloud Hosting: Accounting for the security impact of hosting in the cloud. WEIS 2010, pp.149-164 (2010)

[8] Berger S., Caceres R., Goldman K. and others Security for the cloud infrastructure: Trusted virtual data center implementation. IBM J. RES & DEV. Vol. 53, No. 5, paper 6, pp.1-12 (2009).

[9] Bussani A., Griffin J. L., Jasen B., Julisch K., Karjoth G., Maruyama H., Nakamura M., et al., ''Trusted Virtual Domains: Secure Foundations for Business and IT Services,'' Research Report RC23792, IBM Corporation (November 2005) (2005).

[10] IEEE Standard 802.1Q for Local and Metropolitan Area Networks—"Virtual Bridged Local Area Networks". IEEE 2005 see: http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf

[11] Chaves, S.A.; Uriarte, R.B. ; Westphall, C.B. "Toward an Architecture for Monitroing Private Clouds," IEEE Communications Magazine Vol. 49, Issue 12, pp. 130-137 (2011)

[12] Gu Y., McCallum A. and Towsley, D. Detecting anomalies in network traffic using maximum entropy. Tech. rep., Department of Computer Science, UMASS, Amherst, pp. 345-350 (2005).

[13] PietraS.D., Pietra V.D. and Lafferty J. Inducing features of random fields. IEEE Transactions on Pattern Analysis andMachine Intelligence 19, 4, pp. 380–393 (1997).

[14] Stephanie Forrest, Alan S. Perelson, L. Allen, and R. Cherukuri. Self -nonself discrimination in a computer. In Proceedings of the 1994 IEEESymposium on Research in Security and Privacy. IEEE Computer Society Press, pp. 360-365 (1994).

 ISBN: 978-1-61208-216-5

Analysis and Optimization of Massive Data Processing on High Performance Computing Architecture

He Huang, Shanshan Li, Xiaodong Yi, Feng Zhang, Xiangke Liao and Pan Dong
School of Computer Science
National Univ. of Defense Technology, Changsha, China
{*huanghe, shanshanli, xdyi, zhangfeng, xkliao, pdong*}*@nudt.edu.cn*

***Abstract*—MapReduce has emerged as a popular and easy-to-use programming model for numerous organizations to deal with massive data processing. Present works about improving MapReduce are mostly done under commercial clusters, while little work has been done under HPC architecture. With high capability computing node, networking and storage system, it might be promising to build massive data processing paradigm on HPCs. Instead of DFS storage systems, HPCs use dedicated storage subsystem. We first analyze the performance of MapReduce on dedicated storage subsystem. Results show that the performance of DFS scales better when the number of nodes increases; but, when the scale is fixed and the I/O capability is equal, the centralized storage subsystem can do a better job in processing large amount of data. Based on the analysis, two strategies for reducing the network transmitting data and distributing the storage I/O are presented, so as to solve the problem of limited data I/O capability of HPCs. The optimizations for storage localization and network levitation in HPC environment respectively improve the MapReduce performance by 32.5% and 16.9%.**

Keywords-high-performance computer; massive data processing; MapReduce paradigm.

I. INTRODUCTION

MapReduce [1] has emerged as a popular and easy-to-use programming model for numerous organizations to process explosive amounts of data and deal with data-intensive problems. Meanwhile, data-intensive applications, such as huge amount of web pages indexing and data mining in business intelligence nowadays have become very popular and are among the most important classes of applications. At the same time, High Performance Computers (HPCs) often deal with traditional computation-intensive problems. Though HPCs are very powerful when dealing with scientific computation problems, the architecture currently is not very suitable for running MapReduce paradigm and processing data-intensive problems.

There have been works done by improving MapReduce performance under HPC architecture. Yandong Wang et al. [3] improves Hadoop performance through optimizing its networking and several stages of MapReduce on HPC architecture. Wittawat et al. [4] integrates PVFS (Parallel Virtual File System) into Hadoop and compare its performance to HDFS and studies how HDFS-specific optimizations can be matched using PVFS and how consistency, durability, and persistence tradeoffs made by these file systems affect application performance. However, specific issues related to HPC architecture, especially the dedicated storage subsystem, are seldom taken into consideration in these former works. In the storage aspect these works are oriented to distributed file system (DFS) which uses local disks of each node to store data blocks, and it is not relevant to the centralized storage subsystem of the HPC architecture.

In this paper, alternatively, we consider every aspect of the HPC architecture, including processor, networking and especially the storage subsystem. In our work, the differences of DFS and centralized storage subsystem are analyzed in detail, and optimizations are proposed for the storage subsystem specifically in HPC environment. The prior concern of this paper is the deploying of MapReduce paradigm on HPCs and its overall performance. First of all, the difficulty and the significance of the Massive Data Processing problem on HPCs is described, and the necessity, feasibility, and problems that may be encountered of deploying MapReduce Paradigm on HPCs are analyzed. Secondly, the performance of MapReduce Paradigm on HPCs, especially the I/O capability of the dedicated storage subsystem and the DFS is analyzed and evaluated. Following that, two optimization strategies for relieving the I/O burden of the system and improving the performance of MapReduce on HPCs are presented, due to the limited data I/O capability of HPCs, which probably cannot meet the requirements of data-intensive applications.

Several challenges exist for deploying MapReduce paradigm and dealing with data-intensive problems effectively on HPCs. Firstly, data blocks are distributed and stored on DFS but centrally stored on the storage subsystem of HPCs. Therefore, how to decrease data transmission in advantage of the centralized storage in order to improve performance is a great challenge. Secondly, the IO and buffering capability of centralized storage is not as good as DFS. How to relieve the burden of storage I/O and improve the overall performance is another challenge.

This paper explores the possibility of building Massive Data Processing Paradigm on HPCs, and discusses how to deal with Massive Data Processing applications efficiently on HPCs and how to improve its performance. The main

 ISBN: 978-1-61208-216-5

contributions are:

(1) The performance of MapReduce Paradigm on HPCs, especially the I/O capability of the dedicated storage subsystem specific to HPCs is analyzed. Results show that the performance of DFS scales better when the number of nodes increases, but when the scale is fixed, the centralized storage subsystem can do better in processing large amount of data.

(2) Two strategies for improving the performance of the MapReduce paradigm on HPCs are presented, so as to solve the problem of limited data I/O capability of HPCs. The optimizations for storage localization and network levitation in HPC environment respectively improve the MapReduce performance by 32.5% and 16.9%.

The paper is organized as follows: related works of HPCs and data-intensive applications are discussed in Section II. In Section III, the specific issues of running MapReduce paradigm on HPCs are analyzed. Following that, in Section IV, two optimization strategies for improving the performance of the MapReduce Paradigm on HPCs are presented, in order to solve the problem of limited data I/O capability of HPCs, which probably cannot meet the requirements of data-intensive applications. Then, the effectiveness of these two optimization strategies is demonstrated respectively through experiments in Section V. Finally, we give a conclusion in Section VI.

II. Related Work

MapReduce is a programming model for large-scale arbitrary data processing. The model popularized by Google provides very simple but powerful interfaces, while hiding complex details of parallelizing computation, fault-tolerance, distributing data and load balancing. Its open-source implementation, Hadoop [2], provides a software framework for distributed processing of large datasets.

A rich set of research has been published on improving the performance of MapReduce recently. Originally, the Hadoop scheduler assumed that all nodes in a cluster were homogeneous and made progress with the same speed. Jiang et al. [5] conducted a comprehensive performance study of MapReduce (Hadoop), concluding that the total performance could be improved by a factor of 2.5 to 3.5 by carefully tuning the factors, including: I/O mode, indexing, data parsing, grouping schemes and block-level scheduling. Zaharia et al. [6] designed a new scheduling algorithm, Longest Approximate Time to End (LATE), for heterogeneous environments where ideal application environment might not be available.

Ananthanarayanan et al. [7] proposed the Mantri system which manages resources and schedules tasks on the MapReduce system of Microsoft. Mantri monitors tasks and culls outliers using cause- and resource-aware techniques and Mantri improves job completion times by 32%. Y. Chen et al. [8] proposed a strategy called Covering Set (CS) to improve the energy efficiency of Hadoop. It keeps only a small fraction of the nodes powered up during periods of low utilization, as long as all nodes in the Covering Set are running. The strategy should ensure that there is at least one copy of all data blocks in the Covering Set. On the other hand, Willis Lang et al. [9] proposed All-In Strategy (AIS). AIS uses all the nodes in the cluster to run a workload and then powers down the entire cluster. Both CS and AIS are efficient energy saving strategies.

The closest work to ours is Hadoop-A as proposed by Yandong Wang et al. [3] and Reconciling HDFS and PVFS by Wittawat et al. [4] The former paper improves Hadoop performance through optimizing its networking and several stages of MapReduce on HPC architecture. It introduces an acceleration framework that optimizes Hadoop and describes a novel network-levitated merge algorithm to merge data without repetition and disk access. Taking advantage of the InfiniBand network and RDMA protocol of HPCs, Hadoop-A doubles the data processing throughput of Hadoop, and reduces CPU utilization by more than 36%.

The second one, Reconciling HDFS and PVFS, explores the similarities and differences between PVFS, a parallel file system used in HPC at large scale, and HDFS, the primary storage system used in cloud computing with Hadoop. It integrates PVFS into Hadoop and compare its performance to HDFS using a set of data-intensive computing benchmarks. It also studies how HDFS-specific optimizations can be matched using PVFS and how consistency, durability, and persistence tradeoffs made by these file systems affect application performance.

Nonetheless, not every aspect of the HPC architecture is taken into consideration. For example, previous works claim that due to the price and the poor scalability of the centralized storage subsystem, disk arrays are not suitable for massive data processing. So in these works they simply use the DFS built upon local disks of each node. Consequently, in this paper a thorough study of dealing with massive data processing on the HPC architecture is given. The impact of every aspect on performance is checked, including processor, networking, and especially the storage subsystem.

III. Specific Issues of MapReduce on HPCs

This section mainly evaluates performance of MapReduce paradigm on HPCs through experiments and analyzes the problems of running MapReduce paradigm on both HPCs and clusters of commercial machines, especially the differences caused by the centralized storage subsystem and the DFS.

When running MapReduce paradigm on HPCs, the input and output data are stored in a dedicated storage subsystem (mainly composed of disk arrays and a parallel file system). Meanwhile, when running MapReduce paradigm on clusters of commercial machines, the Distributed File System (DFS) is responsible for managing the input and output data, and the data is actually stored on local disks of each node.

 ISBN: 978-1-61208-216-5

Table I
CLUSTER I/O PERFORMANCE UNDER DIFFERENT SIZES (MB/S)

#nodes	DFS		Storage Subsystem	
	Read	Write	Read	Write
4	2,960	408	13,300	1,024
7	4,690	630	15,500	1,010
10	6,400	860	19,000	1,020
20	12,120	1,680	29,044	1,022
40	23,760	3,280	31,100	1,200
80	44,000	6,400	31,000	1,190
100	62,200	8,080	31,093	1,160

Note that the read & write throughput of the cluster is evaluated by the Hadoop benchmark TestDFSIO. The throughput of 2,960, e.g., denotes that the cluster has a throughput of 2,960 MB/s for read.

Table II
MAPREDUCE PERFORMANCE UNDER DIFF. AMOUNT OF DATA

		60G	80G	100G	120G	140G
DFS	1	2'42	2'57	6'45	10'22	18'22
	2	3'16	6'06	7'55	12'17	15'34
	3	2'43	6'22	11'49	17'38	15'49
Disk Array	1	4'17	6'30	7'27	8'47	10'32
	2	4'49	6'51	8'08	9'52	11'40
	3	5'02	6'43	8'09	9'40	11'25

Note that the job finish time is evaluated by the Hadoop benchmark Sort with different amount of data. The finish time of 2'42, e.g., denotes that the job was finished in 2 min 42 sec. The size of both clusters is fixed with 10 nodes. Every job is evaluated for three times.

In order to analyze the performance of MapReduce paradigm on HPCs, it is needed to compare its performance with the performance of MapReduce paradigm on cluster of commercial machines under the same scale. So experiments in this section are divided into two groups: the first group store input and output data on dedicated storage subsystem of HPCs, representing the HPC computing environment. The second group uses the same scale of nodes, and differences are that the data is actually stored on local disks of each node and the DFS is responsible for managing data storage.

Data-intensive applications are different from traditional scientific computation applications. They need much more data accessing I/O bandwidth (i.e., disk accessing I/O bandwidth and network accessing I/O bandwidth) than computation resources. For I/O bandwidth is so important, the I/O capacity of the cluster should be evaluated first.

First of all, cluster I/O performance under different sizes is evaluated. Evaluation is done respectively in a commercial cluster and under the HPC environment. Nodes in these two clusters are the same, but during each assessment the number of nodes increases. The size of the DFS increases as the cluster scales, but the size of the centralized storage subsystem stays all the same: the dedicated storage system is composed of 167 600 GB fiber channel disks, and managed by Lustre [11] parallel file system. The I/O capacity of the cluster under different scales is listed in Table I.

From Table I, we can see that the I/O performance of DFS can improve linearly as the number of nodes increases. This is because as the number of nodes increases, the number of local disks in the DFS also increases. Under the management of the DFS, the I/O performance of the cluster can take advantage of all local disks to achieve aggregation I/O bandwidth, bringing linear performance improvement.

On the other hand, for dedicated storage subsystem, its I/O performance depends on the scale of disk arrays in the storage subsystem, and is almost not relevant with the number of computing nodes. As in the current experiment the size of the storage subsystem is fixed, the I/O bandwidth it can provide for all compute nodes is limited. Therefore, we get *Analysis 1: the scalability of DFS is better than that of the centralized storage system. The performance of DFS improves when the number of nodes increases, and the performance of centralized storage improves only when new disk arrays are added.*

Secondly, the performance of these two clusters under the same scale is evaluated. Table II describes MapReduce performance under different amount of data when the system is composed of 10 nodes and the I/O capability of DFS and centralized storage is nearly equal. From Table II, we can see that when the amount of data is small (less than 100 GB), the MapReduce performance of the DFS is better than the performance of the centralized storage subsystem. On the contrary, When the amount of data is large (more than 100 GB), the MapReduce performance based on centralized storage subsystem becomes much better. Besides the disadvantage in scalability of the centralized storage, this phenomenon reveals an advantage of the disk arrays and we get *Analysis 2: when the DFS and the centralized storage subsystem can provide equal I/O capability, the disk arrays of the centralized storage subsystem are better at dealing with huge amount of data, compared to disks of the DFS.*

In general, the experiments show that compared to the computation resources, the I/O accessing bandwidth is a valuable resource under HPC architecture and may has great impact on the performance of MapReduce. Firstly, the scalability of DFS is better than that of the centralized storage system. Meanwhile, the cost of enlarging the scale of centralized storage is much higher than that of DFS. Secondly, an advantage of the centralized storage is, the disk arrays are better at handling large amount of data, compared to the DFS. Based on the above analysis, optimizations for relieving the burden of I/O system and improving the overall performance are proposed in the next section.

 ISBN: 978-1-61208-216-5

IV. Optimizations of MapReduce Paradigm on HPCs

Based on the analysis of Section III, designs for improving the performance of the MapReduce Paradigm on HPCs are proposed, in order to solve the problem of limited data I/O capability of HPCs, which probably cannot meet the requirements of data-intensive applications. Therefore, two optimizations for relieving the burden of I/O system and improving the overall performance, Intermediate Results Network Transfer Optimization and Intermediate Results Localized Storage Optimization, are proposed in this section.

A. Intermediate Results Network Transfer Optimization

Data blocks are distributed and stored on DFS but centrally stored on the storage subsystem of HPCs. The main idea of Intermediate Results Network Transfer Optimization is to decrease data transmission in advantage of the centralized storage in order to reduce the time cost on networking and improve performance.

HPCs use dedicated storage subsystem and parallel file system to provide storage services for all computing nodes. DFS handle data blocks that is physically distributed on disks of each node, and logically organize these data blocks into a unified name space. On the other hand, when using dedicated storage subsystem, the difference is that data blocks are stored in disk arrays that are physically centralized. Therefore, the Map Output Files (MOFs) in the MapReduce paradigm are centrally stored in the storage subsystem, not distributed on every disk of each node, which brings possibility of optimizing network transmission of MOFs.

On clusters of commercial machines, after the Map phase of a MapReduce job finishes, the intermediate results (MOFs) are stored locally on the disk of the Map task execution node. At the Shuffle phase, all nodes that execute Reduce tasks must get the MOFs from the Map task execution node. So, these data blocks (MOFs) must be transmitted over the network between nodes. At this time, the MOFs are stored on distributed nodes, and their network transmission is inevitable.

Different from MapReduce jobs on clusters of commercial machines, these jobs on HPCs store intermediate results in the dedicated storage subsystem. Therefore, in this case, Map tasks just need to transmit the division and storage information of MOFs to all Reduce tasks, and then the Reduce tasks themselves are responsible for reading the corresponding intermediate results directly from the dedicated storage subsystem. Intermediate Results Network Transfer Optimization is illustrated in Figure 1.

From Figure 1, we can see that after Intermediate Results Network Transfer Optimization, Map tasks just transmit the division and storage information of MOFs to all Reduce tasks, and then the Reduce tasks themselves go for reading the corresponding intermediate results directly from the dedicated storage subsystem. This eliminates the process of transmitting the intermediate results to Reduce tasks over network and can relieve the networking I/O burden of the system. If the network is the performance bottleneck of the system, this optimization can improve the overall system performance.

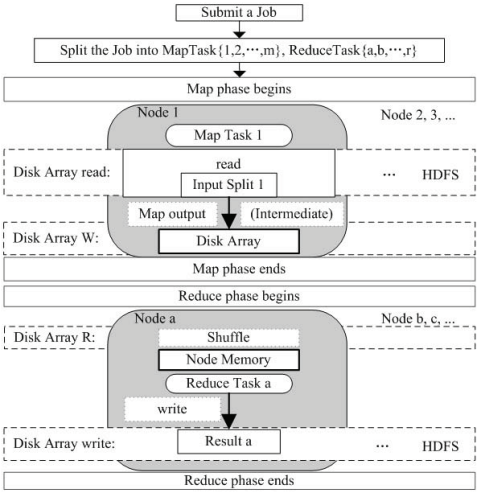


Figure 1. Intermediate Results Network Transfer Optimization

B. Intermediate Results Localized Storage Optimization

Compared to the networking I/O resources, the storage I/O resources provided by the centralized storage system are more likely to become the performance bottleneck of the system. The main idea of Intermediate Results Localized Storage Optimization is to distribute the storage I/O to both the centralized storage and local disks of each computing node. By storing temporary data files on local disks of computing nodes, the I/O pressure of centralized storage can be reduced greatly.

The I/O capability which the dedicated storage subsystem can provide is limited by the size of the storage subsystem itself. On the other side, when the MapReduce paradigm is initially designed, the intermediate results (MOFs) are not written to DFS, but stored temporarily on the local disk of each node. We can learn from this design, and store the intermediate results temporarily on the local disk of each node to reduce data I/O pressure of the centralized storage system.

Furthermore, as the dedicated storage subsystem of HPCs is expensive and limited in scale, the capacity and I/O capability of the storage subsystem often become a kind of scarcer resources, rather than network I/O bandwidth or computation resources. Then it is more urgent to relieve the I/O pressure of the storage system, rather than to optimize

 ISBN: 978-1-61208-216-5

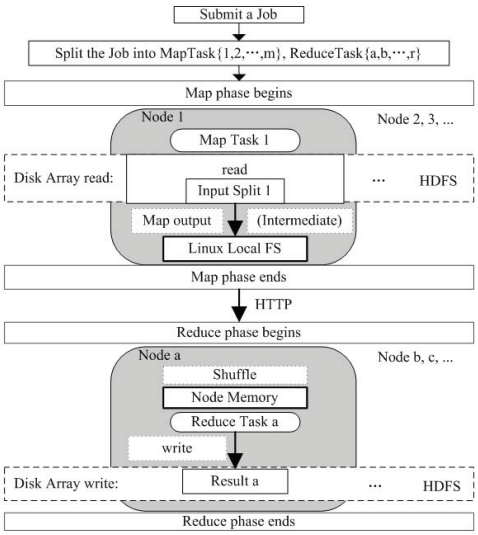


Figure 2. Intermediate Results Localized Storage Optimization

the data transmission over the network to improve the performance of MapReduce paradigm on HPCs.

In view of this, we can learn from the practice of distributed file systems, and buffer the intermediate results locally. As MOFs are temporary files and belong to specific jobs, and are usually deleted after the completion of their corresponding job, buffering the MOFs locally does not affect the correct execution of MapReduce jobs. At the same time, buffering intermediate results locally can relieve the burden of the centralized storage greatly. Intermediate Results Localized Storage Optimization is illustrated in Figure 2.

From Figure 2, we can see that the job input and output data are read and written to the centralized storage, but the intermediate results are no longer written to the centralized storage. Instead, they are buffered locally on disks of each computing node. Therefore, the I/O of the whole system is distributed and the burden of centralized storage is relieved greatly.

V. EVALUATION

Experiments are done respectively in a commercial cluster and the HPC environment. The commercial cluster and HPC environment both have 100 compute nodes, each node has dual-way six-core 2.93 GHz Intel Xeon processors and 50GB memory. In the commercial cluster nodes are connected by 1 GB Ethernet. In HPC environment nodes are connected by 40 Gbps optical fiber channel and InfiniBand [10] network. The dedicated storage system is composed of 167 600 GB fiber channel disks, and managed by Lustre [11] parallel file system.

Three groups of evaluation are done in this section. First of all, the scalability of the DFS and the centralized storage

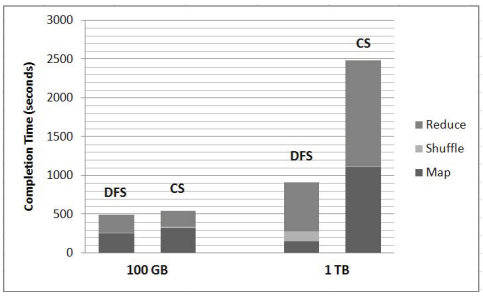


Figure 3. The Performance Scalability of DFS and CS

is evaluated and compared. Then, the effectiveness of the Intermediate Results Network Transfer Optimization and the Intermediate Results Localized Storage Optimization is demonstrated respectively, when the networking or storage I/O becomes the performance bottleneck of the overall system.

Firstly, the Terasort benchmark of Hadoop is run to evaluate the performance scalability of DFS and centralized storage. 100 GB data is sorted on 10 nodes and 1TB data is sorted on 100 nodes respectively. And the networking of both groups is 40 Gbps InfiniBand. The results are illustrated in Figure 3. In Figure 3, DFS represents the distributed file system and CS represents the centralized storage. The total time cost is composed of the time cost of three MapReduce phases: Map, Shuffle and Reduce.

From Figure 3, we can see that, when 100 GB data is sorted on 10 nodes, the performance of DFS and CS is nearly equal. But when the system scales, that is, when 1 TB data is sorted by 100 nodes, the performance of CS is worse than that of DFS. As the computation hardware and networking are the same, the differences come from distinctive storage system. This validates our analysis in Section 3: the scalability of DFS is better than that of the centralized storage system. In fact, the cost of enlarging the scale of centralized storage is much higher than that of DFS, as disk arrays are much more expensive than simple disks attached to computing nodes.

Secondly, the effectiveness of Intermediate Results Localized Storage Optimization is demonstrated. The same from above, 1 TB data is sorted on 100 nodes with centralized storage, for the first time without optimization and the second time with storage localization optimization. The networking of both tests is 40 Gbps InfiniBand. From the former experiments we can see that the performance bottleneck of the overall system lies on the storage I/O. After Intermediate Results Localized Storage Optimization, the MOFs are not written to the centralized storage system anymore, and it greatly relieves the pressure of the disk arrays of the storage subsystem. The results are illustrated in Figure 4.

From Figure 4, we can see that after storage localization

 ISBN: 978-1-61208-216-5

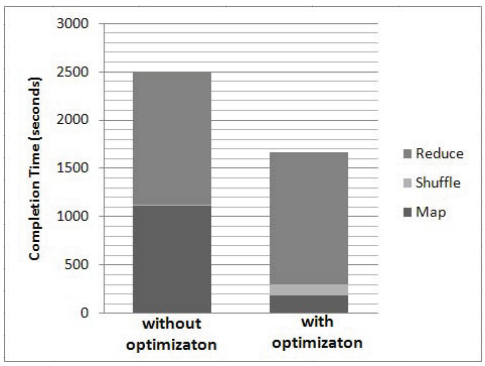


Figure 4. Validation of Intermediate Results Localized Storage Optimization

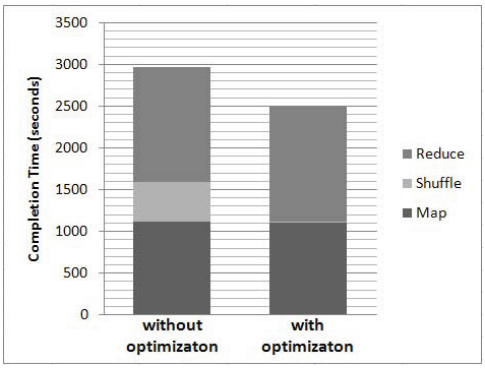


Figure 5. Validation of Intermediate Results Network Transfer Optimization

optimization, the time cost decreases greatly, especially the time cost of Map phase. Because during Map phase, it is not needed any more to write intermediate results to the centralized storage, the performance of Map phase improves a lot. In fact, the Intermediate Results Localized Storage Optimization in HPC environment can improve the MapReduce performance by 32.5%.

Thirdly, the effectiveness of Intermediate Results Network Transfer Optimization is demonstrated. The same from above, 1 TB data is sorted on 100 nodes with centralized storage, for the first time without optimization and the second time with network levitation optimization. But the networking changes to 1 GB/s Ethernet this time, in order to see the networking as performance bottleneck. The results are illustrated in Figure 5.

Different from the former experiments, we can see from Figure 5 that the performance bottleneck of the overall system this time lies on both the networking and the storage I/O. Note that if the 40 Gbps InfiniBand is used this time, the networking would not be the bottleneck and the Intermediate Results Network Transfer Optimization would become useless.

Figure 5 shows that when the networking capability turns into the bottleneck of the system, the Intermediate Results Network Transfer Optimization in HPC environment can improve the MapReduce performance by 16.9%. In fact, only the Shuffle phase consumes the networking resources, and the Intermediate Results Network Transfer Optimization improves the performance of the Shuffle phase a lot. Most data transmitted over network is MOFs, and after the networking levitation optimization most network flow of the Shuffle phase is eliminated.

VI. Conclusion

This paper aimed at exploring the possibility of building Massive Data Processing Paradigm on HPCs. The performance of MapReduce Paradigm on HPCs, especially the I/O capability of the dedicated storage subsystem specific to HPCs is analyzed. Two optimizations for storage localization and network levitation in HPC environment respectively improve the MapReduce performance by 32.5% and 16.9%. The conclusion is that when the corresponding I/O capability is the performance bottleneck of the overall system, these optimizations can help improve MapReduce paradigm under HPC architecture.

Acknowledgment

The authors are thankful to the anonymous reviewers. This work was supported by NSF China grant 61133005.

References

[1] J. Dean, and S. Ghemawat, “Mapreduce: Simplified Data Processing on Large Clusters,” In Proc. of OSDI, 2004, pp. 137-150.

[2] Hadoop, http://hadoop.apache.org/, retrieved: May, 2012.

[3] Y. Wang, X. Que, W. Yu, D. Goldenberg, and D. Sehgal, “Hadoop Acceleration through Network Levitated Merging,” In Proc. of Super Computing, 2011, pp. 57-66.

[4] W. Tantisiriroj, S. Son, S. Patil, S. Lang, G. Gibson, and R. Ross, “On the Duality of Data-Intensive File System Design: Reconciling HDFS and PVFS,” In Proc. of Super Computing, 2011, pp. 67-78.

[5] D. Jiang, B. Ooi, L. Shi, and S. Wu, “The Performance of Mapreduce: An In-Depth Study,” In Proc. of VLDB, 2010, pp. 472-483.

[6] M. Zaharia, A. Konwinski, A. Joseph, R. Katz, and I. Stoica, “Improving Mapreduce Performance in Heterogeneous Environments,” In Proc. of OSDI, 2008, pp. 29-42.

[7] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, and B. Saha, “Reining in the Outliers in Map-Reduce Clusters using Mantri,” In Proc. of OSDI, 2010, pp. 265-278.

[8] Y. Chen, L. Keys, and R. Katz, “Towards Energy Efficient Hadoop,” In UC Berkeley Technical Report, number UCB/EECS-2009-109, 2009.

[9] W. Lang, and J. Patel, “Energy Management for MapReduce Clusters,” In Proc. of VLDB, 2010, pp. 129-139.

[10] IBTA, “InfiniBand Architecture Specification, Release 1.0,” http://www.infinibandta.org/specs/, retrieved: May, 2012.

[11] Lustre Parallel Filesystem, “The Lustre Storage Architecture,” http://www.lustre.org/, retrieved: May, 2012.

 ISBN: 978-1-61208-216-5

Providing a Solution for Live Migration of Virtual Machines in Eucalyptus Cloud Computing Infrastructure without Using a Shared Disk

Shayan Zamani Rad
Mazandaran University of Science and Technology
Computer Engineering and IT Department
Babol, Iran
sh.zamani@ustmb.ac.ir

Morteza Sargolzai Javan
Amirkabir University of Technology
Computer Engineering and IT Department
Tehran, Iran
msjavan@aut.ac.ir

Mohammad Kazem Akbari
Amirkabir University of Technology
Computer Engineering and IT Department
Tehran, Iran
akbarif@aut.ac.ir

Abstract— **Today, cloud computing is used as a model in most scientific, commercial, military, other fields. In this model, the main body of the system are virtual servers, which currently provide services to customers around the world. In these circumstances, since the servers are virtual, they can be transferred as a file from one machine to another, which is known as migration. Migration practice is done for a variety of purposes, including load balancing, fault tolerance, power management, reducing response time, increasing quality of service, and server maintenance. Because the use of this technique is highly dependent on cloud computing infrastructure architecture, in some cloud infrastructures, such as Eucalyptus, the virtual machine migration technique has not been used yet. In this paper, we propose a solution for VM migration technique on Eucalyptus Cloud environment. The experiments show the validity of the proposed solution in non-shared disk Cloud environments, where the total migration time and transferred data have been significantly increased.**

Keywords-Eucalyptus; Cloud computing infrastructure; Virtual Machine; Migration.

I. INTRODUCTION

Cloud computing has been considered as a new way of providing Information Technology services to individuals and organizations. In addition, given the increase tendency of users and companies, academic and research centers strive to provide solutions and new tools in the Cloud computing. While commercial products are offered with the goal of cost reduction and customer satisfaction, productivity tools in academic centers are to discover new solutions based on open source technologies. Eucalyptus is an open-source Cloud-computing framework that uses computational and storage infrastructure which is commonly available to academic research groups to provide a platform that is modular and open to experimental instrumentation and study [7]. One of the weaknesses in this Cloud framework is the lack of virtual machine (VM) migration technique. Given that migration technique is done for different purposes, such as load balancing, fault tolerance, power management, reducing response time and increasing quality of service, server maintenance, etc. Therefore, there is not any mentioned algorithm in Eucalyptus. In this paper, we implemented migration technique by presenting solution in the Eucalyptus and establish a basis for providing other security and management algorithms.

The rest this paper is organized as follow: In Section 2, we will describe different migration methods. In Section 3, we examine the architecture of Eucalyptus with its components. In Section 4, we will describe the proposed method. Finally, in Section 5, we evaluate our methods.

II. RELATED WORKS

Currently, there are some works on the migration which can be referred to Pre-copy method in [2] [3] [4]. This method has three steps for migration a Virtual Machine: in the first step, the virtual machine memory pages are transferred in several rounds and then, in step two, the virtual machine CPU states are sent to the destination. After that in step three, the memory pages in source and destination are synched with each other.

Also, Hines and Gopalan [5] present a Post-copy technique with optimization methods. In [6], the method of CR/TR has been presented, which aims to send Logs file instead memory pages toward the destination.

Considering the benefits of Pre-copy approach, this is the main migration method, which is supported in most Hypervisors such as XEN and KVM. Thus, we used Pre-copy for sending memory pages and CPU states of virtual machines. However, Pre-copy approach has some shortcomings; one of these shortcomings is the lack of disk migration (transfer) algorithm. Hence, we cannot use default Pre-copy method in the non-shared disk environments, because the virtual machine disk must be transferred.

In this condition, our method has a disk transmission algorithm which is not dependent on the shared disk and can be used in the above-mentioned environments. In addition,

 ISBN: 978-1-61208-216-5

use of Pre-copy and other migration methods in Eucalyptus is impossible, because it has certain challenges; therefore, in our method, these challenges have been solved and this is an important difference between our method and that of others.

III. EUCALYPTUS

Eucalyptus is an open source implementation of Cloud computing infrastructure that has a particular architecture. By using Eucalyptus, we can make public and private Clouds. The architecture of the Eucalyptus system is simple, flexible and modular with a hierarchical design, reflecting common resource environments found in many academic settings. In essence, the system allows users to start, control, access, and terminate entire virtual machines using an emulation of Amazon EC2's SOAP and "Query" interfaces [7]. That is, users of Eucalyptus interact with the system using exactly the same tools and interfaces that they use to interact with Amazon EC2 [8].

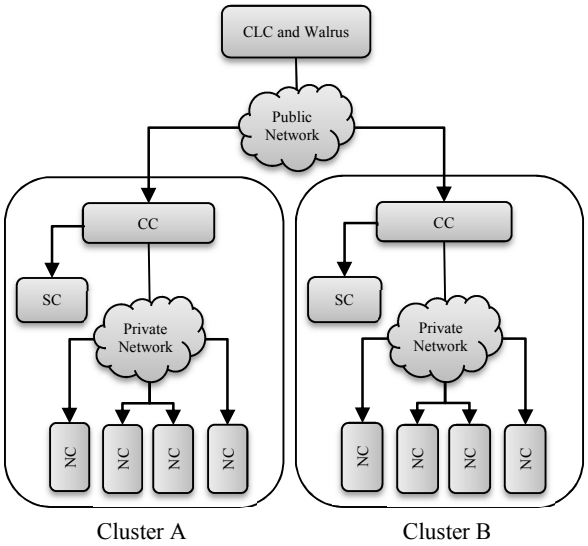


Figure 1. Eucalyptus Architecture [7]

The Eucalyptus is composed of five main components [7]:

- Cloud Controller (CLC): This component is frontend of the infrastructure and through a Web interface interacts with users and provides possibility of controlling virtual machine.
- Walrus: It is a put/get storage service that implements Amazon's S3 interface, providing a mechanism for storing and accessing virtual machine images and user data.
- Cluster Controller (CC): This component is responsible for the management of one or more node controllers. It also manages and sends the order of running of the instances on them. In Eucalyptus, Virtual Machine is known as Instance.
- Storage Controller (SC): This component provides virtual disks for instances, allowing them to permanently store and keep the information. This is very similar to the EBS service.
- Node Controller (NC): It controls the execution, inspection, and terminating of VM instances on the host where it runs.

The Eucalyptus architecture and its components are shown in Figure 1.

IV. CHALLENGES

There are some challenges in implementing the migration technique in Eucalyptus Cloud Infrastructure. These challenges do not allow to implementing ordinary VM migration methods; in fact, these challenges are the properties of Eucalyptus.

A. Clearing the Instance Data after Turning it off

In Eucalyptus, when the instance is turned off even temporarily, its information would be completely removed from CC, NC and CLC. However, the instance would enter in suspend mode for a short time (60 milliseconds) in all migration methods.

B. Operations Management is Performed by CLC

All operations and activities must be performed under the CLC and the CC, and if any actions get implemented without these two components, the structure of Eucalyptus will change. But, in all migration methods, migration operations are done under the hypervisor. Now, the NC and the hypervisor are executive components in Eucalyptus, and all operations will report to the higher administrative units.

C. Lack of Shared Disk

In most migration methods, the disk of virtual machine is considered as a shared disk that is the source and destination hypervisors have access to it. Therefore, during the migration process, only memory pages and CPU states are displacing, but if there is no shared disk available in Eucalyptus, the disk must be transferred when moving a VM.

D. Some Common Mistakes

In Eucalyptus, when instances are displaced, some information must be updated and changed, e.g., available resources, the number of being established instances, the number of running instances, etc. If any of this information has incorrect content, Eucalyptus performance and overall cloud would decrease and its structure might be out of control.

E. Instance Death Zone Time

In Eucalyptus, if the running instance cannot send any response (heart beat) to NC in ranged 20 to 24 seconds for any reasons, CLC will assume that instance is terminated. Then, CLC release it from the list of running instances and delete all information about it. Consequently, API functions will not be used for this instance, because the CLC would not know an instance with this name. Therefore, migration operation should be less than the mentioned time period which we have named it the "Instance Dead Zone Time".

 ISBN: 978-1-61208-216-5

V. SYSTEM ARCHITECTURE

In Eucalyptus, all commands are issued by the CLC, and the CC and NC behave as an observer and a worker (commands runner), respectively. Furthermore, the user (client) and cloud administrator input their requests to the CLC through running API functions. Therefore, in order to create the migration capability in Eucalyptus, we first created an API function that is responsible for the migration. This function has the following format:

euca-migrate-instance -i instance_id –d destination_node

Through running *euca-migrate-instance* API, first, the CLC finds node's IP address that the instance is currently running there. Then, these three values are transferred to the related cluster controller (where the cluster is running the instance). Figure 2 shows the process in the CLC.

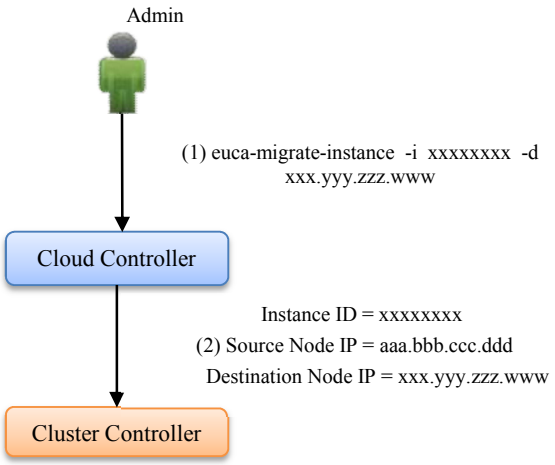


Figure 2. CLC with defined API Function

After CC's stub receives commands, it checks the needed resource amounts and makes decisions about sending migration command to the appropriate nodes.

After sending the migration command from CC to source node (NC), created stub in the NC receives related information (Figure 3). Next, it tries to communicate with the destination node and its hypervisor. After establishing relations, the source hypervisor would send memory pages and CPU states with uses Pre-copy method (Figure 4). After a few seconds, the migration process ends. Now, the instance is running on the destination node. Finally, the instance's information must be updated on the source and destination NCs, the CC and the CLC.

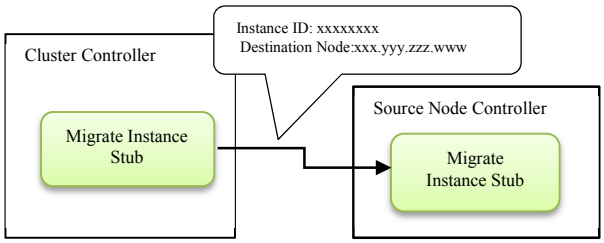


Figure 3. Relationship of CC and NC through stubs

At this point, the destination NC updates instance's information, which is running on it. Afterward, source NC removes instance's name and information from the own list of running instances. Next, the CC updates its information and changes instance location. This change causes the following executive orders to be sent to the destination NC. These orders include terminate, reboot, attach and detach volumes to instance.

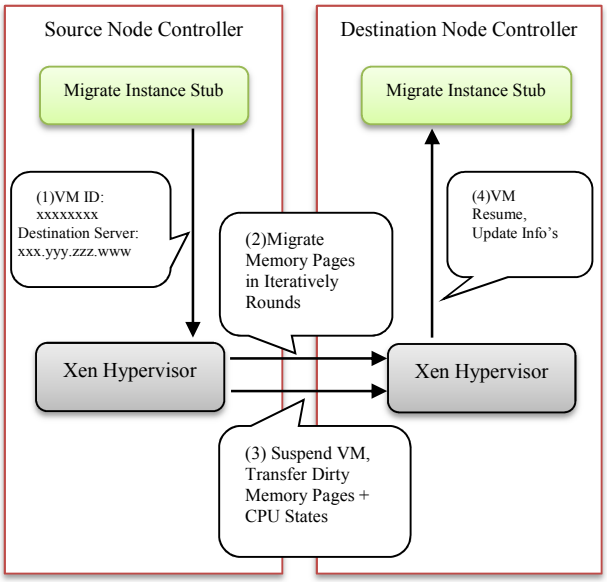


Figure 4. Relationship between Source and Destination NCs for Migration Operation

A. Disk Transfer Algorithm

In Eucalyptus, if there is no available shared disk, when the instance is migrated, its disk must be transferred. So, if the Xen hypervisor [9] is used without any changes for instance migration, instance moves with memory in destination node and its disk is located in source node. As a result, this is a fault in migration process. Therefore, it must be used an algorithm to transfer instance's disk. This algorithm must be written within the hypervisor. At the beginning of migration process, exactly before transferring memory pages and CPU states, the instance's disk must be transferred to the destination node. In fact, transmission of disk blocks lasts usually much longer than the transferring memory pages and CPU states. Thus, the disk transfer must be done before memory pages and CPU states transfer

 ISBN: 978-1-61208-216-5

begin. In the following, we illustrate our proposed disk transferring algorithm:

```
1. for i=min to max (number of disk blocks)
2.   Begin to Transfer Blocks
3.   if I/O Request is coming and  I/O = WRITE then{
4.      block-bitmap[i] = 1}
5. While bock-bitmap[i]=1 {
6.   Transfer Dirty Block_i to Destination}
7. End
```

Pseudo-code of Disk Transfer Algorithm

Also, we have used LZO [10] compression algorithm in our disk transfer algorithm presented. The disk blocks in the source node are compressed before transmission. And then, they are sent to the destination node to be decompressed. This action causes optimal use of available bandwidth and also the data will be transferred in much less time than it is usually the case (without compression). Figure 5 shows the steps of our migration method.

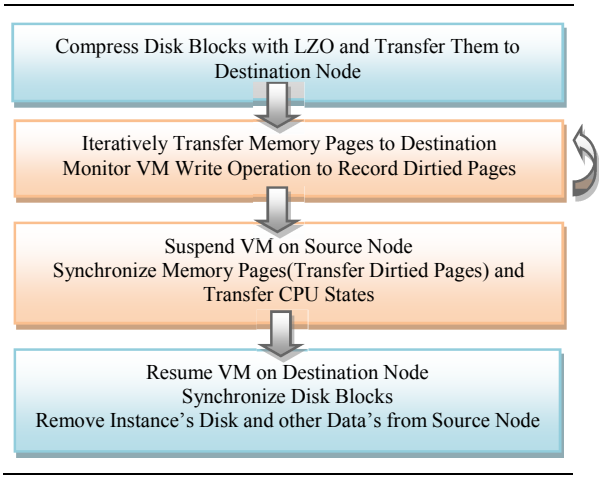


Figure 5. Steps of our migration method

VI. IMPLEMENTATION AND EVALUATION

In Eucalyptus environment, each running instance gives service to one or more customers; so, in instance migration process, the total migration time, network throughput and response time (to customers) are very important and must be evaluated. We could use other experiments such as disk performance evaluate (with the bonnie++ benchmark), but when the instance's disk must be transferred, this evaluation would not be helpful. We used two scenarios to evaluate our migration technique with above criteria. Each experiment was performed three times and average values were recorded.

A. Evaluation Environment

In order to implement Eucalyptus components and create a Cloud environment, we used two machines with AMD Quadro Core 800 MHz processor with disk capacity 500 GB and 8GB memory as NCs. Furthermore, we installed Xen 3.4 hypervisor on the NCs. Also, we have a machine with Intel Core2Duo 2.66GHz CPU, 320 GB disk capacity and 4 GB of memory as CLC and CC. These three machines are connected through a LAN network with 100 Mbps bandwidth. The migration operation is performed on the instance; it has one vCPU, 2GB disk space with 128 MB memory. On all components, the Linux Centos 5.6 OS is also installed. You see evaluation environment in Figure 6.

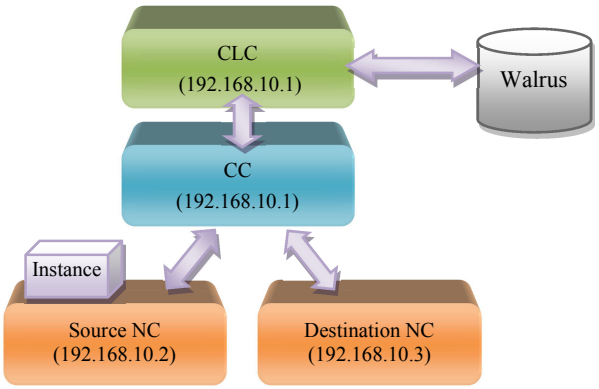


Figure 6. The Cloud Implemented Environment

B. First Test: Network Throughput

In the first experiment, we evaluated the network bandwidth of instance during migration in terms of throughput. In this test, we have used Netperf benchmark, version 2.5.0 in order to measure the network bandwidth in normal conditions (not using migration technique) and when the instance is migrating. You see the test results in Figure 7. As one can see in the figure, when the instance is suspended and moved from origin to destination, throughput rate of the network is reduced by half.

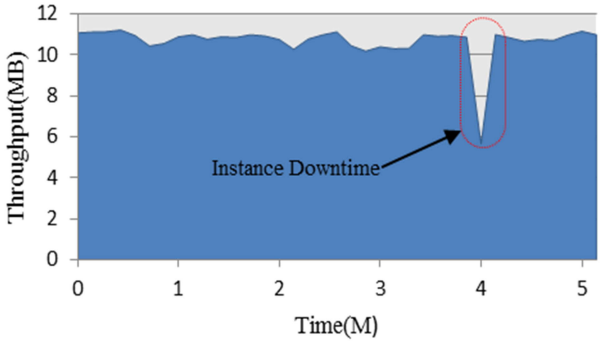


Figure 7. The Throughput of the Network during Instance Migration

C. Second Test: Response Time

The aim of this test is to evaluate response time to incoming requests by the instance during the migration operation. In this test, the instance functions as a server that provides a service to the users and begins migrating from a place to another place.

 ISBN: 978-1-61208-216-5

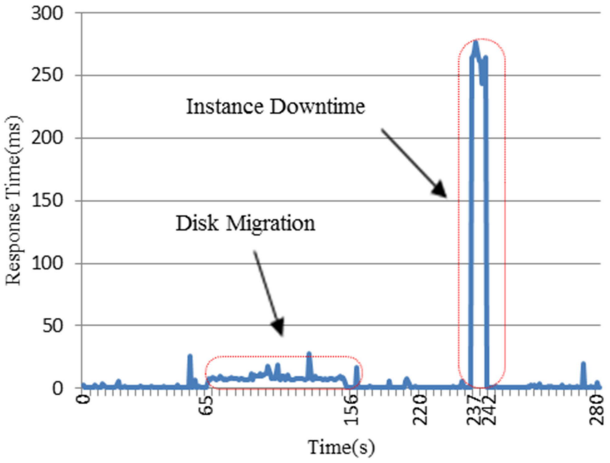


Figure 8. Response Time to Incoming Requests during Instance Migration

The results of this test have been shown in Figure 8. At first, when the migration process is starting, disk blocks compress. Afterward, disk transfer starts in second 65 and end on second 156. Now, destination node decompresses received disk blocks and then sends a message based on preparing to receive memory pages to source node. Next, in second 220, the operation of transferring memory pages is started. After a few seconds, in second 237, the instance was entered in suspend state. Now, memory pages and CPU states are transferred to destination. After a few seconds, in second 242, instance will resume working on the destination node and synchronization operation of disk blocks between source and destination was began from second 243 to 280.

VII. CONCLUSION

In this paper, we proposed a way to implement the migration of virtual machines on Eucalyptus Cloud Infrastructure. As mentioned, Eucalyptus has a unique architecture; thus, our method should be compatible with this architecture, so our main purpose is compatibility. Considering that there has been no migration feature in this cloud environment yet, other management features and algorithms such as load balancing and power management have not been implemented. Through using the method, we provided the field with further development and empowerment of the Cloud environment, and paved the way for the creation of more powerful algorithms in future.

REFERENCES

[1] Sh.Z. Rad, M.S. Javan, and M.K, Akbari, “A survey on virtual machine migration methods and performance evaluations”, First CSUT Conference on Computer, Communication, Information Technology (CSCCIT), Tabriz, Iran, 2011.

[2] C.P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M.S. Lam, and M. Rosenblum, "Optimization the migration of virtual computers", In Proceeding of 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI-02), December 2002.

[3] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, “Live migration of virtual machines”, In Proc. of the second USENIX Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA, USA, May 2005.

[4] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schioberg, “Live wide-area migration of virtual machines with local persistent state”, VEE’07, June 2007.

[5] M. R. Hines and K. Gopalan, “Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning”, Proceeding of the 2009 ACM SIGPLAN/SIGOPS International Conf. on Virtual Execution Environments, Binghamton University (State University of New York), Feb 2009.

[6] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, “Live migration of virtual machine based on full system trace and replay”, in Proceedings of the 18th International Symposium on High Performance Distributed Computing (HPDC’09), 2009, pp.101–110.

[7] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “Eucalyptus open source Cloud-computing system”, In CCA08: Cloud Computing and Its Applications, 2008.

[8] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/, [retrieved: May, 2012].

[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. “Xen and the art of virtualization.” In SOSP ’03: Proceedings of the nineteenth ACM symposium on Operating systems principles, pages 164–177, New York, NY, USA, 2003. ACM.

[10] M.F.X.J. Oberhumer, “LZO – a real-time data compression library”, http://www.oberhumer.com/opensource/lzo/, [retrieved: May, 2012].

 ISBN: 978-1-61208-216-5

Proactive Performance Optimization of IT Services Supply-Chain Utilizing a Business Service Innovation Value Roadmap

Short Paper for a work-in-progress

Ethan Hadar
Corporate Technical Strategy,
Distinguished Engineer
Senior Vice President
CA Technologies, Inc.
Herzelia, Israel
ethan.hadar@ca.com

Jason Davis
Sr Principal Services Architect
CA Technologies, Inc.
Berkshire, GB
jason.davis@ca.com

Donald F. Ferguson
Chief Technology Officer,
Distinguished Engineer
Executive Vice President
CA Technologies, Inc
New York, NY
donald.ferguson@ca.com

Abstract— **Business investment in IT is increasingly linked to IT delivering new or enhanced services that leverage the capabilities of the cloud. The challenge that IT faces especially in cloud environment, is to continuously assess and proactively optimize the performance and quality of supporting IT services being delivered. This evaluation must be performed in the context of the overall Service Level Agreements (SLA) of the composite application and subsequently, the underlying dynamic compound IT services. Our paper presents a proactive optimizer solution that provides on-going service improvement driven by the regular evaluation alternatives of the performance if individual services. Through a Business Service Innovation value roadmap, the Enterprise Architect can model and assemble candidate services for composite applications and automatically use tools to deploy and assure the performance of the composite application. Using our solution, over time the architect can manage the composite application by comparing the quality delivered against simulated alternatives and make recommendations for change. Consequently, using the solution presented in this paper, IT changes are aligned to the challenge to leverage the constantly improving quality of supply-chain IT service while maintaining or reducing costs.**

Keywords: Business Service Innovation; cloud optimization; cloud quality; IT services supply-chain

I. INTRODUCTION

In cloud computing, competition among service providers is affecting the flexibility and dynamics of possible combinations of underlying services within the IT Service supply-chain. This flexibility enables proactive, predictive optimization for both cost reduction and revenue increase [4] of the underlying supporting services.

Business transactions flow through the composite application underlying services and supporting hardware and software resources. Leveraging cloud computing, these services can conceptually be replaced with smart self-service and automation tools [5].

The composite application owner's challenge is to constantly innovate and improve their business service quality while reducing cost. The owner can achieve this goal by proactively adapting and optimizing the composite application's underlying supply-chain services, and composite IT systems. The optimization recommendations are considered based on qualified and quantified metrics. Through frequent recalculation, this dynamic adaptation can drive down Operational and Capital Expenditure (OPEX and CAPEX), raise Service Level Agreements (SLA) quality, and adhere to increased security and privacy compliance needs [1][7].

This paper suggests a system that assists the Enterprise Architect in replacing existing IT supply-chain cloud services according to business needs. The suggested refactoring changes (replacements) to existing IT services and process are based on configured goals for improving internal SLA. The system utilizes services from internal and external, private and public clouds (SaaS, PaaS and IaaS).

Section 2 of this paper presents the conceptual lifecycle framework, the Business Service Innovation upon which our system is structured. Section 3 presents the value for consumers and users of our solution. Section 4 details the necessary conceptual elements, followed by Section 5 that describes the technical modules that implement this solution. Section 6 highlights the value of our solution with associated needs for extensions and future work.

II. STATE OF THE ART

In previous work, we proposed a data store that contains normalized metrics of the services quality based on a Complex Event Processing (CEP) engine [3].

The CEP system is extended in this paper into the "proactive performance optimizer" solution that compares and proposes alternatives to the underlying services of the IT supply-chain. The proposed system's main principles follow a predetermined Business Service Innovation value roadmap (Figure 1), which structures our systems' management steps.

 ISBN: 978-1-61208-216-5

The Business Service Innovation steps are *Model-Assemble–Automate-Assure* and overall *Manage* the evolution, interwoven with IT *Security*.

The detailed steps are:

1. *Model* and *Assemble* a list of applied service elements (composite IT systems and composite applications), and present alternative options to the supporting supply-chain services (candidate services).
2. *Automate* the deployment and *Assure* the quality all underlying services (the ones that participate in the composite application, as well as the candidate services that are not currently part of the composite applications but could replace an existing service.)
3. *Manage*, regularly evaluate the quality and indicate/highlight if an alternative candidate service is superior to the one currently in use, as compared to predefined filter and search criteria.
4. Return to the first step, and accept or reject (manually or automatically) the recommended changes by remodeling the change.

Less common proactive optimization for reducing costs will be to replace a single element with high performance attributes that have a high cost, due to the nature of the high quality SLA, with a lower quality one. This might be the case when this instance (service) is coupled with other transactions, which have a much lower aggregated SLA due to other services and components. The overall SLA is the lowest common denominator, therefore paying a high price for a quality service that does not get used could be considered wasteful.

This example illustrates just one dimension of optimization, in which the cost alternative of the underlying services that participate in a composite transaction can be replaced at any given point in time.

Naturally, there are many other dimensions for proactive optimization such as increasing availability, improving load, increasing speed for change, better robustness and more. Compliance and liability as well as and insurance coverage are additional examples for improvement and replacement [6].

III. THE SYSTEM VALUE

Unlike existing process and composite application design systems that structure a service or a process from a stable state environment point of view, the proposed system constantly improves and evolves a flexible business process. The system operates in a cloud-computing environment that is categorized by a changing environment and service composition possibilities, due to an open market and ease of change. In addition, the proactive optimization system is applicable for non-cloud services as well, and offers the same solution in a system that does not change as often as cloud alternative.

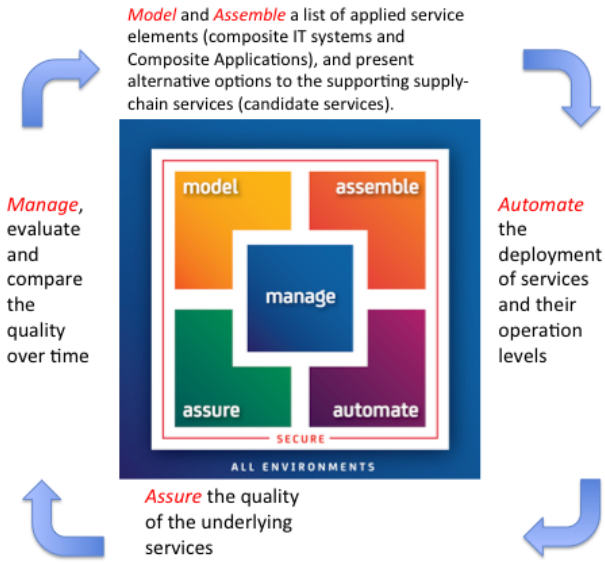


Figure 1: Business Service Innovation value roadmap of the proactive optimizer.

Thus, the proactive optimization of cloud service performance aspects provides several unique value propositions.

- Ongoing suggestions for changes to existing well-designed solutions that could be improved through modifications to underlying services (proactive optimization).
- Notifying the designer of plausible alternatives for existing consumed services.
- Improvement of a monitored service which may be applicable for a certain customer, but considered inappropriate for a different customer due to consumer-specific SLA and quality goals.
- The solution couples the abilities to monitor and compare thresholds of public cloud services (or any service for that matter) with predetermined service levels of consumers, as well as scanning alternative similar services for optional replacements.
- Apply an agile approach for regular incremental and iterative improvement of processes composite applications in production.
- Rationalize the change of the entire portfolio of composite application based on an overall aggregated quality rather than on the underlying single service.

IV. CONCEPTUAL STRUCTURE

This paper presents a refactoring service that continuously monitors possible underlying IT services within the context of a supply-chain of IT services, supporting a composite application. Based on complex event processing (CEP) and predefined threshold metrics, refactoring service triggers assessment of the suggested changes to the optimized services. The system's main modules are SLA thresholds and triggers, and the detection of needed/recommended change.

 ISBN: 978-1-61208-216-5

A. SLA Thresholds and Triggers

The SLA's that are associated with the supplied services that are part of the overall composite application determine the level of aggregated service or SLA that can be achieved. Improvement of the SLA for an overall service is driven by the SLA's or OLA's (operation level agreement) that relate to the underlying service. The system can either monitor a single most impactful supplied service as a candidate for improvement, or, monitor the overall SLA/OLA [6] with a mathematical weighting of the individual contribution of each supply-chain IT service. With a focus on improving the overall service, rather than a single service, the system provides a mechanism that allows the user to set goals for SLA improvement on the overall composite application, and sub-divide it in to the internal services derived OLA/SLA/goals. In the case of self-adaptation, these rules will trigger a suggestion for a change to a service, in the form of a Change Recommendations or external Service Design process.

B. Detection of change in Quality of Service using Complex Event Processing

The Complex Event Processing engine continuously scans internal and external clouds for detection of quality changes to composite elements such as SaaS, IaaS or PaaS that make up an overall composite application. These monitoring tools for service assurance scan for alternatives for improved metrics values.

Once data is collected, the CEP [3] system correlates the information gathered using specific formulas to determine the overall improvement or degradation of quality of service that is being delivered.

C. What-if modeler

For each of the proposed alternatives, an aggregated overall potential SLA is presented. Several of these alternatives can be presented and maintained in the modeler component that captures the structure of the composite application, presenting, over time, the trends and possible quality levels.

V. IMPLEMENTATION STRUCTURE

This section presents the implementation modules and a prototypical usage scenario to the solution according to the Business Service Innovation value roadmap (Figure 2).

A. The implemenation modules

The participating modules [2] are:

- CA AppLogic - constructs and test composite IT systems that support the composite application. The module defines the architecture structure and IT system physical dependency, load balancing, and network configurations.
- CA Service Operations Insight (CA SOI) – monitors internal cloud services and implements the CEP system. This module also defines the behavioral dependency of the supplied services and measured SLA/OLA.
- CA Application Performance Management Cloud Monitor (CA APM) - monitors external cloud services.
- CA Business Service Insight– presents alternative suggestions for change based on measured reported metrics of the vendors as well as aggregated statistics based on surveying users.
- CA Automation Suite for Clouds – automates the changes in infrastructure provisioning and capacity.
- CA IT Process Automation Manager (CA ITPAM) – supports automation of changes in more complex structures, triggering federated identity provisioning, or incident management.
- CA Performance Optimizer - providing preconfigured optimization capacity changes to private datacenter (for infrastructure services).

B. A prototypical usage scenario

In order to realize the connectivity between the modules, consider prototypical activation by an Enterprise Architect.

In this scenario, the enterprise architect selects services, denoted as "atomic services" for use in the composite applications. The services are selected from a list in CA APM Cloud Monitor and CA Business Service Insight (external services) and from a list of infrastructure components (CA AppLogic). The infrastructure services are combined using the CA AppLogic modeler. The integrated external IT services are mapped on a behavioral model presented, and later on monitored, by the CA SOI modeling tool. The structure of CA SOI Composite Application model and the CA AppLogic Composite IT System model defines what underlying services are candidates for changes.

These replaceable services are frequently compared with other options that provide the same conceptual service, yet, currently provide worse SLA, or cost more. CA APM Cloud Monitor and CA Business Service Insight provide the list of compared services.

A comparative "what-if" structure of a potential alternative to the composite application is calculated using a simulator instance of CA SOI behavioral model. Note that this is not the production system version of CA SOI, rather a testing system. In this case CA SOI a combination of production services as well as alternative underlying services. The Enterprise Architect defines the candidate services that can be considered for replacement, and limits the search space for design alternative. A fully opened optimization is not practical, due to contractual limitations, as discussed in the next section.

On a regular basis, the overall SLA for the composite application is calculated for the production composite application, and is constantly compared to the candidate

 ISBN: 978-1-61208-216-5

composite application in which some of the supply-chain services are allowed to change. For every permutation possible, a relative aggregated SLA is presented to the Enterprise Architect, over time.

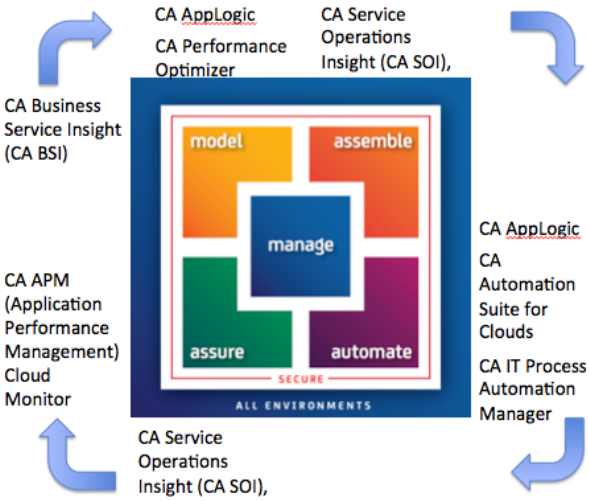


Figure 2 – Proactive optimizer Business Solution Innovation implementation modules.

If over a predefined interval the comparison shows improvement, change automation tools can implement an architect approved change using either CA Cloud Automation suite or ITPAM.

Accordingly, this procedure is repeatable, assuming services are added or removed, as captured on the contractual agreements defined in CA Business Solution Insight.

VI. DISCUSSION AND CONCLUSION

We presented our work-in-progress that provides a proactive optimization solution for enabling ongoing replacement of IT services in cloud environments. Activated according to a Business Service Innovation value roadmap, the solution leverages SLA performance measurements of existing production level applications and their underlying composite IT systems, compared against simulated and monitored alternatives. Consequently, our implemented solution reduced costs and/or improved quality, thereby addressing the challenges of the enterprise architect while providing the business rational for the change. If the change may be applicable to other situations, automation tools can activate the change over dynamic and elastic cloud environments.

However, change typically has associated cost and risk factors that can impact production systems. As a result, change activation should be performed only if the costs savings or ROI is higher and associated risk is lower than the existing state. With pure dynamic resource allocation management over virtual environments, these considerations can be eliminated.

The system does have its limitations; the computational change is not a complex optimization problem as it may be considered. The reason is that not all of the supply-chain IT services can change due to contractual agreements and limitations of liability. However, the complexity is noticeable when overall balancing of all the composite applications in the enterprise are considered, in particular in the domain of mashup and situational applications. Our future research work involves handling optimization for composite situational applications for the entire enterprise.

Even more, optimization is subjective to each customer, based on financial and quality based needs. Different internal consumers may require different service levels. As a result, the ability will be to best match a given SLA required level with supporting provider, changing the proactive optimization to a matchmaking algorithm or better yet, feasibility constraints target function.

Our future work is focusing on providing matchmaking optimization within feasibility box-constraints for assigning the best available supply-chain services

VII. REFERENCES

[1] Blum D., Schacter P., Maiwald E., Krikken R., Henry T., Boer M., and Chuvakin A., “2012 Planning Guide: Security and Risk Management”, G00224667, Burton IT1 Research, 1 November 2011

[2] CA Web site for tools and products, http://www.ca.com/us/products.aspx, last accesed on May 10 2012

[3] Gal A. and Hadar E., book chapter: “Generic Architecture of Complex Event Processing Systems”, in “Handbook of Research on Advanced Distributed Event-Based Systems, Publish/Subscribe and Message Filtering Technologies”, edited by Annika Hinze and Alejandro Buchmann, IGI Global press, 2009

[4] Ferguson D.F. and Hadar E., “Optimizing the IT business supply chain utilizing cloud computing”, The 8th International Conference on Emerging Technologies for a Smarter World (CEWIT2011), page 1-6, Hyatt Regency Long Island, Hauppauge, New York, November 2-3, 2011.

[5] Hadar E., Connelly K., and Lagunova O., “Agile Evolution of Information Systems Using Model Driven Architecture of IT Services”, Proceeding of the “Architecture in an Agile world” workshop, October 25, OOPSLA 2009, Orlando, Florida, US, 2009

[6] Hadar E. and Danielson D.J., “Certified IT services in a box for cloud computing environments”, CLOSER 2012, 2nd international conference for cloud computing and service sciences, Porto, Portugal, 18-21 April 2012.

[7] Hadar E., Hadar I. and Ferguson D.F., “QDSL - Quality Domain Specific Language for cloud composite applications”, CLOSER 2012, 2nd international conference for cloud computing and service sciences, Porto, Portugal, 18-21 April 2012.

 ISBN: 978-1-61208-216-5

Load Balancing in Cloud Computing Systems Through Formation of Coalitions in a Spatially Generalized Prisoner's Dilemma Game

Jakub Gasior
Systems Research Institute, Polish Academy of Sciences
Warsaw, Poland
E-mail: j.gasior@ibspan.waw.pl

Franciszek Seredynski
Polish-Japanese Institute of Information Technology
Warsaw, Poland
E-mail: sered@pjwstk.edu.pl
Department of Mathematics and Natural Sciences
Cardinal Stefan Wyszynski University
Warsaw, Poland
E-mail: sered@pjwstk.edu.pl

***Abstract*—The efficiency, in terms of load balancing and scheduling problems as well as security of both communication and computation processes, belong to the major issues related to currently built cloud computing systems. We present a general framework to study these issues and our research goal is to develop highly parallel and distributed algorithms working in environments where only local information is available. In this paper we propose a novel approach to dynamic load balancing problem in cloud computing systems. The approach is based on the phenomena of self-organization in a game-theoretical spatially generalized Prisoner's Dilemma model defined on the two-dimensional cellular automata space. The main concept of self-organization used here is based on the formation of temporal coalitions of participants (computational nodes) of the spatial game in the iterative process of load balancing. We present the preliminary concept design for the proposed solution.**

Keywords-Cloud computing; Cellular automata; Load-balancing; Spatial prisoner's dilemma.

I. INTRODUCTION

Cloud computing is one of the emerging developments in distributed, service-oriented, trusted computing. It offers the potential for sharing and aggregation of different resources such as computers, storage systems data centers and distributed servers. The goal of a cloud-based architecture is to provide some form of elasticity, the ability to expand and contract capacity on-demand. That means there needs to be some mechanism in place to balance requests between two or more instances of client's applications. The mechanism most likely to be successful in performing such a task is a load balancer.

It provides the means by which instances of applications can be provisioned automatically, without requiring changes to the network or its configuration. It automatically handles the increases and decreases in capacity and adapts its distribution decisions based on the capacity available at the time a request is made.

In this paper, we consider the aspect of effective load balancing, i.e., the process of distributing the load among various nodes of a distributed system to improve both resource utilization and job response time. The load can be defined as CPU load, memory capacity, delay, network load, etc. We formulate a purely theoretical conceptual model defined as follows: given a set of virtual resources in the Cloud ($M_1, M_2, ..., M_n$), a number of cloud clients ($U_1, U_2, ..., U_k$) and a random set of applications (also jobs or tasks) run by the clients ($J_1, J_2, ..., J_i$), find such an allocation of jobs to the resources to equalize the system workload [1].

We are interested in parallel and distributed algorithms working in environments with only limited, local information. Therefore, we propose a game-theoretical approach combining a spatially generalized Prisoner's Dilemma (SPD) model and the cellular automata (CA) paradigm. Each computational node is presented as a selfishly rational agent. Such a problem formulation is alike to a CA in the sense that the strategy first determines the rule based on the neighbors' configuration and the rule in turn determines the next action [2].

Competing players in such a system should act as a decision group choosing their actions in order to realize a global goal. Main issues that must be addressed here are: a) incorporating the global goal of the multi-agent system into the local interests of all agents participating in the game; and b) such a formulation of cellular automata's local rules, that will allow to achieve those interests [12].

The paper is organized as follows. The following section presents the basic concepts of spatial Prisoner's Dilemma game and cellular automata theory. Section 3 presents our mathematical model of cloud computing system. Section 4 details the load-balancing algorithm from the game theoretical point of view. Finally, Section 5 provides some concluding remarks.

II. PRISONER'S DILEMMA AND CELLULAR AUTOMATA

The concept of the evolution of cooperation has been successfully studied using various theoretical frameworks.

 ISBN: 978-1-61208-216-5

Table I
A GENERAL PRISONER'S DILEMMA PAYOFF MATRIX

	Cooperate	Defect
Cooperate	*(R,R)*	*(S,T)*
Defect	*(T,S)*	*(P,P)*

In particular the Prisoner's Dilemma (PD) is one of the most commonly employed games for that purpose, a type of non-zero sum game played by two players who can choose between two moves, either to cooperate with or defect from the other player. The problem is called the prisoner's dilemma, because it is an abstraction of the situation felt by a prisoner who can either cut a deal with the police and tell on his partner (defect) or keep silent and therefore tell nothing of the crime (cooperate). While mutual cooperation yields the highest collective payoff, which is equally shared between the two players, individual defectors will do better if the opponent decides to cooperate. The key tenet of this game is that the only concern of each individual player is to maximize his payoff during the interaction, which sets the players as naturally selfish individuals.

The dilemma arises when a selfish player realizes that he can not make a good choice without knowing what the opponent will do. Non-zero sum describes a situation where the winnings of one player are not necessarily the losses of the other [4]. As such, the best strategy for a given player is often the one that increases the payoff to the other player as well. *Table I* shows a general payoff matrix, which represents the rewards an entity obtains depending on its action and the opponent's one. In this matrix, *T* means the *Temptation* to defect, *R* is the *Reward* for mutual cooperation, *P* the *Punishment* for mutual defection and *S* the *Sucker's payoff*. To be defined as a PD, the game must accomplish the condition $T > R > P > S$.

This payoff structure ensures that there is always the temptation to defect since the gain for mutual cooperation is less than the gain for one player's defection. The outcome *(D,D)* is therefore a *Nash equilibrium* - despite the knowledge and awareness of the dilemma, both players opt to defect even though both know they are going to receive inferior scores [7]. In terms of evolutionary game theory *defection* is the unique evolutionary stable strategy (ESS) [8].

Nowak and May [3] have proposed a way to escape from the dilemma. A variation of prisoner's dilemma game working in the two-dimensional cellular automata space where agents are mapped onto a regular square lattice with periodic boundary conditions. In every round, players interact with the immediate neighbors according to a strategy. The fitness of each individual is determined by summing the payoffs in games against each of its neighbors. The scores in the neighborhood, including the individual's own score, are typically ranked. In the next round, all individuals update their strategy deterministically. This approach is typical for cellular automata models. From a biological perspective, the utility of an individual is interpreted in terms of reproductive success. Alternatively, from an economic perspective, the utility refers to individuals adapting their strategy to mimic a successful neighbor [7].

Nowak and May have shown that such spatial structure enables the maintenance of cooperation for the simple Prisoner's Dilemma, in contrast to the classical, spatially unstructured Prisoner's Dilemma where defection is always favored. It was determined that players do not need to play the game with the whole population. By making this assumption, different equilibria are likely to be established in different neighborhoods. More importantly, the spatial structure allows cooperators to build clusters in which the benefits of mutual cooperation can outweigh losses against defectors [2]. Thus, clusters of cooperative strategies can invade into populations of defectors that constitute an ESS in non-spatial populations [3].

III. PROBLEM FORMULATION

In this section, we formally define basic elements of the model and provide corresponding notation. Then, we define possible characteristics of the model that change the available information and the type of jobs to be scheduled.

For the sake of simplicity, it is assumed that every node placed on a two-dimensional cellular automata represents a *virtualized resource* (M_k) - an abstraction of an entity that process jobs. Computational power C_k of a certain resource M_k is defined by a number of operations per unit of time it is capable of performing. We distinguish between cooperative (job taking) nodes and selfish (non-job taking) nodes. The motivation for non-cooperative nodes to enter the cloud is to just use resources to fulfill their own processing tasks in the role of clients and refuse to contribute as a worker (although they could due to their capabilities). Note that if nodes do not benefit from cooperation incentives (e.g., the possibility to submit jobs to others in the future), selfishness will be the optimal strategy for each node.

Job (denoted as J_k) is an equivalent of application run by the cloud clients. Every application is independent and has no link between each other whatsoever, e.g., some require more CPU time to compute complex tasks, and some may need more memory to store data, etc. Resources are sacrificed on activities performed on each individual unit of service. In order to measure direct costs of applications, every individual use of resources (i.e., CPU cost, memory cost, I/O cost) must be measured. To simplify the problem, we assume that *job* is simply an entity that, in order to be completed, requires an access to a resource during certain time p_k. For the sake of the theoretical analysis, unless otherwise stated, we assume that the jobs J_k are produced by a Poisson process. The size of a job is known immediately after the job has arrived to the system. At any given time,

 ISBN: 978-1-61208-216-5

let the local load L_k stand for the time moment when the computation of the last currently known local job ends, thus it can be defined as ratio between total size of node's queued jobs and its computational power:

$$L_k = \frac{\sum_{i=1}^{n} p_k^i}{C_k}, \quad (1)$$

where: *n* stands for the total number of jobs assigned to a single node.

Informally, the goal of the scheduler is to find the allocation and the time of execution for each job. The distribution of the tasks must be done in such a way that the system's throughput is optimized. All scheduling and load balancing decisions are taken locally by the agents. The algorithm analyzes the node's status in terms of its utilization and capabilities. This status is matched against the job's requirements (as given by the job's meta-data, p_k) considering user-configurable policies that define the desired degree of resource contribution. Subsequently, each node may begin execution of assigned tasks, or split them among its neighbors.

Ideally, each node should receive the same (or nearly the same) number of tasks. If the same amount of work is associated with all the nodes, equal distribution of tasks ensures a good load balance. This statement holds true assuming that communication cost between neighbor nodes is negligible. However, such an assumption is unlikely to be fulfilled in real-world environments. Thus, we introduce one more parameter defining the amount of time needed to transfer the workload from one node to another and denote it as q_{ij}, where: *i* and *j* stand for identifiers of nodes participating in the exchange. For simplicity's sake, we assume that communication cost between neighbor nodes is equal to *one*, and grows linearly with each additional cell, except, of course a node may communicate with itself at no cost.

It is important to note that, in this work we make very few assumptions. We can deal with either static or dynamic load. The network topology can be of any type as long as it is connected. Nodes and networks can be homogeneous or heterogeneous. Load balancing algorithms are operating in a fully localized, distributed fashion. The required knowledge is limited to the computation speed, local workload of the neighbors and the computation time per one unit of load. All these information are supposed to be given, calculated or estimated.

IV. The Dynamic Load Balancing Problem

We wish to distribute the workload among resources of the system to minimize both: a) *load imbalance* and b) *communication cost* between them. For that purpose, a set of cellular automata's local rules must be evolved according to a specific *utility function*. Let us start by defining the cost and the benefit of a load balancing process. The cost is the time lost by exchanging the workload, due to communication. The benefit is the time gained by exchanging the workload, due to a better balance and faster execution of tasks.

Let E_{ij} stand for the exchange of workload between nodes *i* and *j*. The benefit given by the exchange E_{ij} can be estimated by the computation time on *i* and *j* without the exchange minus the computation time on *i* and *j* after this exchange [1]. Intuitively, the benefit of a load exchange must be positive if the computation time is reduced by this exchange and negative in the other case. The following equation denotes the benefit of load balancing scheme, assuming that node *i* transfers workload to node *j*:

$$\begin{aligned} \text{Benefit}(E_{ij}) = max(L_i, L_j) - \\ max(L_i - E_{ij}, L_j + E_{ij}), \end{aligned} \quad (2)$$

where L_i and L_j define local loads on nodes *i* and *j*, respectively. Let us now consider the communication part of the load balancing process. The cost of communication from one node to another depends on the network architecture (i.e., network bandwidth, network traffic, buffer size). A truly portable load balancing algorithm would have no option but to send sample messages around and measure those metrics, then distribute the workload appropriately. In this paper, however, we shall avoid this question by assuming that all pairs of computational resources are equally far apart. We can make the assumption that the total communication cost is equal to the amount of time needed to transfer the workload from node *i* to node *j* (denoted as q_{ij}) and thus:

$$\text{Cost}(E_{ij}) = q_{ij}. \quad (3)$$

Additionally, we make an assumption that any node which took part in the balancing operation is obliged to return resulting data to the originating node. This issue can be solved by simply propagating the results backwards through the initial load balancing route. Such a problem formulation, however, may become ineffectual in a case of large quantities of workload being shared among many neighboring nodes. It is possible, that in such a case, there exist an alternative way back to the originating node; shorter than original load balancing route. The issue is illustrated in Figure 1, where *A*, represents *source node*, and *B* represents *destination node*. Green line indicates original load balancing route, while red line shows the optimal way back.

We propose a simple solution to this problem by implementing a gradient-based communication model. We define the node's *proximity (P)* as the shortest distance from itself to the sender node. All cells are initialized with a proximity of P_{max}, equal to the diameter of the system lattice. The proximity is set to 0 if node becomes overloaded and its state changes to *sender*. All other nodes *i* with local neighbors n_i, compute their proximity as:

 ISBN: 978-1-61208-216-5

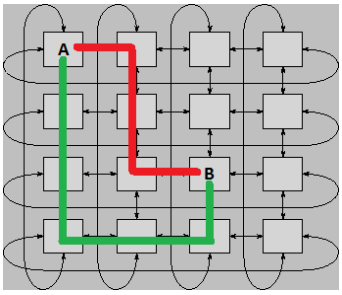


Figure 1. The issue of determining communication cost between *source node (A)* and *destination node (B)*. Green route shows original communication route according to the load balancing algorithm. Red route indicates an alternative (optimal) way back.

$$P(i) = min(P(n_i)) + 1. \quad (4)$$

The resulting proximity map is later used used to perform the migration phase. Results are routed through the system in the direction of the sender node (Figure 2).

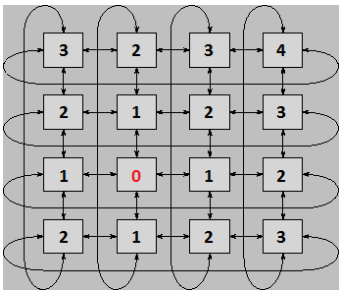


Figure 2. The gradient-based communication model. Computational nodes send results in the direction of the sender node (red) via the gradient map of proximity values. Cellular automata space comprises the von Neumann neighborhood - the four cells orthogonally surrounding a central cell on a two-dimensional square lattice.

Given this parameter, the cost function of load balancing process from Equation 3 can now be extended and denoted as:

$$\text{Cost}(E_{ij}) = q_{ij} + P(i), \quad (5)$$

assuming that node *i* is transferring its workload to node *j*. Such a formulation is possible because node's proximity is equal to the amount of time needed for propagating the results back to the originating node. Additionally, it ensures that load balancing profitability is decreasing linearly with an increase in distance from the source.

We may now construct our *utility function*, Γ, as the sum of parts describing benefits and costs of the load balancing operation, respectively:

$$\Gamma = \sum_k \text{Benefit}(E_{ij}^k) - \mu \sum_k \text{Cost}(E_{ij}^k), \quad (6)$$

where: *k* denotes the amount of workload exchanged between neighbor nodes and μ is a parameter expressing the balance between the two aspects of load balancing scheme - communication and computation. For programs with a great deal of calculation compared to communication, μ should be relatively small, and *vice versa*. As μ increases, the number of processors in use will decrease until eventually the communication is so costly that the entire calculation must be done on a single node. Score calculated according to Γ is awarded to every node taking part in the load balancing scheme. Its magnitude is strictly dependent on agent's action taken in the PD game as shown in *Table II*.

Table II
THE PRISONER'S DILEMMA RESCALED PAYOFF MATRIX

	C (Send load)	D (Compute locally)
C (Accept)	$\Gamma/2$, $\Gamma/2$	*0, 0*
D (Reject)	Γ, 0	*0, 0*

After *s* (*strategy update cycle*) steps of interactions with the neighbors, all nodes are presented with an opportunity to update their strategy in a similar manner to the standard SPD game. The present set of strategy imitation rules is based on pairwise comparison of payoffs between two neighboring agents. In each subsequent elementary step of the evolutionary process we choose two neighboring players (*i* and *j*) at random, we determine their payoff G_i and G_j, and player *i* adopts the strategy s_j with a probability given by the Fermi-Dirac distribution function as proposed in [9]:

$$W(s_i \leftarrow s_j) = \frac{1}{1 + exp[(G_i - G_j)/K]}, \quad (7)$$

where: K characterizes the uncertainty related to the strategy adoption process, serving to avoid trapped conditions and enabling smooth transitions towards stationary states [5].

It is well known that there exists an optimal intermediate value of K at which the evolution of cooperation is most successful [6, 10], yet in general the outcome of the PD game is robust to variations of K. For $K \ll 1$, selection is weak and the payoffs are only a small perturbation of random drift. For $K \gg 1$, selection is strong and the individual with the lower payoff will change its strategy. In statistical physics, K is the inverse temperature: for $K \to 0$, the dynamics of the system is dominated by stochasticity (the temperature of selection is high), whereas in the limit $K \to \infty$ stochastic effects can be neglected (the temperature of selection is zero) [11]. This phenomenon is fully illustrated in Figure 3. Without much loss of generality, we use $K = 0.1$, meaning that it is very likely that the better performing players will pass their strategy to other players, yet it is not impossible that players will occasionally learn also from the less successful neighbors.

It can be seen that agent's performance in the dynamic load balancing scheme directly affects its scores acquired in the PD game, by shifting the magnitude of payoff values. Thus, agent with a more effective balancing strategy will

 ISBN: 978-1-61208-216-5

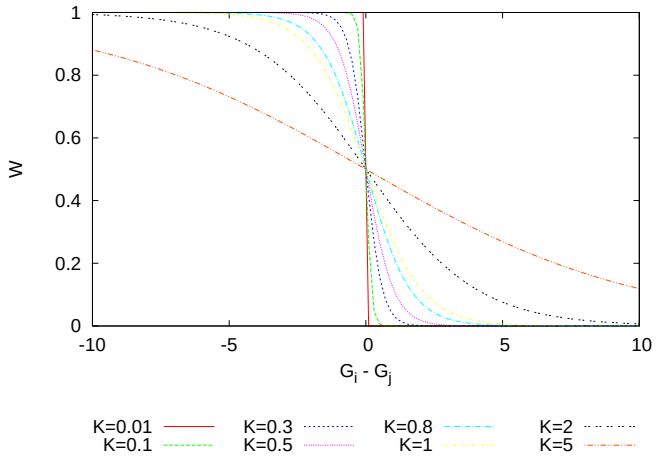


Figure 3. Strategy adaptation probability graph as a function of the payoff difference and variable K, characterizing the uncertainty related to the strategy imitation process.

acquire higher scores in the PD game, which in turn will increase the probability of imitating that strategy by his less successful neighbors and propagating it in the system. This in turn should lead to an optimal load distribution in the cloud computing environment.

V. Conclusion and Future Work

We have proposed in this paper a novel paradigm for a parallel and distributed evolutionary computation in cloud computing systems based on the model of spatio-temporal Prisoner's Dilemma game. We presented the rules of a local interaction among agents providing a global behavior of the system as well as the analysis of costs and benefits of workload exchange. Game-theoretic approach allowed us to model organizational heterogeneity of cloud computing systems. Currently, the model is a subject of the experimental study.

Our future work is threefold. Firstly, we want to further enhance our model in order to study the problem of evolution of global behavior and formation of coalitions between agents. Secondly, we intend to extend the model to enhance security of both communication and data processing. In particular, we want to focus on aspects of reputation and cryptography. This could be important, for instance, when agents have to decide which action to take against outsiders. If these outsiders have a reputation degree, such information could be used in the decision-making process. Also, reputation may turn important among members of coalitions themselves, for instance to decide when coalitions should be dissolved. Finally, we would like to port this solution to real-world scenarios that involve data networks such as P2P, sensor, and ad-hoc networks.

Acknowledgment

This contribution is supported by the Foundation for Polish Science under International PhD Projects in Intelligent Computing. Project financed from The European Union within the Innovative Economy Operational Programme 2007-2013 and European Regional Development Fund (ERDF).

References

[1] E. Jeannot and F. Vernier, "A practical approach of diffusion load balancing algorithms," pp. 211–221, 2006. [Online]. Available: http://dx.doi.org/10.1007/11823285_22

[2] Y. Katsumata and Y. Ishida, "On a membrane formation in a spatio-temporally generalized prisoner's dilemma," pp. 60–66, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-79992-4_8

[3] M. Nowak and R. May, "Evolutionary games and spatial chaos," *Nature 359*, pp. 826–829, 1992.

[4] M. Osborne, *An Introduction to Game Theory*. USA: Oxford University Press, 2003.

[5] M. Perc and A. Szolnoki, "Social diversity and promotion of cooperation in the spatial prisoner's dilemma game," *Physical Review E 77*, vol. 77, p. 011904, Jan 2008. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.77.011904

[6] M. Perc, "Coherence resonance in a spatial prisoner's dilemma game," *New Journal of Physics*, vol. 8, no. 2, p. 22, 2006.

[7] G. Rezaei and M. Kirley, "The effects of time-varying rewards on the evolution of cooperation," *Evolutionary Intelligence*, vol. 2, pp. 207–218, 2009, 10.1007/s12065-009-0032-1. [Online]. Available: http://dx.doi.org/10.1007/s12065-009-0032-1

[8] J. M. Smith, *Evolution and the Theory of Games*. Cambridge University Press, 1982.

[9] G. Szabó and C. Tőke, "Evolutionary prisoner's dilemma game on a square lattice," *Phys. Rev. E*, vol. 58, pp. 69–73, Jul 1998. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.58.69

[10] G. Szabó, J. Vukov, and A. Szolnoki, "Phase diagrams for prisoner's dilemma game on two-dimensional lattices," *Physical Review E*, vol. 72, p. 047107, 2005.

[11] A. Traulsen, M. A. Nowak, and J. M. Pacheco, "Stochastic payoff evaluation increases the temperature of selection," *Journal of Theoretical Biology*, vol. 244, no. 2, pp. 349–356, 2007.

[12] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.

Cloud Computing Brokering Service: A Trust Framework

Service Level Agreements: An Analytical Study in Progress

Prashant Khanna
Institute of Engineering and Technology
JK Lakshmipat University
Jaipur, India
e-mail: perukhan@gmail.com

Budida Varahala Babu
Institute of Engineering and Technology
JK Lakshmipat University
Jaipur, India
e-mail: profbvbabu@gmail.com

***Abstract*— The paper highlights existing research voids in defining and designing binding and enforceable service level agreements (SLA) between three actors in the cloud computing framework defined by NIST – the cloud brokers, the cloud consumers and the cloud providers. The paper presents a techno-managerial perspective to the issue of how cloud brokers would handle service provisioning and whether binding service level agreements would be useful tools for the NIST cloud framework to function. A template constituent framework is also recommended as part of this ongoing study.**

Keywords-Cloud Brokers; Cloud Computing; SLA; Service Provisioning; Trust.

I. INTRODUCTION

Cloud Computing is an emerging computing paradigm that promises to change the landscape of the present service models on offer in provisioning of Information Technology services. The "Cloud", as a term has found prominence in an increasingly large number of publications, both in the academia as well as in industry literature. It is a buzz word and the buzz is getting louder by the day. The definitions of cloud computing are many, and varied. The industry has, only in late 2011, finally decided to accept one that was proposed by National Institute of Standards and Technology (NIST), U.S. Department of Commerce [1]. As per the Draft Computing Technology Roadmap published by NIST, Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. The definition has listed five essential characteristics that would be common to all cloud computing services, namely: on-demand self service, broad network access, resource pooling, rapid elasticity and measured service. It recommends three service models: Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS), and four deployment models i.e. private, public, community and hybrid clouds.

The reference architecture in the NIST document highlights interactions amongst these entities and provides a companion cloud computing taxonomy detailing the definitions and relationships of a control vocabulary. The document also identifies five major actors to enable the reference model to work, namely, cloud consumer, cloud provider, cloud carrier, cloud auditor, and cloud broker. Each actor is an entity (a person or an organization) that participates in a transaction or process or performs tasks in cloud computing. A lot has been said and written about the model and the way the players interact in this model to derive services. Each of the players have been defined and redefined in literature and the use case(s) to make the model successful has also been commented upon extensively. Amongst the actors defined in the NIST model [1], the cloud broker was an add-on after much thought. Gartner, in a report in 2011 [2] indicated that cloud brokering services in the cloud service marketplace is emerging as a promising low-risk business model for offering new and value-added services through cross provider service delivery and partnership. This assertion has made a major impact on the industry as well as the academia.

Any service brokering architecture, in general, must have the ability to support a service delivery infrastructure for integration, delivery and management of composite services in a multi-provider heterogeneous networks environment. It is no different in the cloud service provisioning environment. In the present stage of evolution of the cloud as a repository of services, this provisioning is far from being ideally achieved. The cloud paradigm is currently in a state of transition and multiple players are trying to dominate the service delivery scene. The cloud providers are competing with the cloud brokers to deliver the intended service to the cloud consumer, but this model of business to consumer interaction is not bearing the desired results due to multiple barriers of scale and other managerial issues. This research on the subject, supported by the industry reports indicate that the player who is likely to emerge as the principal stake holder in provisioning and arbitraging of services as a truly elastic and dynamic package for the consumer would be the cloud broker. Such service provisioning is already appealing to the small and medium business entrants who are not yet as big as Google or Amazon, but have the understanding of how the cloud works [2]. Forrester [3], in their annual report in 2011, also cite brokering services in the cloud to be the next game changer in the service provisioning space. However, the present state of cloud implementation is highly proprietary and private, akin to islands of highly autonomous island solutions which do not have any linking ferry services which can carry the inhabitants across. The cloud brokering

 ISBN: 978-1-61208-216-5

service available today is thus confined to a miniscule subset of matching services that are seamlessly able to speak to each other. There is a serious void in interoperability between cloud solutions that are not been addressed by the present generation of brokering service providers, either due to technological incompatibilities or due to managerial issues. The present NIST framework for the cloud-based service model, as others similar frameworks, are based on adopting managerial practices in organizations which are implemented by using a preferred underlying technology. This is truer today with the inclusion of the cloud broker as an actor in the models under consideration.

We appreciate that this is as much a managerial issue as it is a technical one. This paper and research is an attempt to highlight existing research voids and present a techno-managerial perspective to the issue of how cloud brokers would handle service provisioning and whether binding service level agreements (SLA) would be useful tools for the NIST cloud framework to function. This is a work in progress and it is anticipated that the research would result in proposing a framework that would make service clouds talk to each other under a universally acceptable interoperability standard, where enforceable and automated SLA become corner stones for provisioning of dynamic and elastic services amongst the actors enumerated in the NIST model.

The structure of the remaining paper will be as given below: Section II presents the Literature Survey on the topic of SLA in the cloud and its relevance to the cloud brokering services, Section III addresses the constituents of an SLA within a service oriented business model, Section IV provides the details of a framework in making for a enforceable SLAs amongst cloud brokers, cloud consumers and cloud providers for efficient service provisioning. Sections V and VI conclude current findings and highlights future work directions.

II. LITERATURE SURVEY

Though Cloud computing is a highly studied topic today and a large body of research has gone into studying specific standards of interoperability amongst clouds and how they are to be achieved, the aspects of brokering services to the end client from amongst those available is finding refereed status only recently [4]. A cloud broker has been described as an entity that manages the use, performance and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers.

Existing work in literature primarily stress on using SLAs to guarantee consumer of cloud services a level of performance, that is defined by abstract metrics, directly from the cloud service providers to the end client or cloud consumers [9], [10], [11]. There is an apparent void in research on SLA formulation strategies between the cloud service broker and the cloud consumer and between the cloud service broker and the cloud service provider. This research is an attempt to highlight the research void and recommend a framework which can be developed for creation of enforceable and implementable SLAs in the cloud paradigm.

The architecture of the cloud, whether public, private, community or hybrid, would make it non trivial to propose and implement a framework for creating of such binding frameworks in the absence of accurate measuring and monitoring mechanisms for provision of services. This is especially true for a use case when the broker is aggregating and arbitraging services from multiple cloud service providers and packaging them as a service bundle for the end client. Previous work on the subject include [5], [6] and [7] that pertains to SLA formulation, but does not address the aspects of the cloud brokering actor's role in the provisioning of services. Alhamad [9] [10] discusses the aspect of SLA and performance measurement in his recent findings but does not address the issue in the perspective of how a broker would become a party to the SLA agreement between the end user or the cloud consumer and the cloud service provider. In [25], Alhamad describes a conceptual framework for SLA in the cloud computing paradigm, but the same is silent on the aspect pertaining to Brokers in the service model. Other work on SLA management and creation includes [11], which describes an approach for negotiating and creating SLA between infrastructure providers and service providers. In [12], Parrilli provides a legal perspective on the aspect of SLA provisioning in the European Union and how the rules on jurisdiction provided by the Regulation 44/2001 where two general distinctions are drawn in order to determine which (European) courts are competent to adjudicate disputes arising out of a SLA. The former is between Business to Business and Business to Consumer transactions, while the latter is in regard to contracts which provide a jurisdiction clause and contracts which do not.

A recent work by Wang et al. [13] addresses the aspects of multi-variable SLA based metrics that manages resource scheduling for application provisioning on the cloud. They also recommend a reputation based system for selecting a cloud provider. In [14], Salvatore et al. discuss a framework for broker assigned SLA management service with a novel high level abstraction model has been recommended. They recommend an architectural design for a system named Cloud Agency that aims to respond to the need for Resources management and offers added value to the existing Cloud services. The proposed system is in charge of brokering the collection of Cloud resources from different providers that fulfills requirements of user's applications as a best effort service. The user is able to delegate to the Agency the necessary checks of the agreement fulfillment, the monitoring of resource utilization and eventually necessary re-negotiations. In [15], Balakrishnan and Somasundaram propose a broker framework where SLA enabled broker evaluate the number of resources available in the environment and the number of policies per resource that need to be implemented. The results presented in the paper indicate that the inclusion of SLA affects the resource selection behavior of the broker. The paper is however silent on the methods to control the affect using an SLA. It does however indicate that the overall performance of the system improves in terms of job throughput with an extra overhead in request processing due to the presence of a broker. These

 ISBN: 978-1-61208-216-5

results are shown on a grid sharing environment and major differences exist in the business model used for the grid service provisioning and cloud service provisioning model.

A number of publications, post 2010 [8], [9], [13], [23], [24], [25] are either addressing the aspect of SLA management for brokering services at the level of a resource scheduler, or abstractions of the same when lifted from the grid computing era. The industry is viewing SLA performance management and service provisioning as a combination of availability parameters and associated factors. The carry forward of concepts of web service based SLAs in literature is also evident while drafting cloud based SLAs in recent papers. However, this research on the topic indicates that the business model of provisioning of these two frameworks is very different and mapping the two under the same head would be a mistake. The same has been asserted by NIST [1]. Quantifiable system level metrics like QoS, CPU utilization, assured storage space, scale up and scale down time in terms of elasticity of service, besides some metrics of security also find mention in industry white papers when they refer to enforceable SLAs. Recent literature also highlight the abstract and non quantifiable aspects of performance management and binding of service issues by cloud service brokers while terming the environment of cloud computing turbulent [16].

The purpose of this paper is to highlight the research gap existing in SLA formulation between the cloud broker-consumers and broker-provider combine. It presents the research done thus far and the likely line of further research to address the void. The researchers believe that the solutions to finding or evolving a framework for enabling such enforceable SLAs would be a combination of adopting appropriate managerial practices by the consumers and incorporating the best available technological means for monitoring and measuring the services available in the cloud. This paper thus presents a techno-managerial perspective to the issue.

The perspective adopted in this paper is that of a cloud broker. It is directed towards a cloud consumer and a cloud provider, when seen from a cloud broker's angle. This paper does not discuss the implication of a binding SLA between the cloud broker and cloud auditors or the cloud carriers. The relationship impact on these actors from the perspective of the broker will be done as a separate study in future.

III. SLA WITHIN A SERVICE ORIENTATION MODEL

An agreement is always based on a measure of trust. Trust concepts have been defined differently when used in varying contexts. Economists, lawyers and information technologists tend to view trust in different light. Numerous models are proposed in literature that attempt to solve the problems that arise when two parties need to establish a business relationship between them. Hussain and Chang [17] highlight the confusion in literature around the concept of trust. The acceptable definition of trust in a common usage scenario is succinctly provided by Dasgupta in [18] where he defines trust as "the expectation of one person about the actions of others that affects the first person's choice, when an action must be taken before the actions of others are known." This paper considers the interaction between the cloud actors in the same context. Gambetta [19], on the other hand, states that "trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action." In the cloud paradigm this relationship maps to the level of trust that exists between the actors involved in the services provisioning.

A. *Trust as a Base for Enforcable SLAs*

SLAs are based in an inherent trust relationship. SLAs are legal and formal documents which presents the manner in which a relationship between two entities would evolve and be conducted during normal and extreme circumstances. Service providers use SLA as a foundation to optimize the use of resources available at their disposal, while ensuring that the necessary levels of service, as defined in the SLA is delivered to the consumer. The cloud consumer on the other hand uses a SLA to assure themselves of a minimum level of service, which gets enumerated in the SLA that defines the relationship. For the service industry, SLAs must be modeled around a series of related metrics which govern performance in the specific industry. More specifically, an SLA must clearly define components that govern the relationship between the players. An SLA format should illustrate the following:

- Describe a service in unambiguous terms so all stake holders understand the implication and expectations from the service.
- Present the level of performance of service in terms of metrics.
- Define a monitoring mechanism that would monitor and report if the defined service levels are being provisioned and available to the consumers.
- A mechanism for measurement of the services being provisioned. It is essential that the process is acceptable to all the players involved in the process.
- Provide a framework for imposing penalty due to diversions from the stated terms in the SLA.
- Provide a mechanism that allows the parties engaged in the SLA to interact and meet on common ground in the event of a dispute.
- Duration of implementation and validity of the SLA.

B. *SLAs in the Service Industry Framework*

The researchers believe that that for the service industry, and especially for the cloud based business model, an SLA must define adherence to some other common metrics. These metrics need customization based on the kind of services needed by the broker (arbitraged or intermediated). Some of the metrics, which can be included in formulating an effective and enforceable SLA, are presented below :-

- Response levels in terms of time for service provisioning.
- Cost of provision of service to the end user.

 ISBN: 978-1-61208-216-5

- Service problem reporting and hierarchy of ticket resolution.
- Resolution mechanisms.
- Monitoring and service reporting accountability in terms of resources responsible for the monitoring and adherence within the time frame agreed upon.
- Liabilities of the service provider in case the desired services are not delivered.
- Terms and taxonomy that is agreed upon by the consumer of services and the other actors involved in the process.
- Conditions extraneous to the agreement which have a binding bearing on the SLA.

C. Factors that Fail SLA based Relationships

It is also pertinent to appreciate factors that have been found to be primary reasons for SLA based relationships to fail at times. Industry literature indicates non-optimal business deals that fall through, do so due to ill conceived or poorly researched SLAs [3]. This research deducted that issues common in such failed SLAs based relationships include:

- Ambiguity in differentiating between results and efforts by the service provider.
- Unclear and incomplete service specifications in the SLA lead to dissimilar level of understanding between the service provider and service consumers and other actors involved in the process.
- Incorrect people in the hierarchy creating and approving the SLA.
- Lack of agreement on common taxonomy and terms of reference.
- Lack of trust after a service related issue between the cloud consumer and cloud provider.

Dinesh [20] cites the three different approaches or models used to create a binding SLA in the service industry. These are the Insurance Model, where the service provider makes its best attempt to satisfy the performance, availability and responsiveness objectives that are specified in the SLA according to its normal operating procedures, the Provisioning Approach where the service provider typically signs different types of service objectives with different customers and allocates the resources within the environment differently to each customer in order to be able to support the service level objectives for each of the individual customer, and finally the Adaptive approach where service provider would dynamically modify the configuration of the system used to support the customer when monitoring mechanisms indicates a change in requirements and a danger that the SLA might get violated. Research in this paper through interaction with the industry and the academia indicates that for the cloud based service framework, all three models would be required and some customization on the model might be used at times, based on the kind of type of services desired by the broker and the consumer.

D. Constituients of an SLA for the Service Industry

In a service oriented architecture, especially on an IP based networks which the cloud paradigm is all about, the creation of an SLA would entail incorporating several system availability, system performance and security related metrics. A tentative list is provided in the GICTF [21], for ready reference. The final aim of providing a service level management framework is to enable the players to offer a business ready service oriented architecture that enables the service economy in a quantifiable and dependable way. This is true for cloud providers, consumers and brokers alike. Thus the intended SLA governing the relationship between these actors must ensure that the following metrics are met:

- The quality characteristics of service are predictable and enforced at run time.
- The SLA management is transparent and defines the exact conditions of service delivery and can be managed across the entire IT service stack as defined in the NIST model.
- The whole process is as automated as possible to ensure that the service delivery is elastic and scalable, besides being responsive.
- The process of creating an SLA must be repeatable.

How this translates to a cloud broker-cloud consumer and cloud broker-cloud provider is the subject of the next section.

IV. SLA WITHIN THE THE CLOUD BROKER PARADIGM

As cloud computing is evolving, the provisioning and monitoring of cloud services is becoming more complex. It has been realized that the present set of services on offer are so complex that normal cloud consumers would not be able to manage and deploy them without significant assistance. In such a scenario, a cloud consumer would request cloud services from a cloud broker, instead of contacting a cloud provider directly [1]. As per NIST, a cloud broker is an entity that manages the use, performance and delivery of cloud services and negotiates relationships between cloud providers and cloud consumers.

A. SLA Formulation Issues in the Cloud

Ensuring SLA formulation in the present cloud service provisioning space is a non trivial task. Compliance to multiple local laws in the location that house the data of the cloud consumer, opacity in terms of location of the resources that are provisioned and other similar non-quantifiable metrics make the drafting, measuring and monitoring difficult. The present framework of cloud provisioning is by no means stable and the interplay between players in the cloud model is presently not able to efficiently and adequately address the needs of consumers or the brokers. There is thus a growing need for adopting SLA frameworks that not only support the service models of IaaS, PaaS and SaaS, but also provide a measuring and quantification methodology for ensuring SLA adherence. This issue finds mention in the Draft NIST Roadmap for Cloud Computing, in Section 2.3, which highlights the need for an industry wide standard SLA for provisioning of services between the cloud provider and the cloud consumer. The draft is however

 ISBN: 978-1-61208-216-5

silent on the need to formulate the SLAs between the broker and other players in the model.

As per the NIST framework, the cloud broker would provide three distinct services: Service Intermediation, Service Aggregation, and Service Arbitrage. These have been explained in detail in the ibid document and the distinction lies in the mode of provisioning of the services and what kind of value addition the broker would provide to the cloud consumer and a business value to the cloud service provider. These require specific and binding agreements between the actors for the reference model to function as intended. Adding complexity to the cloud brokering framework are the varied deployment models that exist in reality, i.e. the public, private hybrid and community deployment models. The broker would require multiple SLAs with the associated stake holders based on the deployment model and the placement of actors in the model.

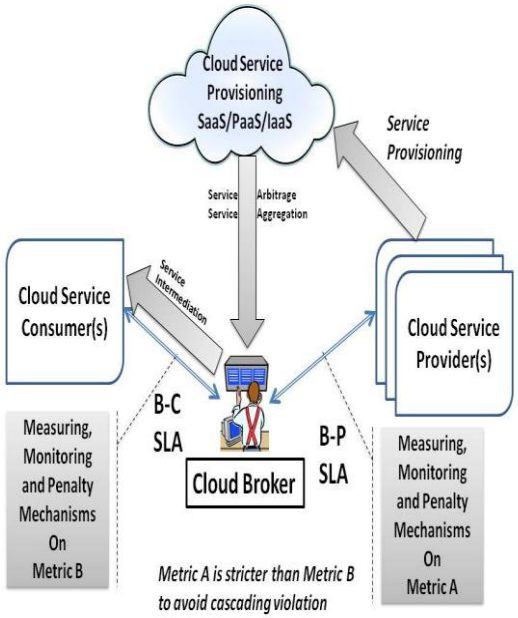


Figure 1. Framework for SLA management and Cascading Effect

B. Cloud Broker and Cloud Consumer SLA

Cloud service oriented SLAs, with the cloud broker as an actor, represent a negotiated service contract between the associated parties that specifies, in measurable terms, what cloud service will be provided to the consumer through the cloud broker. This necessitates that key elements required for cloud services including warranties, guarantees and related performance metrics are not left out of the SLA. If left out, they often tend to make the SLA unenforceable. The broker would need to make the consumer understand and appreciate the nuance of such elements and make sure that the agreement between the broker and the service provider also reflects the terms in an unambiguous manner. The aim is to make sure all parties understand and anticipate the course of action in provisioning of the service.

Research indicates that the usage of common terms and definitions within the SLAs are accepted to avoid misunderstandings between all three parties. The terms of reference need to be universally defined at the beginning of an SLA in a manner that it becomes unambiguous to the consumer, the provider and the broker as to what the service agreement entails.

C. Cloud Broker and Cloud Provider SLA

It is also necessary to create an environment which allows the broker to objectively compare competing services and offer them as bundles to the intended consumers. As the broker would be involved in service intermediation, aggregation, and arbitrage, it is necessary to have a comparative framework where the services provisioning and service usage are both compared in an objective manner. The authors are of the opinion that reputation based systems would be ideal to achieve such objectivity. Design of such systems would be a work in progress and evolve based on the stability of the cloud broker system.

SLAs that would define the relationship between cloud brokers and cloud providers would need to be based on the same lines as those between the cloud broker and the cloud consumers. There is a need for enumerating the same level of service provisioning guidelines which get mentioned in the broker-consumer SLA.

Fig. 1 illustrates the relationship between the actors involved in the service provisioning model and how enforceable SLAs would provide a systematic assessment of the services on offer based on the measuring, monitoring and penalty metrics. The figure also illustrates the effect of a failure of an SLA on the provisioning model. With the cloud broker as an entity in the NIST recommended cloud framework, it is imperative that metrics of service agreement agreed upon between the cloud broker and cloud service provider need to be more stringent than those between the broker and the cloud consumer. A failure in provisioning of the agreed upon services by the cloud service provider will have a 'Cascading Effect' on the service model. The 'cascade' will be aggravated in the cloud paradigm as multiple associations exist between the cloud broker and the service consumers (one-to-many and, at times, many-to-many). The aspect of service arbitrage by the cloud broker would thus need to be deliberated very minutely in the event of a failure of service. The researchers strongly believe that the affect of SLA failures will lead to a cascade effect in terms of service outage for multiple cloud service consumers. In April 2012, Amazon Inc., faced a major outage of host of its services [22]. Such outages reflect the effect of the cascade due to the failure of a bundle of services from one provider on multiple, sometimes more than a million consumers – which we feel is a cascade of service outages.

The researchers are also convinced based on interaction with the industry and the academia that there is a need for a reputation based system, based on the assumption of a stronger metric enforcement between the cloud broker and cloud service provider vis-à-vis the cloud broker-consumer, for arriving at a comparative framework for selecting the service bundle and defining the system level and availability based metrics in the SLA between the cloud service provider

 ISBN: 978-1-61208-216-5

and the broker. The reputation based system can be based on relevant service metrics as would be proposed for the consumer-broker SLA. Some of the metrics, which the researchers feel could be used in selecting the appropriate service bundles by the brokers could include response time in provisioning (SaaS), rate of successful delivery of promised services levels of a defined period of time (PaaS), risk preventing mechanisms in place by the provider and SLA success metrics of the provider. This also brings upon the aspect of measuring mechanisms which need to be in place while drafting the SLA. This is especially true while drafting the provider–broker SLAs as it is anticipated that aggregation of multiple, differing services is the way ahead and cloud brokers would need to have a mean to measure the service been hired. This is also illustrated in Figure 1 above. This research illustrates that the violation of service agreements between the broker and provider has a consequent affect on the agreements between the broker and the consumer and this can lead to a cascading degradation in service provisioning, if not checked in time through effective monitoring mechanisms. SLA drafting and management by incorporating effective monitoring and measuring mechanisms is thus an essential task in ensuring better cloud services provisioning. The metrics recommended in this research for basing a SLA between different actors in the cloud framework would need further study and the researchers also believe based on the work thus far that these metrics would change based on the service bundle desired by the consumer and arbitraged by the broker.

V. CONCLUSION

Creating an effective trust relationship between the cloud brokers and cloud consumers is essential to maintain the desired level of service provisioning in the cloud. This trust is enforced using effective agreements between actors. This trust is often realized when agreements are based on clearly defined and effectively executed contract agreements, or SLAs, which are a corner stone for provision of well executed, responsive and elastic services in the cloud. The aspect of SLA management between cloud brokers and cloud providers as well as between cloud brokers and consumers is a research void at present and has been highlighted in this paper. The SLAs between the three actors have a bearing on each other. Industry reports coupled with the research done on the subject indicate that the cloud broker's role in the framework for cloud service provisioning is increasing and thus the relationship between these individual SLA assume increased importance. It has also been realized through this research that a strong contractual SLA between the cloud broker and the cloud service provider is necessary for the cloud framework to maintain its stability. The research also highlights the affects of failure of the agreed upon services illustrated in a SLA between the broker and the provider and the consequent service outage which ensues. The researchers have termed this as 'Cascading Effect' in the cloud service model. The utility of a well defined and enforceable SLA based on quantifiable metrics with the broker as a central actor is thus of paramount importance. The researchers also believe that there is a need for a reputation based, comparative system for arbitraging services from different service providers. Such a system can be used by brokers for selecting the bundle of service more efficiently and the design of such a reputation based system is a work in progress.

VI. FUTURE WORK

This paper is a work in continuation as part of a doctoral thesis on cloud computing and affects on managerial aspects of an organization when working in a cloud paradigm. As a future work the authors are examining the NIST framework and exploring how measurable metrics can be defined to create a universally acceptable interoperability framework required for dissimilar clouds to talk to each other. The authors are also of the opinion that there is a need to further work on drafting comprehensive and binding SLA templates that address the lacunae existing in service provisioning between the three actors. There is also a need to further understand the cascading affect due to terms of service violation when seen from the perspective of a cloud broker. Monitoring and measuring frameworks also form an essential part of the SLA management process and are a topic for future research. Another work in future could be the affect of these SLAs on the cloud auditors and cloud carriers when viewed from the perspective of a cloud broker.

REFERENCES

[1] NIST, "Draft Cloud Computing Technology Roadmap" NIST Special Publication 500-293.

[2] K. E. Cheng, Y. M. Gottlieb, G. M. Levin, and Fuchun Joe Lin, "Service brokering and mediation: Enabling next generation market and customer driven service delivery," Proc. Tenth International Symposium on Autonomous Decentralized Systems (ISADS '11). IEEE Press, Mar. 2011, pp. 525-530, doi: 10.1109/ISADS.2011.100.

[3] L. Herbert and J. Erickson, "The ROI of cloud apps" in A Total Economic Impact™ Analysis Uncovers Long-Term Value In Cloud Apps, Forrester, 2011.

[4] C. A. Yfoulis and A. Gounaris, "Honoring SLAs on cloud computing services: a control perspective", Proc. 2nd Workshop on Bio-inspired Algorithms For Distributed Systems, Jan. 2010, pp. 29-38.

[5] D.D. Lamanna, J. Skene, and W. Emmerich, "Slang: A language for defining service level agreements," Proc. Nnth IEEE workshop of Future Trends of Computing 2003, IEEE Press, May 2003, pp. 100-106, doi:10.1109/FTDCS.2003.1204317.

[6] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck, "Web service level agreement (WSLA) language specification," IBM System Journal, vol. 43, Jan. 2004, pp. 136-158, doi:10.1147/sj.431.0136.

[7] A. Paschke, "Rbsla - A declarative rule-based service level agreement language based on rulem," Proc. International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06), IEEE Computer Society Dec. 2005, vol. 2, pp. 308-314, 28-30, doi:10.1109/CIMCA.2005.1631486.

[8] H. Boley, S. Tabet, and G. Wagner, "Design rationale of ruleml: A markup language for semantic web rules" 2001, pp. 380-401 [retrieved: May, 2012].

 ISBN: 978-1-61208-216-5

[9] M. Alhamad, T. Dillon, and E. Chang, “SLA-Based Trust Model for Cloud Computing,” Proc. 13th International Conference on Network-Based Information Systems (NBIS '10). IEEE Computer Society, Dec. 2010, pp. 321-324. doi:10.1109/NBiS.2010.67

[10] M. Alhamad, T. Dillon, and E. Chang, “A survey on SLA and performance measurement in cloud computing,” Proc. Confederated International Conference on On the Move to Meaningful Internet Systems - Vol II(OTM'11), Springer-Verlag, Dec. 2011, pp. 469-477.

[11] A. Lawrence, K. Djemame, O. Wäldrich, W. Ziegler, C. Zsigri, “Using service level agreements for optimising cloud infrastructure services,” Proc. International Conference on Towards a Service-based Internet (ServiceWave'10), Springer-Verlag Berlin, 2011, pp. 38-49.

[12] D. M. Parrilli, “The determination of jurisdiction in grid and cloud service level agreements,” Proc. 6th International Workshop on Grid Economics and Business Models (GECON '09), Springer-Verlag, 2009, pp. 128-139, doi:10.1007/978-3-642-03864-8_10.

[13] M Wang, X Wu, W. Zhang, F. Ding, J. Zhou, and G. Pei , "A Conceptual Platform of SLA in Cloud Computing," Proc. IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC), 2011, Dec 2011, pp.1131-1135, 12-14, doi: 10.1109/DASC.2011.184.

[14] S. Venticinque, R. Aversa, B. Martino, M. Rak, and Dana Petcu, “A cloud agency for SLA negotiation and management,” Proc. Conference on parallel processing (Euro-Par 2010), Springer-Verlag, Aug. 2010, pp.587-594.

[15] P. Balakrishnan and T. S. Somasundaram, “SLA enabled CARE resource broker,” Proc. Future Gener. Comput. Syst, vol. 23, Mar 2011, pp. 265-279, doi:10.1016/j.future.2010.09.006.

[16] V. C. Emeakaroha, I. Brandic, M. Maurer, I. Breskovic, “SLA-Aware Application Deployment and Resource Allocation in Clouds,” Proc. Computer Software and Applications Conference Workshops (COMPSACW 2011), IEEE Press, July 2011, pp. 298-303, doi:10.1109/COMPSACW.2011.97.

[17] F. K. Hussain and E. Chang, "An overview of the interpretations of trust and reputation", Proc. The Third Advanced International on Telecommunications (AICT 2007), May 2007, pp. 30-30. doi:10.1109/AICT.2007.11.

[18] A. Dasgupta and A. Prat, "Reputation and asset prices: A theory of information cascades and systematic mispricing”, Manuscript, London School of Economics, Sep. 2005.

[19] D. Gambetta, “Trust: Making and breaking cooperative relations,” Basil Blackwell, New York, 1990.

[20] D. C. Verma, M. Beigi, R. Jennings, “Policy Based SLA Management in Enterprise Networks,” Proc. International Workshop on Policies for Distributed Systems and Networks, Jan. 2001, pp. 137-152.

[21] GICTF, “Use case and funtional requirements for inter cloud computing,” Aug 2010, url: http://events.oasis-open.org/home/sites/events.oasis-open.org.home/files/20111012-ICS-goto-GICTF.pdf , [retrieved : May 2012].

[22] Rick Vanover, “What the recent Amazon Web Service Mean in our own Cloud Journey,” url: http://www.techrepublic.com/blog/networking/what-the-recent-amazon-web-services-outages-mean-in-our-own-cloud-journey/3910?tag=content;siu-container [retrieved : May 2012].

[23] R. N. Calheiros, R. Ranjan, A. Beloglazov, D. Rose, A. F. César, and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” Journal of Software Practice and Experience, vol 41, Aug 2010, pp. 23-50, DOI: 10.1002/spe.995.

[24] M. Macías, J. O. Fitó, and J. Guitart, "Rule-based SLA management for revenue maximisation in Cloud Computing Markets," International Conference on Network and Service Management (CNSM), Oct. 2010, pp. 354-357, doi:10.1109/CNSM.2010.5691226.

[25] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA framework for cloud computing," 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST, Apr. 2010, pp. 606-610, doi: 10.1109/DEST.2010.5610586.

 ISBN: 978-1-61208-216-5

Towards a Domain-Specific Language to Deploy Applications in the Clouds

Eirik Brandtzæg
University of Oslo
SINTEF IKT
Oslo, Norway
eirik.brandtzaeg@sintef.no

Parastoo Mohagheghi
NTNU
SINTEF IKT
Trondheim, Norway
parastoo@idi.ntnu.no

Sébastien Mosser
Networked Systems and Services
SINTEF IKT
Oslo, Norway
sebastien.mosser@sintef.no

***Abstract*—The cloud-computing paradigm advocates the use of virtualised resources, available "in the clouds". Applications are now developed in order to be cloud-aware. Unfortunately, the deployment of such applications is still manually done, or relies on home-made shell script. In this paper, we propose to model cloud applications using a component-based approach. It leverages the existing deployment descriptors into a high-level domain-specific language. The language is then illustrated through the modeling of a prototypical application used to teach distributed programming at the University of Oslo.**

Keywords-Cloud-computing; Modeling; Deployment.

I. INTRODUCTION

Cloud Computing [1] was considered as a *revolution*. Taking its root in distributed systems design, this paradigm advocates the share of distributed computing resources designated as "*the cloud*". The main advantage of using a cloud-based infrastructure is the associated scalability property (*e.g.*, *elasticity*). Since a cloud works on a *pay-as-you-go* basis, companies can rent computing resources in an elastic way. A typical example is to temporary increase the server-side capacity of an e-commerce website to avoid service breakdowns during a load peak (*e.g.*, Christmas period). However, there is still a huge gap between the commercial point of view and the technical reality that one has to face in front of "*the cloud*".

A company that wants to deploy its own systems to the cloud (*i.e.*, be part of the cloud *revolution*) has to cope with existing standards. The Cloud-Standard wiki [2] lists dozens of overlapping standards related to Cloud Computing. They focus on infrastructure modeling or business modeling. These standards do not provide any support for software modeling or deployment. Thus, the *deployment* of a cloud-based system is a difficult task, as it relies on handcrafted scripts. It is not possible to reason on the deployment, nor to assess it with respect to (*w.r.t.*) to cloud business policies.

The Cloud-computing paradigm emphasizes the need for automated deployment mechanisms, abstracted from the underlying technical layer. As cloud-computing considers that the number of resources available in the cloud is not limited, it triggers new challenges from a deployment point of view. Even if several approaches consider the deployment target as "open" (*i.e.*, new host machines can be added in the environment), the "virtually unlimited" dimension provided by the cloud-approach is not taken into account.

Our contribution in this paper is to propose a component-based approach [3] to model software deployment in the clouds. This approach is provided as a *Domain-Specific Language* (DSL), which is given to the software designer. The language is based on a reduced component meat-model, and support the modeling of the deployment relationship between components. For the sake of concision, we only focus in here on the description of the cloud deployment language usage, and we do not address in this paper the run-time enactment. This work is done in the framework of REMICS [4], an European project dedicated to the migration of legacy application into cloud-based applications. Section II discusses related works, and Section III illustrates the challenges on a running example. Section IV describes the language meta-model, Section V describes its usage, and finally, Section VI concludes this paper.

II. RELATED WORKS

We propose here to analyze the state of the art about software deployment, identifying good practices to be reused in our own solution, dedicated to cloud-computing. The cloud model always assume that the software to be deployed will be running on an host machine, virtualised in the cloud. Thus, its deployment depends on a lot of characteristics provided by the host, *e.g.*, IP address, operating system, available remote protocols. The deployment might also depends on the software to be deployed, *e.g.*, implementation language, configuration capabilities.

A. Deployment Models

Several approaches were proposed to abstract the user from the underlying platform *w.r.t.* the deployment point of view. These approaches propose to model the deployment of a software in a generic way, using the concepts described in a meta-model. In this domain, the two main approaches are *(i)* the UML Deployment Diagrams [5] and *(ii)* the OMG D&C meta-model [6]. These approaches are complemented by academic approaches like ORYA [7] and GADE [8].

 ISBN: 978-1-61208-216-5

UML Deployment Diagrams: using the UML Deployment Diagram approach, one can use *artifacts* to model the physical elements involved in the deployment (*e.g.*, a compiled executable to be copied on the host machine). Artifacts follows a composite pattern (*i.e.*, an artifact can be composed by others), and are expressive enough to model complex software dependencies. These elements are bound to physical *devices* to model which software artifact must be deployed on which machine. The infrastructure is modeled thanks to the definition of communication path between different devices.

OMG D&C meta-model: D&C means "*Deployment and Configuration*". It was built to tackle the challenges encountered while standardizing the deployment of CORBA components. This meta-model defines *(i)* meta-data to be used during the deployment process (*e.g.*, configuration information for a given package) and *(ii)* a target model relying on these meta-data to describe the deployment process. The approach is extremely verbose, and suffers from the number of concepts to be used to model a deployment, even in front of a simple case. Another weaknesses is its close relationship with CORBA: the meta-model is too close to the one defined by CORBA, and existing work based on OMG D&C focus on the deployment of CORBA components [9], [10].

Academic approaches: We consider here two prototypical examples. ORYA is similar to the UML deployment diagram approach, as it provides a purely descriptive meta-model to describe a deployed system. ORYA also provides concepts to model administrative and legal issues in the deployed system. But it suffers from the same drawback, *i.e.*, its lack of a clear semantics (or a least a reference implementation) to properly support the deployment in an automated and reproducible way. GADE is the complete opposite, as it concretely targets the deployment of software components in grid-computing environment. It focus on the capture of the grid domain, supporting the user in the deployment of processes to be executed on the grid. This approach emphasizes the need for a deep understanding of the domain while modeling a deployment meta-model.

B. State of Practice: Cloud-based solution

Cloud providers have already understood that deployment is crucial while talking about clouds. Thus, they provide mechanisms to support the user during the deployment of applications. This support can be textual (*e.g.*, Amazon Cloud Formation [11]), graphical (*e.g.*, Applogic [12]). But it immediately suffers from the "vendor lock-in" syndrome. Thus, several libraries can be found (*e.g.*, libcloud [13], jclouds [14], δ-cloud [15]) to abstract these providers.

Amazon Cloud Formation: it is a service provided by Amazon from their popular *Amazon Web Services* (AWS). It give users the ability to create template files, which they can load into AWS to create stacks of resources. This is mainly meant for users that want to replicate a certain stack, with the ability to provide custom parameters. Once a stack is deployed it is only maintainable through the AWS Console, and not through template files. The structure and semantics of the template itself is not used by any other providers or cloud management tooling, so it can not be considered a multi-cloud solution and enforce a vendor lock-in syndrome.

Applogic: it is a proprietary model-based application for management of private clouds. This interface let users configure their deployments through a diagram with familiarities to component diagrams with interfaces and assembly connectors. They also provide an *Architecture Deployment Language* (ADL) to enforce properties on the modeled deployment. But this solution is only made for private clouds running their own controller, this can prove troublesome for migration, both in to and out of the infrastructure.

Application Programming Interface (API): Libcloud and *jcloud* are APIs that aim to support the largest cloud providers through a common API. Libcloud have solved the multi-cloud problem in a very detailed manner, but the complexity is therefore even larger. The *δ-cloud* approach has a similar procedure as *jclouds* and *Libcloud*, but with a web-service approach (introducing a bottleneck).

C. Conclusions

The deployment models available in the state of the art demonstrate that a descriptive modeling of deployment is elegant and well understood by the end user. Such an approach must stay simple and focused, to avoid the multiplication of concepts. The approach must also be tailored to address its target domain, *i.e.*, cloud-computing in our case. The available tools analyzed from the state of practice demonstrate that the heterogeneity of the different underlying platforms needs to be abstracted. Anyhow, the current approaches are available at the code level, and does not provide an abstraction layer to be used by the application designer to properly model a cloud-based application to be deployed.

III. Running Example & Challenges

We consider here a simple application, sufficient to underline the intrinsic complexity of cloud–application deployment modeling. This application is called **`BankManager`**, and is used at the University of Oslo to teach distributed systems, based on the very classical *"bank account management"* case study. It consists of the two following parts:

- A back-end that contains a **`Database`**, used to store information about customers and accounts,
- A front-end that implements a web-based application, used to access to the different accounts and transfer money between accounts.

From a software architecture point of view, this application simply consists of a relational database to support the back-end, and java-based *servlets* bundled in a WAR archive to support the front-end. The front-end must hold a reference to the back-end to address the proper database. But when

 ISBN: 978-1-61208-216-5

confronted to the "cloud-computing" domain, the following points needs to be also considered:

- Clouds implement open environments. As a consequence, we do not know where the application will be deployed. Thus, establishing the link between the front–end and the back–end requires a particular attention.
- Clouds provides different mechanisms to support application deployment. Where infrastructure cloud (IaaS) mainly provides low-level (*e.g.*, SSH, FTP) connectivity to the virtual machines, platform clouds (PaaS) provides deployment protocols dedicated to the technology they implement (*e.g.*, WAR deployment).
- Clouds work on a pay-as-you-go basis. Thus, one can consider to deploy both back-end and front-end artifacts on the same virtual machine, to reduce costs during development. Another alternative is to deploy these two artifacts on two different virtual machines. In concrete case, the variability of deployment possibilities is humongous.
- Clouds emphasizes reproducibility. Thus, a given deployment descriptor should be easily re-usable as–is, in the same context or in a new one.
- Clouds support scalability through replication and load-balancing. The deployment descriptor should be easily replicable to support the on-demand replication of computation-intensive artifacts.

Our goal is to provide a meta-model that supports the application designer while deploying a cloud application.

IV. A DSL TO SUPPORT DEPLOYMENT IN CLOUDS

We named the language *Pim4Cloud DSL*, as it is a *Platform Independent Model* dedicated to *Clouds*. The key idea of the Pim4Cloud DSL is to support the deployment of application in the cloud. An overview of the approach is depicted in Figure 1. Using the DSL, the application designer models the software to be deployed. In parallel, the infrastructure provider describes the available infrastructure to be used by the application. From a coarse-grained point of view, it means that the designer requires "computation nodes" (*e.g.*, virtual machines) from the cloud, and the infrastructure provider describes such nodes (based on its own catalog). An interpreter is then used to identify which resources have to be used in the infrastructure to fulfill the requirements expressed by the application designer. The interpreter then do the provisioning, and actually deploys the modeled application. It returns as feedback to the designer a *living model* of its application, annotated with run-time property bound to each modeled artifact (*e.g.*, the public IP address associated to a given virtual machine).

Based on the points previously described, we propose to use a component-based approach to model the deployment of cloud applications. This approach was successfully used by the DEPLOYWARE framework in the context of adaptive

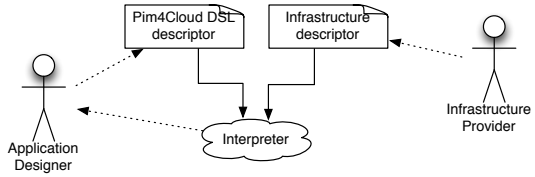


Figure 1. Pim4Cloud DSL overview

component system [16], [17], and we propose to transpose its core idea to cloud deployment.

To achieve this goal, we use a reduced component meta-model, described in Figure 2. This meta-model is expressive enough to support the modeling of both infrastructure and applicative artifacts in an endogenous way. **`Component`**s can be scalars or composite, *i.e.*, containing sub-component inside their boundaries. A **`Component`** may offer one or more deployment **`Service`**s, *i.e.*, deployment protocols one can used to deploy other components onto this one (*e.g.*, a *servlet* container will offer a **`WAR`** service to support the deployment of java-based web applications). Obviously, it may require one **`Service`** if it aims to be deployed on another one (*e.g.*, a WAR artifact will require a **`WAR`** service). **`Components`** are connected among others through **`Connectors`**. A component can offer and expects **`Property`**, *e.g.*, a database component may expect both **`username`** and **`password`**, and provide an **`url`** to be remotely accessed. These elements are used at run-time (asked in a deployment descriptor, or filled using the feed-back obtained from the underlying cloud infrastructure). In a **`Composite`**, one can express bindings between properties, that is, a formal link between an expected and an offered property. These links (**`RuntimeBinding`**) are used at run-time to properly transfer the expected information.

Implementation: This meta-model is intended to be specialized according to user's needs, as its intrinsic simplicity makes it easy to introduce in user's code. We provide a reference implementation of this approach using the Scala language, exposed as an internal domain-specific language to support the usage of this meta-model in JVM-based

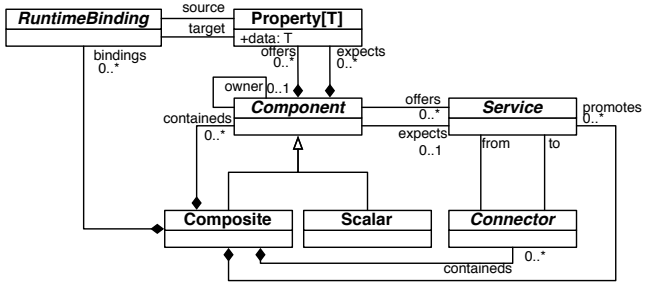


Figure 2. Modeling cloud components: a generic meta-model

 ISBN: 978-1-61208-216-5

languages. The DSL is designed in a modular way, and implements several constructions (*e.g.*, "offering a service", "containing a component") as independent modules, implemented as *traits*. This design support the evolution of the DSL, as adding a new syntactic construct is assimilated as the mix of a new *trait*.

V. USING THE LANGUAGE

Based on this internal DSL, one can model a cloud-based software to be deployed.

A. Modeling a Simple Component

We represent in Figure 3 a graphical representation of a **WarContainer** model, using standard graphical notation for component assemblies. This container is used to host **WAR**-based artifacts. It is made as the composition of *(i)* a virtual machine obtained from a IaaS provider and *(ii)* a *Jetty* server used to actually support the hosting of **WAR** artifacts:

- The virtual machine is modeled as a component named **vm**, typed as a **SmallVM**. This component does not require any other, and is therefore considered in the models as an element obtained from an external provider (outside of the scope of the modeled system). It offers a **ssh** service, and one can use this protocol to interact with the component at run-time. This can be considered as the *IaaS* layer of this example
- The **WAR** hosting artifact is modeled as a component named **container**, typed as a **Jetty** server. It offers a **war** service, and one can use it to deploy WAR-based application. This component relies on the **APT** package system to be properly deployed. Replacing the hosting server (*e.g.*, from **Jetty** to **Tomcat**) only means to replace this component by another one.
- The final component (**WarContainer**) composes the ones previously described as the following: it *(i)* promotes the **war** port of the **container** component, and *(ii)* binds the **apt** requirement of the **container** to the **ssh** offering provided by the **vm** one.

As the language relies on Scala, the declaration of a scalar component is assimilated to the declaration of a class, that extends the concepts previously described. Thus, the user is completely free in such a class to write all the code he/she thinks necessary. The DSL is only used to support the user when dealing with its system from a deployment point of view. Internal DSLs immediately benefit from the mechanisms of the hosting language, *e.g.*, variable visibility and scoping mechanisms. We describe in Listing 1 the code necessary to model this system with the DSL.

```
class WarContainer
    extends CompositeComponent with WarOffering {
  private[this] val container = instantiates[Jetty]
  private[this] val vm = instantiates[SmallVM]
  override val war = promotes(container.war)
  this deploys container.apt on vm.ssh
}
```

Listing 1. **WarContainer** code

The **WarComponent** class extends the **CompositeComponent** concept (it is able to contains other components), and mixes the **WarOffering** trait (statically informing other components that it offers a **war** port). It instantiates two internal sub-components: *(i)* a **Jetty** component named **container** to host the servlets applications and *(ii)* a virtual machine of type **SmallVM**. It promotes the **war** service offered by the **container** sub-component, and finally deploys the servlet container on the virtual machine.

B. Multiple Topologies for *BankManager*

Based on the previously described mechanisms, we can now model several version of our initial example, the *BankManager*. This software is implemented in Java, and requires the two following elements: *(i)* a database for its back-end and *(ii)* a web server able to host WAR-based software. We represent in Figure 4 different deployment configuration for such a system.

- Figure 4(a). In this version, the front-end and the back-end are deployed on the same virtual machine. This is

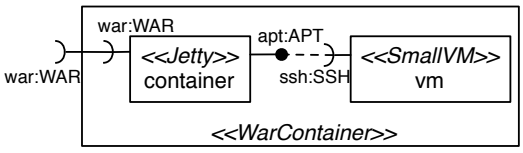
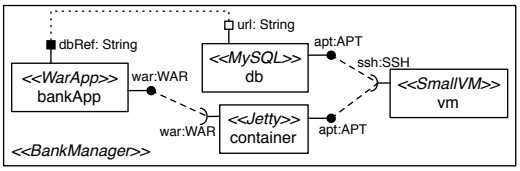
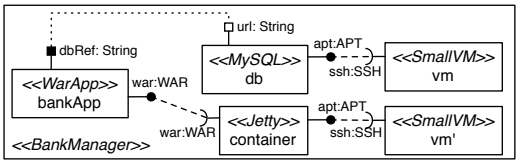


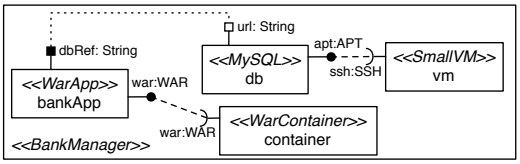
Figure 3. **WarContainer**: component diagram representation



(a) **BankManager**, virtual machine sharing



(b) **BankManager**, independent virtual machines



(c) **BankManager**, re-using **WarContainer**

Figure 4. **BankManager** deployment variability

 ISBN: 978-1-61208-216-5

typical for test purpose, where the idea is to minimize the cost of the rented infrastructure during development. The database component exposes a property named **url**. This property will be filled at run-time by the deployment engine associated to the Pim4Cloud DSL (out of the scope of this paper). The **bankApp** component expects a property named **dbRef**, and a binding is expressed at the composite level to specify that this property will be set based on the value obtained from **db** at run-time.

- Figure 4(b). In this version, two virtual machines are used. This is the main difference when compared to the previous one. This separation allows the replication of the **container** component, ensuring elasticity through horizontal scalability.
- Figure 4(c). This versions demonstrates the strength of the component approach when applied to this domain. It is immediately possible to re-use the previously described **WarContainer**. As a component is considered as a black-box, the end-user will not care about hos it works internally from an infrastructure point of view. It will simply re-use a given component that provides the needed deployment services.

We give in Listing 2 the DSL code that models these different topologies. First, we define our application (**MyCloudApp**) as an abstract class: it factorizes shared elements, and each concrete topology will extends this class to refine its content. The top-level class instantiate a **BankManager** component (the WAR file that contains the application), as well as a **MySQL** database. It defines an abstract **container**, with the assumption that this subcomponent will offer WAR deployment (it is typed as **WarOffering**). The bank manager application is then deployed on this container. The database property required by the application is filled with the url provided by the database.

We then present in Listing 3 the three different components that actually implements such deployment topologies. The first one (**VirtualMachineSharing**) instantiates a single virtual machine and deploys both the container and the database on it. The second component (**IndependentVirtualMachine**) deploys the servlet container and the database on different virtual machines (**vm1** and **vm2**). Finally, the last component (**UsingWarContainer**) reuse the

```
abstract class MyCloudApp extends CompositeComponent {
  private[this] val bankApp = instantiates[BankManager]
  protected val db = instantiates[MySQL]
  protected val container: WarOffering
  this deploys bankApp.war on container.war
  this sets bankApp.dbRef using db.url
}
```

Listing 2. **BankManager**: Abstract class to model **MyCloudApp**

```
class VirtualMachineSharing extends MyCloudApp {
  override val container = instantiates[Jetty]
  private[this] val vm = instantiates[SmallVM]
  this deploys container.apt on vm.ssh
  this deploys db.apt on vm.ssh
}

class IndependentVirtualMachine extends MyCloudApp {
  override val container = instantiates[Jetty]
  private[this] val vm1 = instantiates[SmallVM]
  private[this] val vm2 = instantiates[SmallVM]
  this deploys container.apt on vm1.ssh
  this deploys db.apt on vm2.ssh
}

class UsingWarContainer extends MyCloudApp {
  override val container = instantiates[WarContainer]
  private[this] val vm = instantiates[SmallVM]
  this deploys db.apt on vm.ssh
}
```

Listing 3. Multiple deployment topologies for **BankManager**

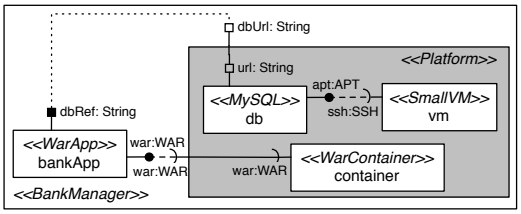


Figure 5. Deploying the **BankManager** on a *PaaS*

WarContainer component defined in Listing 1 to host the servlet container.

C. Modeling Platform as a Service Artifacts

The DSL allows us to model in an endogenous way *IaaS* and *PaaS*. Building a *PaaS* becomes as simple as modeling a software stack on top of virtual machines (Figure 5). In this case, we modeled a **Platform**, which exposes a **war** port for service hosting and a **dbUrl** property for persistence. This *platform* is then used to deploy the bank application, but can also be used to host any application implemented as a **War** and requiring a database.

From a DSL point of view, one can imagine a library of available platforms. In Listing 4, we describe a platform named **AGivenPlatform**, provided by **AGivenProvider** (modeled as a package). Then, one can use this platform by simply importing it in its component, and using it like any other. The **UsingPaaS** component in Listing 4 shows how it can be done with the DSL.

```
package AGivenProvider {
  class AGivenPlatform extends CompositeComponent with
       WarOffering {
    private[this] val db = instantiates[MySQL]
    private[this] val vm = instantiates[SmallVM]
    private[this] val container =
         instantiates[WarContainer]
    override val war = promotes(container.war)
    val dbUrl = externalize(db.url)
    this deploys db.apt on vm.ssh
  }
}
```

```
class UsingPaaS extends CompositeComponent {
  import AGivenProvider.AGivenPlatform
  private[this] val bankApp = instantiates[BankApp]
  private[this] val platform =
       instantiates[AGivenPlatform]
  this deploys bankApp.war on platform.war
  this sets bankApp.dbRef using platform.dbUrl
}
```

Listing 4. Modeling a *Platform as a Service* using Pim4Cloud DSL

VI. Conclusion & Perspectives

We described how the Pim4Cloud DSL can be used to support the application designer while modeling an application to be deployed in the clouds. We also describe how the DSL is implemented, using Scala as a hosting language. We showed on a prototypical example how the DSL is used to properly model the deployment. Application deployment can be modeled in an agnostic way *w.r.t.* the targeted cloud provider. The approach support the definition of static analysis (*e.g.*, type consistency), as well as the reuse of components from a deployment to another one (*i.e.*, architectural patterns can be reified as cloud components). This approach also support the endogenous modeling of both Paas and Iaas.

This work is currently pursued, including a standardization effort at the OMG in the context of the Remics project. Short terms perspectives of this work includes the two following axis: *(i)* "models@run.time" and *(ii)* verification. The feed-back returned to the user is for now reduced to its minimum, that is, the IP of virtual machines provisioned in the cloud. With regard to the large amount of data available from cloud providers (*e.g.*, load average, cost), one of our objective to enhance this feed-back to take into account more information. We plan to achieve this goal with a *"Models@run.time"* approach. Instead of returning a set of IP addresses, the Pim4Cloud interpreter will return a model of the running system, available at run-time. It will maintain the link between the running system and the models, providing a model-driven way of querying the cloud–based application (*e.g.*, about its status, its load). From the verification point of view, the current mechanisms included in the DSL are static for now, and intensively rely on the type system: the engine assumes that a static model (*i.e.*, a model that can be compiled) will always be properly deployed in the cloud. We plan to use a transactional approach coupled to the action-based mechanism previously described. Thanks to the *acidity* of the transactional model, the action interpreter will be able to recover when an error will be encountered during the deployment process.

Acknowledgments

This work is partially funded by the EU Commission through the Remics project (FP7-ICT, Call 7, contract number 257793, www.remics.eu)

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[2] CloudStandard, "The Cloud Standards Coordination Wiki," http://cloud-standards.org/, [retrieved: 05, 2012].

[3] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] REMICS, http://remics.eu/, [retrieved: 05, 2012].

[5] O. Object Management Group, "UML 2.0 Superstructure Spec." Object Management Group, Tech. Rep., Aug. 2005.

[6] ——, "Deployment and Conf. of Component-based Distributed App. Spec., Version 4.0," Tech. Rep., Apr. 2006.

[7] P.-Y. Cunin, V. Lestideau, and N. Merle, "ORYA: A Strategy Oriented Deployment Framework," in *Component Deployment*, ser. Lecture Notes in Computer Science, A. Dearle and S. Eisenbach, Eds., vol. 3798. Springer, 2005, pp. 177–180.

[8] S. Lacour, C. Pérez, and T. Priol, "Generic Application Description Model: Toward Automatic Deployment of Applications on Computational Grids," in *GRID*. IEEE, 2005, pp. 284–287.

[9] G. Deng, D. C. Schmidt, and A. S. Gokhale, "CaDAnCE: A Criticality-Aware Deployment and Configuration Engine," in *ISORC*. IEEE Computer Society, 2008, pp. 317–321.

[10] J. Dubus and P. Merle, "Applying OMG D&C Specification and ECA Rules for Autonomous Distributed Component-Based Systems," in *MoDELS Workshops*, ser. Lecture Notes in Computer Science, T. Kühne, Ed., vol. 4364. Springer, 2006, pp. 242–251.

[11] A. AWS, "Amazon Cloud Formation Language," http://aws.amazon.com/en/cloudformation/, [retrieved: 05, 2012].

[12] CA, "Applogic CA," http://www.ca.com/us/products/detail/CA-AppLogic.aspx, [retrieved: 05, 2012].

[13] Apache, "Apache Libcloud, a Unified Interface to the Cloud," http://libcloud.apache.org, [retrieved: 05, 2012].

[14] JClouds, "JClouds," http://www.jclouds.org/, [retrieved: 05, 2012].

[15] Apache, "δ-cloud: Many Clouds. One API. No problems." http://deltacloud.apache.org/, [retrieved: 05, 2012].

[16] A. Flissi, J. Dubus, N. Dolet, and P. Merle, "Deploying on the Grid with DeployWare," in *CCGRID*. IEEE Computer Society, 2008, pp. 177–184.

[17] J. Dubus, "Une démarche orientée modèle pour le déploiement de systèmes en environement ouverts distribués," Ph.D. dissertation, Université des Sciences et Technologies de Lille, Lille, France, Oct. 2008.

 ISBN: 978-1-61208-216-5

Cloud-based Healthcare:
Towards a SLA Compliant Network Aware Solution for Medical Image Processing

Shane Hallett[1], Gerard Parr[1], Sally McClean[1], Aaron McConnell[1], Basim Majeed[2]
India-UK Advanced Technology Centre (IU-ATC) of Excellence in Next Generation Networks, Systems and Services
School of Computing and Information Engineering, University of Ulster[1], Coleraine, UK,
hallett-s@email.ulster.ac.uk, {gp.parr, si.mcclean, a.mcconnell}@ulster.ac.uk,
ETISALAT BT Innovation Center[2], Abu Dhabi, UAE, basim.majeed@bt.com

***Abstract*—Medical image processing in the Cloud can involve moving large data sets and/or applications across the network infrastructure. With the aim of minimizing the total processing time, the optimal placement of image data and processing algorithms on a large scale, distributed Cloud infrastructure is a challenging task. This work presents a genetic algorithm-based approach for data and application (virtual machine) placement using hypervisor and network metrics to avoid service level agreement violations. The solution involves placing medical image data and associated processing algorithms at optimized processing and compute nodes located within the Cloud. The results of initial experiments show that a genetic algorithm-based placement approach can increase Cloud-based application performance.**

Keywords-cloud computing; virtual machine placement; genetic algorithm; network awareness.

I. INTRODUCTION

The rapid growth in the use of Electronic Health Records (EHR) across the globe along with the rich mix of multi-media held within an EHR combined with the increasing level of detail due to advances in diagnostic medical imaging means increasing amounts of data can be stored for each patient [1][2]. In a scenario where a consultant may view and process medical images remotely for the purpose of producing a diagnosis it may be necessary to move large data sets across the network for processing to take place [3]. Moving such data sets has the potential to introduce undesirable latency and also degrade application performance to an unacceptable level, causing service level agreement (SLA) violations and degrading network performance for other users of the same infrastructure.

Cloud Computing has come to the fore as a new model of computing service delivery as a utility over the Internet. Virtualisation technology [4] lying at the heart of the Cloud allows greater utilisation of physical and virtual resources. Depending on the resources available physical hosts or nodes on the Cloud can host numerous virtual machines, which in turn can host applications and data. Migrating medical imaging applications and data to the Cloud can allow healthcare organisations to realise significant cost savings relating to hardware, software, buildings, power and staff, in addition to greater scalability, higher performance and resilience [5][6]. Cloud Computing uses a 'pay as you go' pricing model whereby users only pay for the amount of resources they consume, e.g., storage, memory, CPU, bandwidth. Additional resources can also be provisioned in an on-demand fashion to allow scaling with application and user demand.

This paper proposes a method for service providers to optimise the combined placement of image processing algorithms (as Virtual Machines - VMs) and image data sets on compute and storage nodes respectively. The state of physical node resources and the network health are given key consideration as critical factors when making placement decisions. The solution uses a genetic algorithm as an initial solution to ensure VMs are placed on nodes, which satisfies SLA and network performance constraints. The results of initial experiments in Section VI show that a genetic algorithm can find optimised solutions, which offer lower total processing cost (image processing and network costs as a function of time) than a random assignment solution. Future work is aimed at improving the convergence time of the genetic algorithm through the design and implementation of a hybrid evolutionary algorithm.

The rest of this paper is organised as follows; Section II describes work relating to optimised VM placement. Section III details a mathematical model of the problem. Section IV defines the design of the proposed solution. Section V describes the initial experiments with results in Section VI. Section VII contains the conclusions and future work.

II. RELATED WORK

Genetic algorithm-based placement solutions have been shown to provide optimised placement in the Cloud [7][8]. Placement of data in Cloud based storage using a genetic algorithm solution has the benefit of reducing the average data access time [7]; however memory, CPU and network constraints are not taken into account in this work. The research presented in [8] is primarily concerned with minimisation of the total execution time and although it does consider network based constraints, critical node constraints such as CPU, memory and storage are not considered. Resource allocation in the Cloud taking CPU and memory requirements in addition to network bandwidth, reliability and throughput requirements has been investigated [9]; but CPU and bandwidth resources are considered as static finite resource with the inability to dynamically scale with demand as and when required. The research outlined above is concerned with the placement of either applications or data independently of one another. Although physical node and

 ISBN: 978-1-61208-216-5

network constraints are taken into account, the placement of application (VM) and associated data is not considered.

Combined application (VM) and data placement taking CPU, memory, storage and network constraints into account has been investigated [10] and a solution using a penalty-based genetic algorithm described; however, the algorithm execution time does show an increase as the number of servers increases, causing a significant delay, which could render it unacceptable if used in a real time solution and may also lead to scalability problems. Hybrid evolutionary algorithms combining the best features of genetic algorithms with the best features of other evolutionary algorithms such as particle swarm optimisation (PSO) [11], ant colony optimisation (ACO) [12], and simulated annealing (SA) [13], have been shown to have a much shorter convergence time than purely genetic algorithm-based solutions [14]. Hybrid genetic algorithms such as the multi agent genetic algorithm [15] can offer superior performance over traditional genetic algorithms when very large scale and dynamic optimisation problems are concerned. Likewise, an improved genetic algorithm (IGA) [16] has been shown to be nearly twice as fast at finding optimised solutions as a purely genetic algorithm placement solution.

III. PROBLEM SPECIFICATION

A. Model Attributes

Processing nodes A and storage nodes B are separated by a network containing a set of network routes R between any set of nodes. A set of virtual machines V containing algorithms are hosted on a set of physical processing nodes A. A set of virtual machines W containing data stores are hosted on a set of physical storage nodes B.

TABLE I. MODEL ATTRIBUTES

Notation	Description
x_{tva}	The placement of task t on vm v on processing node a
y_{dwb}	The placement of dataset d on vm w on storage node b
T	Set of tasks
D	Set of datasets
V	Set of processing virtual machines
W	Set of datastore virtual machines
A	Set of processing nodes
B	Set of storage nodes
R	Set of network routes between nodes a and b
C_a	CPU capacity of processing node a
M_a	Memory capacity of processing node a
S_b	Storage capacity of storage node b
Ct	CPU requirement of task t
M_t	Memory requirement of task t
S_d	Storage requirement of dataset d
C_{tdab}	The cost of task t processing dataset d on nodes a and b
K_a	The network cost between nodes a and b
bw_{ab}	The minimum end to end bandwidth (kbps) of the network path between nodes a and b
lat_{ab}	The network latency (ms)between nodes a and b
T_{sla}	Required response time specified in an SLA

A set of tasks T are executed on a set of processing nodes A. Each processing node a has a resource capacity in terms of CPU C_a and memory M_a. Each task t has resource requirements in terms of CPU C_t and memory M_t. A set of datasets D are stored on a set of storage nodes B. Each storage node b has a resource capacity in terms of storage S_b. Each dataset d has a storage requirement S_d.

B. Mathematical Model

1) Image Processing

$$x_{tva} = \begin{cases} 1 \text{ if task } t \text{ is executed on vm } v \text{ on node } a \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

$$\text{for } t = 1,..,T; v = 1,...,V; a = 1,...,A$$

2) Data Storage

$$y_{dwb} = \begin{cases} 1 \text{ if dataset } d \text{ is stored on vm } w \text{ on node } b \\ 0 \text{ otherwise} \end{cases} \quad (2)$$

$$\text{for } d = 1,...,D; w = 1,...,W; b = 1,...,B$$

3) Objective Function

The aim is to minimise the cost of executing task t on dataset d on processing node a and storage node b – taking the network cost (as a function of time) between a and b into account. Therefore the objective function is to minimise:

$$\sum_{t=1}^{T}\sum_{d=1}^{D}\sum_{v=1}^{V}\sum_{w=1}^{W}\sum_{a=1}^{A}\sum_{b=1}^{B} x_{tva}\, y_{dwb}\, c_{tdab} + \sum_{a=1}^{A} k_a \sum_{t=1}^{T}\sum_{v=1}^{V} x_{tva} \quad (3)$$

4) Network Cost

The network cost K_a between processing node a and storage node b is derived from the dataset size S_d divided by the minimum network bandwidth bw_{ab} plus the network latency lat_{ab} on the end to end network route r between node a and node b.

$$k_a = \sum_{r_{a,r,b} \in R} \frac{S_d}{bw_{ab}} + lat_{ab} \quad (4)$$

C. Physical Constraints

1) Processing Constraint - VM to Processing Node

Each task t is executed on a VM v on a processing node a. Each task t has a CPU requirement C_t. Node a must have sufficient CPU capacity C_a to meet the CPU requirement C_t of task t, subject to:

$$\sum_{t=1}^{T}\sum_{v=1}^{V} c_t\, x_{tva} \le c_a \text{ for } a = 1,..., A \quad (5)$$

2) Memory Constraint – VM to Processing Node

Each task t has a memory requirement M_t. Processing node a must have sufficient memory capacity M_a to meet the memory requirement M_t of task t, subject to:

$$\sum_{t=1}^{T}\sum_{v=1}^{V} m_t\, x_{tva} \le m_a \text{ for } a = 1,..., A \quad (6)$$

 ISBN: 978-1-61208-216-5

3) Data storage constraint – VM to Storage Node

Each dataset d has a storage requirement S_d. Each storage node b must have sufficient storage capacity S_b to meet the storage requirement S_d of dataset d, subject to:

$$\sum_{d=1}^{D}\sum_{w=1}^{W} s_d \; y_{dwb} \le s_b \text{ for } b = 1,\ldots,B \quad (7)$$

4) SLA Time Constraint – Data to User

The total processing time must be less than the required response time specified in the SLA T_{sla}, subject to:

$$\sum_{t=1}^{T}\sum_{d=1}^{D}\sum_{v=1}^{V}\sum_{w=1}^{W}\sum_{a=1}^{A}\sum_{b=1}^{B} x_{tva} \; y_{dwb} < T_{sla} \quad (8)$$

D. Logical Constraints

Each task t has one dataset d, subject to:

$$\sum_{d=1}^{D}\sum_{v=1}^{V}\sum_{a=1}^{A}\sum_{w=1}^{W}\sum_{b=1}^{B} x_{tva} \; y_{dwb} = 1 \text{ for } t = 1,\ldots,T \quad (9)$$

Each dataset d has at least one task t, subject to:

$$\sum_{t=1}^{T}\sum_{v=1}^{V}\sum_{a=1}^{A}\sum_{w=1}^{W}\sum_{b=1}^{B} x_{tva} \; y_{dwb} > 1 \text{ for } d = 1,\ldots,D \quad (10)$$

Each VM v is allocated to at least one processing node a, subject to:

$$\sum_{a=1}^{A} x_{tva} \ge 1 \text{ for } v = 1,\ldots,V \quad (11)$$

Each VM w is allocated to at least one storage node b, subject to:

$$\sum_{b=1}^{B} y_{dwb} \ge 1 \text{ for } w = 1,\ldots,W \quad (12)$$

Each task t is executed on at least one VM v on at least one processing node a, subject to:

$$\sum_{v=1}^{V}\sum_{a=1}^{A} x_{tva} \ge 1 \text{ for } t = 1,\ldots,T \quad (13)$$

Each dataset d is stored on at least one VM w on at least one storage node b.

$$\sum_{w=1}^{W}\sum_{b=1}^{B} y_{dwb} \ge 1 \text{ for } d = 1,\ldots,D \quad (14)$$

IV. SOLUTION DESIGN

The aim of the proposed solution is to optimally place data and image processing algorithms on the service provider infrastructure whilst avoiding customer SLA violation. Figure 1 gives an overview of the proposed system. When placing the image processing application CPU, memory, and network constraints need to be satisfied, likewise when placing data a certain amount of storage, adequate network bandwidth and an acceptable latency is required.

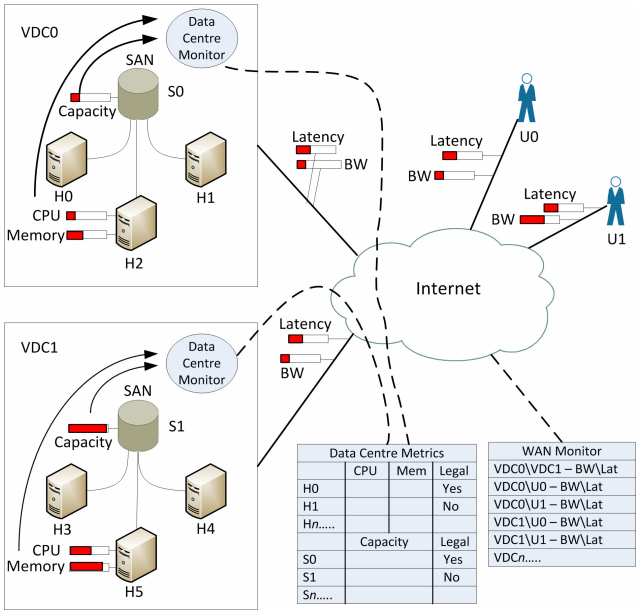


Figure 1. Architectural overview of the proposed system

The 'Data Centre Monitor' is responsible for monitoring the CPU and memory utilisation of hosts (e.g., H0, H1, H2) and the storage capacity of storage area network (SAN) nodes (e.g., S0, S1) within each Virtual Data Centre (VDC). Data centre node metrics are gathered by distributed agents along with network health metrics collected by the 'WAN Monitor', which uses a modified version of BWPing [17] to monitor the end to end bandwidth and latency between all VDCs and users. The node and network health metrics are normalised and form a combined fitness score for each node, which can satisfy the physical and logical constraints. A genetic algorithm is used to find an optimised solution within the pool of viable nodes.

V. INITIAL EXPERIMENTS

A genetic algorithm was developed using Microsoft Visual Studio 2008. A synthetic dataset containing values representing realistic CPU, memory, storage and network metrics for 20 physical nodes was generated. A randomly generated initial population of 50 was used with binary tournament parent selection with a 10% population mutation rate chance. The number of physical nodes was constant at 20, whilst the number of VMs requiring placement increased in increments of 5, ranging from 5 to 75.

Two scenarios were investigated in initial experiments: random placement and genetic algorithm placement. The experiments for each scenario were repeated 30 times and the mean taken. The experiments were conducted on a PC running Windows XP with a 2933 MHz Intel Processor and 4GB of RAM.

 ISBN: 978-1-61208-216-5

VI. RESULTS

The initial results in Figure 2 below show that a genetic algorithm solution (depicted as the lower solid line) produces placement decisions with a lower total processing cost than a random placement solution as depicted by the upper dashed line (initial fittest) in the graph. The costs for each solution are similar when the number of VMs requiring placement are small. Both solutions show a linear increase in cost as the number of VMs requiring placement increases, but the total image processing cost for the genetic algorithm is significantly lower than that of the random placement solution. With a maximum number of 75 VMs for placement the cost associated with random placement is 3229, whilst the genetic algorithm solution is 1294, which is just over 40% of the cost of the random placement solution.

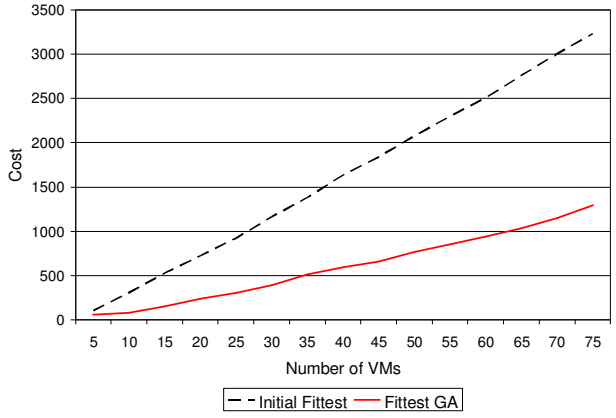


Figure 2. Performance comparison between initial fittest (random placement) and genetic algorithm placement solutions.

VII. CONCLUSION AND FUTURE WORK

A model of VM and data placement including physical node and network constraints was presented. Results from initial experiments show that a genetic algorithm taking multiple constraints into account can be used to make optimised network aware and SLA compliant combined VM and data placement decisions. The total image processing cost was reduced by nearly 60% when compared to a naive random placement solution.

A solution based purely on a genetic algorithm may suffer from scalability issues stemming from long convergence times found in large solution search spaces [10][14], potentially causing unacceptable latency in live systems. Future work will consist of expanding the model to include additional constraints relating to intellectual property (IP) rights. Initial experiments will be scaled to investigate the upper bounds of performance with greater numbers of nodes and VMs, which will be used as an evaluation baseline for future solutions. The development of a hybrid evolutionary algorithm, combining the best features of several evolutionary algorithms will be investigated with the aim of improving performance and resource utilisation.

A prototype system is under development using the NETCOM Cloud testbed facility at the University of Ulster. It will be used to validate current and future results on a dynamic real time Cloud infrastructure.

ACKNOWLEDGMENT

BT EPSRC CASE Award in collaboration with BT EBTIC.

REFERENCES

[1] T. Chia-Chi, J. Mitchell, C. Walker, A. Swan, C. Davila, D. Howard and T. Needham, "A medical image archive solution in the cloud," Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on, pp. 431-434, 16-18 July 2010

[2] H. QingZang, Y. Lei, Y. MingYuan, W. FuLi and L. RongHua, "Medical Information Integration Based Cloud Computing,"Network Computing and Information Security (NCIS), 2011 International Conference on, vol.1, pp. 79-83, 14-15 May 2011

[3] M.I.B. Nordin and M.I. Hassan, "Cloud resource broker in the optimization of medical image retrieval system: A proposed goalbased request in medical application," National Postgraduate Conference (NPC), 2011, pp. 1-5, Sept. 2011

[4] M. Mahjoub, A. Mdhaffar, R.B. Halima and M. Jmaiel, "A Comparative Study of the Current Cloud Computing Technologies and Offers," Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on, pp. 131-134, 21-23 Nov. 2011

[5] L.A. Bastiao Silva, C. Costa, A. Silva and J.L. Oliveira, "A PACS Gateway to the Cloud," Information Systems and Technologies (CISTI), 2011 6th Iberian Conference on, pp. 1-6, 15-18 June 2011

[6] S. Ahmed and A. Abdullah, "E-healthcare and data management services in a cloud," High Capacity Optical Networks and Enabling Technologies (HONET), 2011, pp. 248-252, 19-21 Dec.2011

[7] K. Jindarak and P. Uthayopas, "Performance improvement of cloud storage using a genetic algorithm based placement," Computer Science and Software Engineering (JCSSE), 2011 Eighth International Joint Conference on, pp. 54-57, 11-13 May 2011

[8] A.V. Dastjerdi, S.K. Garg and R. Buyya, "QoS-aware Deployment of Network of Virtual Appliances Across Multiple Clouds," Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, pp. 415-423, Nov. 29 2011-Dec. 1 2011

[9] K.H. Prasad, T.A. Faruquie, L.V. Subramaniam, M. Mohania and G. Venkatachaliah, "Resource Allocation and SLA Determination for Large Data Processing Services over Cloud," Services Computing(SCC), 2010 IEEE International Conference on, pp. 522-529, 5-10 July 2010

[10] Z.I.M. Yusoh and T. Maolin, "A penalty-based genetic algorithm for the composite SaaS placement problem in the Cloud," Evolutionary Computation (CEC), 2010 IEEE Congress on, pp. 1-8, 18-23 July 2010

[11] J. Kennedy and R. Eberhart. "Particle swarm optimization," In IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948, 1995

[12] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey" in Theoretical Computer Science 344 (2–3) (2005), pp.243–278, 2005

[13] S. Kirkpatrick, C.D. Gellatt and M.R. Vechi, "Optimization by simulated annealing". Science, 1983, 220: pp. 671-680

[14] C.C.T. Mark, D. Niyato and C.K. Tham, "Evolutionary Optimal Virtual Machine Placement and Demand Forecaster for Cloud Computing," Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on, pp. 348-355, March 2011

 ISBN: 978-1-61208-216-5

[15] Z. Kai, S. Huaguang, L. Lijing, G. Jinzhu and C. Guojian, "Hybrid Genetic Algorithm for Cloud Computing Applications," Services Computing Conference (APSCC), 2011 IEEE Asia-Pacific, pp. 182-187, 12-15 Dec. 2011

[16] Z. Hai, T. Kun and Z. Xuejie, "An Approach to Optimized Resource Scheduling Algorithm for Open-Source Cloud Systems," ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual, pp. 124-129, 16-18 July 2010

[17] BWPing, http://bwping.sourceforge.net/, accessed 14.05.2012

 ISBN: 978-1-61208-216-5

Cloud Objects: Programming the Cloud with Object-Oriented Map/Reduce

Julian Friedman
Dept. of Computer Science
University of York
York, UK
julz@cs.york.ac.uk

IBM Cloud Labs
IBM United Kingdom Ltd.
Hursley Park, UK
julz.friedman@uk.ibm.com

Manuel Oriol
Dept. of Computer Science
University of York
York, UK
manuel@cs.york.ac.uk

ABB Corporate Research
Industrial Software Systems
Baden-Dättwil, Switzerland
manuel.oriol@ch.abb.com

***Abstract*—Cloud Objects parallelizes Object-Oriented programs written in Java using Map/Reduce by adding simple, declarative annotations to the code. The system automatically persists objects to a partitioned filesystem and efficiently executes methods across the partitioned object data. Using Cloud Objects, data-intensive programs can be written in a simple, readable, object-oriented manner, while maintaining the performance and scalability of the Map/Reduce platform. Cloud Objects shows that it is possible to combine the benefits of an object-oriented model and the power of Map/Reduce.**

***Keywords*-Map/Reduce, Hadoop, JPA, Cloud Computing**

I. Introduction

Data-parallel frameworks such as Map/Reduce [1] have become increasingly popular. The developers of Internet applications have turned to these technologies to harness the power of large numbers of distributed machines to address the challenges of processing huge amounts of data.

Map/Reduce was designed for a specific domain — data-dependent batch computations — it was not designed to be a general approach to designing whole applications. Programmers must fit their code into the structure of a Map/Reduce algorithm. Programming a Map/Reduce application involves splitting the algorithm into separate Mappers, Reducers, InputFormats and Drivers (to name a few) and encourages a tight coupling of these components.

Splitting the application logic between many tightly coupled classes compromises many of the advantages of object-oriented design, such as composability, modularity and encapsulation. The lack of proper object orientation makes it difficult to evolve and compose Map/Reduce programs in to larger systems, to maintain Map/Reduce programs, and to quickly develop new applications using Map/Reduce.

This paper describes *Cloud Objects*, a new system which allows Map/Reduce applications to be written in an object-oriented style. *Cloud Objects* uses simple declarative annotations to describe how object data should be persisted to the distributed filesystem. As well as automatically generating the code to persist the objects to a partitioned datastore (in a manner similar to existing systems such as DataNucleus [2]), the system generates the Map/Reduce code to run methods across the partitioned object data. A program can be structured and composed in an object-oriented style and deployed to existing Map/Reduce clusters.

The paper is structured as follows. The following section, Section II, introduces the programming model with a simple example. Section III describes the programming model in detail. Section IV describes our prototype implementation. Section V discusses *Cloud Objects* in the context of related work. SectionVI concludes the paper.

II. An Introductory Example

Cloud Objects' persistence annotations are based on JPA (Java Persistence Annotations, JSR 317 [3] and 220 [4]). This allows maximum compatibility with existing code and minimizes the need for developers to learn a new syntax. Persisting an object to a distributed store is as simple as using standard JPA annotations. Once persisted, methods can be run in parallel across an object's partitioned data.

In Listing 1, we illustrate the programming model with a simple example. The Wiki class contains a map of page names to WikiPage objects which the framework will persist to the distributed filesystem. Assuming the map is large, the data will be partitioned across many machines.

While persistence annotations alone allow persisting and querying object data in the distributed file system, they do not allow efficient processing of the object data. The scalability and efficiency of the Map/Reduce model is based on the ability to distribute code to data. Map/Reduce is a computation framework as well as a data storage and querying framework. While a simple approach based on JPA alone would suffice to persist the object data, it would not be able to efficiently run object methods with data-locality; it would gain the benefits of the distributed file system to persist and retrieve the object data, but not the advantages of the Map/Reduce model to execute fault-tolerant, resilient methods across the data.

Cloud Objects adds the ability to add @Multicast methods to a class which can be distributed automatically, with data-locality, by the framework. A multicast method across the 'pages' member variable is shown in Listing 1. The multicast method is automatically run across each shard of the partitioned 'pages' variable using Map/Reduce, and the results are combined to a single array using a standard UNION reduction.

 ISBN: 978-1-61208-216-5

```
@Entity public class Wiki {
  @OneToMany @KeyField("url") private Map<String, WikiPage> pages;

  public Collection<String> search(String phrase) { this.search(pages, phrase); }

  @Multicast(reduce=UNION)
  protected Collection<String> search(Map<String, WikiPage> pages, String phrase) {
     Collection<String> matches = new ArrayList<String>();
     for(Map.Entry<String, WikiPage> page : pages) {
       if(page.getContents().indexOf(phrase) > -1) { matches.add(url); }
     }
     return matches;
  }
}
```

Listing 1: A Distributed Wiki

The results of the search(..) method are themselves written to the distributed filesystem and can be processed with data-locality by another @Multicast method. This allows seamless, efficient composition of multicast methods in to full applications.

III. Programming Model

The aim of *Cloud Objects* is to allow the business logic of an application to be expressed in a simple, object-oriented style while efficiently running methods across partitioned object data using Map/Reduce.

Cloud Objects can be split into annotations which relate to persisting and retrieving objects from the datastore (which are based on JPA), and annotations related to running methods across the partitioned data. As the persistence-related annotations are based directly on a subset of the JPA standard, in this section, we focus only on the additional annotations we have introduced to run methods across the persisted object data.

A. Multicast Methods

Multicast methods interact with partitioned instance data by running appropriate Map/Reduce jobs over their input data (which is stored in the distributed filesystem).

The programmer uses an EntityManager instance to retrieve a Cloud Object from the datastore. When the EntityManager retrieves an object, it creates proxy collections to wrap distributed member data. These proxy collections are initialized with the location of their data in the distributed file system, and contain methods to configure a Map/Reduce job to run against their contents.

On the client machine, the EntityManager replaces any methods of a returned instance which have been annotated with the @Multicast annotation using byte-code rewriting.

1) Inputs to Multicast Methods: For simplicity, multicast methods only allow one partitioned input collection to be passed as an argument. The programmer is free to run methods over multiple partitioned inputs by using a multicast method to create a collection containing a cross product or join of two other lists. This resulting list will be stored on the distributed filesystem and can be passed as an input to another multicast method. Alternatively one of the lists (usually the smaller) can be passed as class data and retrieved from the distributed filesystem on demand.

To maximize encapsulation, the programmer is encouraged to provide an external client method (not annotated with @Multicast) which calls a protected or private @Multicast method with the needed arguments. This is shown in the 1-arg and 2-arg versions of the search (..) method in Listing 1.

2) Outputs from Multicast Methods: Safely running a Multicast method across a number of machines requires a number of constraints on multicast methods. If the method replicas were allowed to write directly to the member variables of the class the individual jobs would no longer be independent. Instead, multicast methods may only write to member variables in the following (safe) ways:

a) Shared: The framework sends instance variables marked with the @Shared annotation to every node using Hadoop's distributed cache. On worker machines, any updates made to @Shared variables other than Counters and Joinables (see next) are ignored and may throw exceptions. Updates made to member variables that are not annotated with @Shared are limited to a particular object on a particular node. This can be useful for caches and other data structures which do not need to be maintained across machines.

b) Counters: Counters allow methods to safely update member variables which increase monotonically. Counters are implemented using the underlying Hadoop framework's Counter functionality. Hadoop's Counters are global, which breaks encapsulation. Counters in *Cloud Objects* are automatically given generated, private IDs based on the object class and the unique identity of the object instance.

c) Joinables: Joinables allow for a more general method of updating an instance variable from multiple methods. Joinables are inspired by the Concurrent Revisions programming model [5]. Joinables may be declared either by deriving from the Joinable marker interface or by the addition of

 ISBN: 978-1-61208-216-5

a specific @JoinedWith(..) annotation. Classes which inherit from the Joinable marker interface are expected to have either a static join(..) method or to be themselves annotated with @JoinedWith(..) to refer to a class with a no-arg constructor and a join(..) method.

B. Reductions

The final result from a multicast method is reduced to a single value using a Reducer class. A set of default reducers is provided for Unions, Sums and Averages, and the programmer is free to name their own reduction class in the @Multicast(reduce = ..) annotation.

IV. Implementation

We have implemented a prototype of *Cloud Objects* based on OpenJPA [2] and Hadoop [6]. We briefly describe the key elements of the implementation in this section.

A. Collections Proxies

Instance variables of *Cloud Objects* which are specified as Lists or Maps are automatically proxied with a HadoopList or a HadoopMap class which reads and writes from the Distributed Filesystem when the object needs to be persisted. This is done by the EntityManager when retrieving the object, and by the generated Mapper classes when they create an object instance to process partitions of a Multicast Method's input data.

The distributed collection classes support two modes of operation. When used as input to a Multicast Method (on the master machine), the collection classes provide a configureInput(Job job) method which configures a Hadoop Job with the input directory containing the collection's data and an appropriate InputFormat which can parse the data. Each Map job is then provided with a single shard of the partitioned data. The shard for the Map job is created using the createShard(Class<T>, Mapper.Context) method of the HadoopMap and HadoopList classes, which converts the input key/value pairs for the current map task in to the type of collection required by the mapper method.

When accessed as an instance variable rather than as a parameter to a MultiCast method, a distributed collection reads and writes its object data from the distributed filesystem. This is potentially inefficient as the object data is unlikely to be local, but is useful for tasks such as printing out the final results of a computation.

B. Multicast Methods

Proxied objects are obtained using a custom JPA EntityManager. We use the open-source OpenJPA [2] implementation of the JPA standard which uses byte-code rewriting to extend plain java objects with persistence information. When the custom EntityManager returns a persistent object or collection, it is scanned for methods annotated with @Multicast and if these are present they are overriden to be dispatched via Hadoop.

The MulticastInvoker class is responsible for dispatching Multicast methods to run on the Map/Reduce cluster using Hadoop. A single Mapper and Reducer (DefaultMapper and DefaultReducer) are used for every job. These classes are configured using job configuration variables set by MulticastInvoker. For example, the DefaultMapper consults the 'com.ibm.cloudlabs.cloudobjects.multicast.target.class' variable; this variable records the class which will be used on each node to run the 'meat' of the job. MulticastInvoker delegates to the input collection to set the job input path based on the location of the passed object on the distributed filesystem.

Each Map job creates a new copy of the delegate object using the no-arg constructor, which must be present in the class. Any @Shared or Joinable variables in the class are initialized from the distributed cache, and HadoopCollection.createShard(..) is used to create a shard of the multicast variable to be passed to the method from the input pairs. Output is saved to a directory configured by a HadoopCollection or HadoopMap and the client automatically creates and returns a proxy collection wrapping the output directory.

C. Joinables

Joinable variables are initialised before a method is run using the distributed cache, so that each node has the same initial value. During the multicast method, the joinable variable maintains any values set during the method. This preserves the independence of the map jobs. The Mapper implementation writes both updated joinable values and the results of the multicast method to the map output, prefixing a 0 or 1 to the stream to differentiate each case. These outputs are sorted by the framework and passed to the reducer. The Reducer merges Joinable variables using the appropriate Joiner class for the variable and delegates to the configured Reduction class to create the final result of the method.

V. Related Work

The Hadoop [6] implementation of the Map/Reduce algorithm [1] provides a Java API to Map/Reduce. This API is, however, a low-level API which requires the programmer to express computations as collections of Map jobs, Reduce jobs and Driver classes. All of these interact to perform a computation and collect results over a distributed, partitioned datastore. Map/Reduce is typically not object-oriented because it requires programmers to express jobs in a functional way. *Cloud Objects* allows applications to use an object oriented style while taking advantage of the scale and power of Map/Reduce. While *Cloud Objects* does not have the full generality of Map/Reduce - in particular, many Map/Reduce algorithms are in practice tuned using techniques such as In-Mapper Combiners, Pairs and Stripes (see e.g., [7]), which rely on a tight coupling between Mapper and Reducer - we believe it is a promising method for creating large scale applications. While Map/Reduce programs tend to rely on tight coupling between Mapper and Reducer, *Cloud Objects* favours the use of standard, reusable reducers - though custom reducers are supported - and higher-level concepts such as Joinable types and Counters.

 ISBN: 978-1-61208-216-5

Cloud Objects follows a trend of higher-level and domain-specific languages such as Pig [8], Hive [9] and JAQL [10] built on top of Map/Reduce. The aim of these languages is to retain the performance, reliability and scalability benefits of Map/Reduce, while presenting a more familiar, simpler or high-level style to programmers.

Sawzall [11] runs on top of Map/Reduce and, similarly to our approach, uses a set of standard reducers to aggregate outputs from custom map methods. Sawzall uses a custom scripting language which processes a single input and emits values to output types such as sum tables and maximum tables which retain the total of their input and the largest of all of their inputs, respectively. Sawzall is however different from the programming language in which the rest of the application is coded. *Cloud Objects* solves this issue.

Pig [8] is an imperative language with a number of group-based built-in functions such as co-joins, projections and restrictions. The aim of Pig is to provide an easy way for programmers familiar with imperative programming to query distributed data using Map/Reduce. It does not provide any way of writing scripts in an object oriented style and focuses on ad-hoc querying of existing data.

JAQL [10] is closer in spirit to *Cloud Objects*, providing a pure functional language for querying Javascript Object Notation (JSON) objects using Map/Reduce. JAQL is designed for ad-hoc queries of large data rather than writing maintainable programs, and while it allows querying serialised objects, it does not provide features to allow its own programs themselves to be written in an object oriented style.

Hive [9] presents an SQL-like declarative interface for querying large-scale data using Map/Reduce. This lacks the generality of the *Cloud Objects* approach.

Collection-style interfaces such as FlumeJava [12] and Crunch [13] have advantages over domain-specific languages such as Pig and JAQL in that they allow the program to be expressed in a single language and are perhaps closest to our approach. These interfaces allow complicated pipelines of operations on collections of objects to be efficiently optimised in to a set of Map/Reduce jobs. These systems focus on manipulating object collections rather than on adding data-parallel methods to existing object-oriented programs.

Other tools exist which provide JPA bindings from Java objects to partitioned data stores such as HDFS. DataNucleus [2] is an open-source JPA provider with support for a variety of backends including Hive. Users of Google's AppEngine [13] environment can use JPA to persist objects to the AppEngine data store, and can separately use the Map/Reduce functionality of AppEngine to run map jobs over entities in the data store.

Alternatives to Map/Reduce also exist. For example, in the .Net ecosystem, Dryad [14] and DryadLINQ [15] have become popular frameworks for expressing data-parallel computations. Dryad provides a more generic model than Map/Reduce, allowing arbitrary directed acyclic graph computations, and DryadLINQ provides a language-integrated query language which can compiles to Dryad jobs. Another example is Sky-writing [16] which provides a functional coordination language to describe computations to be run on CIEL [17], a Map/Reduce-like system for cluster computation. *Cloud Objects* are at a higher level of abstraction and could be applied on top of these as well.

VI. Conclusion

This paper introduced *Cloud Objects*, an object-oriented programming model which exposes the power of Map/Reduce in a simple, encapsulated, modular way. To use *Cloud Objects*, programmers only need to add a couple of annotations to a regular Java program.

Our prototype implementation of *Cloud Objects* uses Hadoop [6] to distribute the actual code and data and extends OpenJPA [2] to store and retrieve persistent objects to a distributed filesystem. We have benchmarked the prototype using EC2. Initial experiments show that the overhead induced is negligible compared to the cost of computation and network.

References

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008. [Online] Available: http://labs.google.com/papers/mapreduce-osdi04.pdf [Accessed: 14 May 2012].

[2] "Datanucleus," http://www.datanucleus.org/ [Accessed: 14 May 2012].

[3] L. DeMichiel, "Jsr 317: Java persistence 2.0," 2009.

[4] L. DeMichiel and M. Keith, "Jsr 220: Enterprise javabeans," 2006.

[5] S. Burckhardt, A. Baldassin, and D. Leijen, "Concurrent programming with revisions and isolation types," in *Proceedings of the ACM international conference on Object oriented programming systems languages and applications*, ser. OOPSLA '10. New York, NY, USA: ACM, 2010, pp. 691–707.

[6] "Apache hadoop," http://hadoop.apache.org [Accessed: 14 May 2012].

[7] J. Lin and C. Dyer, *Data-Intensive Text Processing with MapReduce*. Morgan & Claypool Publishers, 2010.

[8] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin: a not-so-foreign language for data processing," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 1099–1110.

[9] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: a warehousing solution over a map-reduce framework," *Proc. VLDB Endow.*, vol. 2, pp. 1626–1629, August 2009.

[10] "Jaql: Query language for javascript object notation," http://code.google.com/p/jaql/ [Accessed: 14 May 2012].

[11] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan, "Interpreting the data: Parallel analysis with Sawzall," *Sci. Program.*, vol. 13, pp. 277–298, October 2005.

[12] C. Chambers, A. Raniwala, F. Perry, S. Adams, R. R. Henry, R. Bradshaw, and N. Weizenbaum, "FlumeJava: easy, efficient data-parallel pipelines," in *Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation*, ser. PLDI '10. New York, NY, USA: ACM, 2010, pp. 363–375.

[13] "Google app engine," http://code.google.com/appengine/ [Accessed: 14 May 2012].

[14] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*. ACM, 2007, pp. 59–72.

[15] Y. Yu, M. Isard, D. Fetterly, M. Budiu, U. Erlingsson, P. K. Gunda, and J. Currey, "DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language," in *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, ser. OSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 1–14.

 ISBN: 978-1-61208-216-5

[16] D. Murray and S. Hand, "Scripting the cloud with skywriting," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. USENIX Association, 2010, pp. 12–12.

[17] D. Murray, M. Schwarzkopf, C. Smowton, S. Smith, A. Madhavapeddy, and S. Hand, "Ciel: a universal execution engine for distributed data-flow computing," in *Proceedings of NSDI*, 2011.

 ISBN: 978-1-61208-216-5

Reliable Approach to Sell the Spare Capacity in the Cloud

Wesam Dawoud
Hasso Plattner Institute
Potsdam University
Potsdam, Germany
wesam.dawoud@hpi.uni-potsdam.de

Ibrahim Takouna
Hasso Plattner Institute
Potsdam University
Potsdam, Germany
ibrahim.takouna@hpi.uni-potsdam.de

Christoph Meinel
Hasso Plattner Institute
Potsdam University
Potsdam, Germany
christoph.meinel@hpi.uni-potsdam.de

***Abstract*—Traditionally, Infrastructure as a Service (IaaS) providers deliver their services as *Reserved* or *On-Demand* instances. Spot Instances (SIs) is a complementary service that allows customers to bid on the free capacity in the provider data centers. Therefore, the decrease in the free capacity may result in terminating instances abruptly. To ensure fair trading, the provider does not charge customers for the interrupted partial hours. However, our experiments show that uncharged time could rise up to 30% of the instance total run time, which means a reduction in the provider's profit. In this paper, we propose an Elastic Spot Instances (ESIs) approach, where instead of abruptly terminating the SI, the provider scales down their capacity proportionally to the increase in the price. Our approach delegates the task of interrupting the instances into the customers, but at the same time keeps the control in the provider side to isolate SIs' impact on the other services. We validate our approach along different periods of SIs history traces.**

Keywords- IaaS; Spot instances; Dynamic scalability.

I. INTRODUCTION

Amazon is the first cloud provider to come up with SIs purchasing system to sell the spare capacity after fulfilling the requests for *Reserved* and *On-Demand* instances. The price of SIs changes dynamically according to free capacity and actual demand. The requests for new SI with bid price higher than or equal the current spot price will be served. On the other hand, if the current prices exceeded the user bid, provider will terminate out-of-bid instances abruptly. SIs reduce the prices from 38% to 44% of the *On-Demand* prices [1]. However, SIs customers are supposed to modify their applications to manage the abrupt termination of SIs.

To manage SIs termination, customers can implement fault tolerant architectures such as MapReduce [2], Grid, Queue-Based [3], and Checkpointing [4][5][6]. The first three architectures typically run two types of nodes (master and worker). One of the master nodes tasks is to manage the failure of worker nodes. The best practice is to run master nodes on *On-Demand* or *Reserved* instances and run worker nodes on SIs to benefit from price reduction. However, these architectures imply major modification to customers' applications. On the other hand, checkpointing is a simple traditional fault tolerant technique. It keeps application execution progress by storing the current state (i.e., snapshot) of the running instance into a persistent storage. Nevertheless, bad checkpointing strategies could impact the performance drastically [7]. For instance, frequent checkpointing results in a high cumulative overhead (i.e., computation is paused at checkpointing time). On the other hand, infrequent checkpointing results in a high overhead caused by the high recovery time (i.e., much computation should be repeated again).

The main goal of this paper is to reduce the checkpointing overhead in SIs environment. This is motivated by the following facts: First, checkpointing is a simple fault tolerant technique that does not require major modifications to customers' applications. Second, checkpointing could be integrated to the other fault tolerant architectures to increase their reliability. Finally, and most importantly, if customers can have checkpoints exactly before terminating VMs instances (i.e., Optimal Checkpointing), then there is no need for the concept of unpaid partial running hours, which on consequently increases the provider profit.

In the next section, we study Amazon EC2 SIs implementation. In Section III, we discuss our proposed ESIs approach: the algorithm, the advantages, and the potential technical challenges. In Section IV we compare our proposed approach performance with current implementation of SIs using price history traces. In Section VI, we present related work done to improve the trade-off between price, reliability, and total run time of applications on SIs. Finally, in Section VII, we conclude and represent our future work.

II. AMAZON EC2 SIS

In this section, we give an overview to Amazon EC2 SIs because it is the first provider who offers SIs purchasing system. The purpose of this section is to determine SIs characteristics that we should consider in our approach.

A. Infrastructure

Amazon EC2 infrastructure [8] is distributed into regions (e.g., US East "Northern Virginia", US West "Northern California", etc.). To prevent failure propagation, each Region is separated into many availability zones. This infrastructure mainly delivers *Reserved* and *On-Demand* instances. The spare capacity is sold as SIs. The SI, as well as the *Reserved* and *On-Demand* instance, can be one of many types depending on resources capacity (e.g., High-CPU Medium

 ISBN: 978-1-61208-216-5

Instance "c1.medium", High-Memory Extra Large Instance "m2.xlarge", etc.).

SI's price is determined by the type, the region, and the operating system. Unlike Zhang et al.'s [9] assumption, in our approach we assume that a physical machine, at the provider side, hosts only instances of the same type and operating system. We support our assumption by observing CPU specifications of each EC2 instance type.

B. Is it a market-driven auction?

Amazon describes SIs purchasing system as a market-driven auction [3]. For example, if the provider has N free resources and it received K bids on the resources, then the provider accepts only the highest N bids, where K is greater than N. The price will be the lowest bid value of the winning subset of the bids. However, by analyzing history traces of SIs' price, Javadi et al. [1] showed sharp changes in the *inter-price* time (i.e., time between price changes) occurred on specific dates at different regions. Javadi et al. conclude that it is artificial (i.e., done by Amazon and not driven by customers demand).

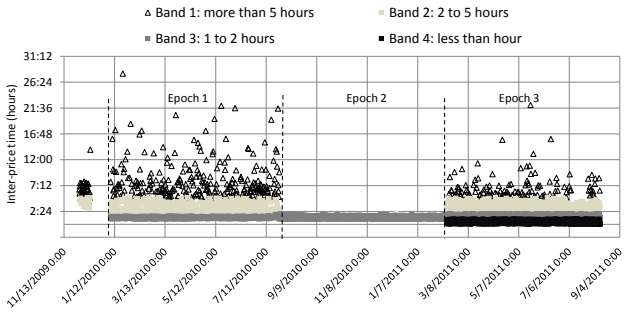


Fig. 1. (US-East), High-CPU Medium Instance's *inter-price* time

Ben-Yehuda et al. [10] went further by showing that the prices also are determined artificially by a random reserve price algorithm and do not represent real customers bids around 98% of the time. They expected that the aim of this random reserved price is to prevent customers from: 1-being complacent and force them to bid higher. 2-inferring the provider's real capacity. Therefore, in case of low number of bids on specific instances type (i.e., K is lower than N), the provider either accepts the lowest bid as the current price or generates a higher value (i.e., pretends less resources [9]) to sell the resources with a higher price. This behavior raises a question about the efficiency of approaches that model the SIs prices. However, we attribute the direct control of the provider on the price to the lack of demand for some instances' type. Nevertheless, regardless of the aim of this random reserved price, our approach does not disclose any hidden information about provider's capacity while it proposes changes to purchasing system rather than to pricing system of SIs.

III. Elastic Spot Instances (ESIs)

We propose ESIs approach to increase the efficiency of selling the free capacity of the IaaS provider. It assumes modifying the SIs purchasing system to increase its reliability without influencing the other hosted services. Fortunately, these modifications do not imply major modification to the current providers' infrastructure while many IaaS providers already use virtualization technologies that can easily accommodate our approach.

Current SIs implementation reacts with the increase demand on *On-Demand* and *Reserved* instances by increasing the SIs price. As a result, out-of-bid SIs are evicted to free more capacity for the complementary services. From the customer point of view, this reduces the SIs reliability. From the provider side, the abrupt interruption of the SIs results in partial unpaid hours (i.e., in some cases, provider will not be paid up to 30% of an instance total run time). Moreover, if the users managed to delay the SI termination, as discussed by [11] and [5], this also increases the probability of unpaid running time.

Implementing our approach requires the following modifications to current SIs purchasing algorithm: First, provider should determine min and max price for each instance type. Second, instead of terminating out-of-bid SI the provider scales down instance's capacity to a value proportional to the increase in the price. Third, running instances can be charged per second because VM instance termination is delegated to the user. According to these modifications, the capacity of ESI can be calculated using Algorithm 1.

Algorithm 1 ESIs's purchasing algorithm

Input: max_price, min_price, current_price, min_cap, and user_bid
Output: VM_capacity
// Calculate the scaling step size
$scale\_step \leftarrow 100/(1000 * (max\_price - min\_price) + 1)$
// Calculate next capacity of VM
if $user\_bid \geq current\_price$ **then**
  $VM\_capacity \leftarrow 100$
else
  if $user\_bid < current\_price$ **then**
    $VM\_capacity \leftarrow 100 - scale\_step * 1000 * (current\_price - user\_bid)$
  end if
  //To prevent VM from starving
  if $VM\_capacity < min\_cap$ **then**
    $VM\_capacity \leftarrow min\_cap$
  end if
end if

According to [1], the price history of most SIs types, except for some types in US-East data center, could be modeled as a Mixture of Gaussian distributions with three or four components with a high fit. This gives the impression that Amazon already has soft minimum and maximum price thresholds for each spot instances type. Moreover, to prevent negative and very low capacities of VM instances, we propose having a minimum capacity of the VM resources, as seen in Algorithm's 1 input. In Section V, we discuss calculating this value considering the free capacity at the physical host.

 ISBN: 978-1-61208-216-5

Algorithm 1 shows that the provider will not have the control to terminate the SIs. At first glance, it seems that customers will be complacent and can simply use a very low bid strategy to have a continued run with a low price. However, if we take the example of "US-West, Linux, High-CPU Medium" instance, the probability density function shows that 99.8% of the prices fall between 0.076 and 0.084. Therefore, scale_step value in Algorithm 1 is calculated as $100/(1000 * (0.084 - 0.076) + 1) = 11.11$, which means that whenever the Spot Price surpasses user bid with 0.001, the capacity of the instances scales down to $(100 - 11.11) \approx 89$. If a user submitted a low bid, for example 0.077, the user will be charged 0.077 per hour for a full capacity instance (i.e., 100%). However, when the market price jump to 0.081, the instance capacity will be scaled down to $100 - 11.11 * (0.081 - 0.077) \approx 56\%$. In spite of the fact that the instance is charged 0.077 per hour, the price is almost doubled according to the low allocated capacity. By this concept, at the case of the overloading, instead of terminating the instances by the provider, the high ratio of price to capacity will push the customers to mange terminating SIs for the optimal price. In Section IV-C, we will discuss the bidding strategies on the light of the proposed modifications to SIs purchasing system.

A. Technical Challenges

Our approach depends on the virtualization technologies' ability to scale the virtualized resource dynamically. However, isolation is a prerequisite for virtualized resources' scalability. It is a demanding problem attracts many researchers [12], [13], [14]. In this section, we discuss isolation and scalability of the following resources: CPU, I/O, and Memory.

CPU isolation is the scheduler's responsibility. Each scheduler has policy that controls the assigned capacity and CPU's time for each virtual CPU (vCPU). Schedulers allow users to change the vCPU's configuration dynamically. However, each scheduler has its characteristics that make it suitable for some environments more than others. For instance, Xen [15] has three schedulers: Borrowed Virtual Time (BVT), Simple Earliest Deadline First (SEDF), and the Credit scheduler [16]. Among these schedulers, only SEDF and Credit scheduler have a non work-conserving mode, which enable the scheduler to cap the capacity of the CPU to specific value (e.g., 50% of the CPU capacity). In spite of the fact that SEDF shows less CPU allocation errors compared to Credit scheduler [16], the global fairness of Credit scheduler makes it the best candidate for our approach.

As in the case of CPU isolation, I/O isolation is also of schedulers' responsibility. However, current implementation of the hypervisors shows that an I/O-intensive VM can influence the performance of other VMs. For instance, in Xen [15], I/O device follows a split-driver model. Therefore, only an Isolated Device Domain (IDD) has access to the hardware using native device drivers. Cherkasova [16] and Gupta et al. [17] demonstrated that the I/O model of Xen complicates CPU allocation and accounting since an IDD processes I/O on behalf of guest VMs. To enhance the accounting mechanism, [17] proposed SEDF-DC. It accounts the CPU usage of an IDD into corresponding guest domains that trigger I/O operations. However, SEDF-DC is still a prototype and it is not implemented to the deployed version of Xen. We leave integrating SEDF-DC to our approach to our extended work.

At initialization time of VMs, the hypervisor allocates an isolated virtual memory for each VM. Memory isolation makes the VM unaware of other VMs' or hypervisor memory demand. A Ballooning technique is developed to enable passing memory pages back and forth between hypervisor and hosted VMs. However, it requires the cooperation of the VM's operating system. Therefore, VM's operating system should be plugged with balloon driver to enable the communication between the VM's operating system and the hypervisor. In case that the hypervisor decide to reduce the VM's memory size (i.e., reclaim pages from VM and inflate the balloon [18]), it determines the target balloon size. If the VM's operating system has plenty of free physical memory, inflating the balloon will be done by just pinning free memory pages (i.e., prevent access to these pages). However, if the VM's is already under memory pressure, the operating system should decide about the memory pages that should be paged out to the virtual swap device[18]. In spite of the fact that paging impacts the VMs performance, Ballooning technique shows better performance compared to Hypervisor Swapping reclamation technique [18]. The lack of the knowledge about the pages contents, in case of Hypervisor Swapping, may result in paging VM's kernel, which has a significant impact on the VM performance. In this paper, we used the balloon driver implemented by Xen to scale down the memory of VMs with the price increase. However, the reality of initiated CPU-intensive workload, in our experiments, did not examine the memory scalability performance. Therefore, we leave examining our approach's performance against different kinds of workload to our future work.

IV. Evaluation

To validate our approach, we carried out two sets of experiments on the physical hardware and using simulation. The first set of experiments, carried out on our Xen test bed, focuses on modeling the virtual machine against CPU-intensive workload with different values of CPU capacity. The other set of experiments carried out by feeding our simulator with the extracted model to simulate running a job of 168 hours (one week). We chose this length of job according to [1] observation that the Spot Price follow specific patterns during the weekdays. Moreover, long run jobs gave us consistence results compared with short jobs. The job run is simulated on a SI and on ESI using SIs' price history traces that are gathered by [19].

A. VM model for CPU-intensive workload

To extract the VM instance model, we ran a VM with two cores on Xen 4.1 hypervisor. The physical server has 2.8 GHz Intel Quad Core i7 Processor and 8GB of physical memory. The workload is CPU-intensive workload generated

 ISBN: 978-1-61208-216-5

by EP Embarrassing Parallel, which is one of NAS Parallel Benchmarks (NPB) [20]. The benchmark generates independent Gaussian random varieties using the Marsaglia polar method. The throughput is measured by Million Operations Per second (MOPs).

At the beginning, the VM instance runs with its full capacity (i.e., 100%). As seen in Fig. 2, the throughput is 37.92 MOPs and the execution time is 56.6 seconds. The same workload is run many times but for different capacities of the VM's CPU. In our experiment, we use Xen *Credit Scheduler* as an actuator for setting the CPU capacity limit of the VM. The *Credit Scheduler* has a non work-conserving mode, which prevents an overloaded VM from consuming the whole CPU capacity of the host and consequently degrading the other VMs performance. For each CPU capacity, we recorded both the MOPs number and the total execution times.

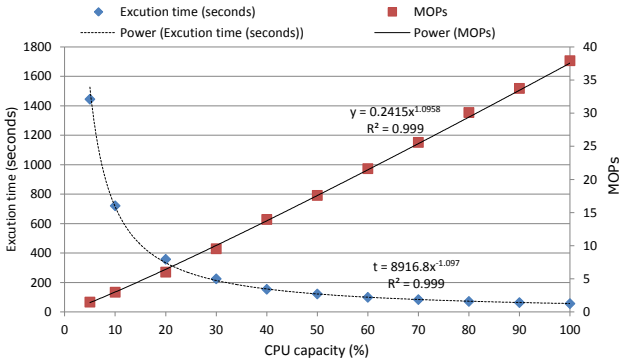


Fig. 2. VM's model against CPU-intensive workload

As seen in Fig. 2, the instance's throughput changes linearly with the virtual CPU (vCPU) capacity according to the following equation:

$$0.2415 * x^{1.0958} \quad (1)$$

where x is the vCPU capacity. However, as the capacity of the VM's vCPU decreases, the execution time increases. At very low capacities (i.e., less than 20%) the execution time increases rapidly. Therefore, in Section IV-C, we explain the bidding strategies that avoid inefficient run for VMs instances.

B. SI simulation

In this section, we simulate running a job of 168 hours on a SI. We chose High-CPU Medium Instance (c1.medium) type while it is an instance type offered to deliver high CPU computation power. Moreover, we selected US-East Region specifically because it shows different values for *inter-price* bands. We would like to study the influence of these bands on the provider profit (i.e., the percentage of unpaid computation hours), as well as on the SI's performance.

In our simulation, we use optimal checkpointing strategy (i.e., checkpointing exactly before the instance termination). During checkpointing time, the computation in VM is paused [11]. Moreover, restoring a VM mounts additional overhead to the checkpointing technique. Sotomayor et al. [21] provide a model to estimate the suspension and restoration time of a VM. The model depends on the number of the co-located VMs and the storage location (i.e., local or remote). However, we cannot predict the number of co-located VMs at public cloud providers. Therefore, we depend on measuring the time required for having a snapshot of c1.medium instance type. Measurements are done 10 times on different time slots of the day at US-East Region. The measured value was always less than a minute. On the other hand, measuring restoration time was ambiguous. Even with very high bids, the measured time between submitting a request and running a SI, from a snapshot, was measured to be 7 to 10 minutes. It is clear that, the bidding algorithm impacts restoring a SI even for very high bids. Therefore, in our simulation, we use the suspension and restoration time which have been estimated by [21] for two VMs to a remote storage. The values are 120 seconds for suspension and 150 seconds for resumption.

In the simulator, we describe the workload as the number of operations that can be done in 168 hours, which could be calculated by (1) as: $168 * 60 * 60 * 0.2415 * x^{1.0958}$ where x = 100. To cover many pricing patterns we chose different starting times from us-east-1.linux.c1.medium spot instance's price history: 2010-01-02, 2010-09-09, and 2011-05-01. These days are selected to span different variations of *inter-price* Bands. However, we verified that running the job on other days, within the same epochs, behaves the same with a slight difference in the price and total run time. The bids range from 0.057 to 0.063 while probability density function, of the history prices of us-east-1.linux.c1.medium, shows that 99.64% of the prices fall within this range.

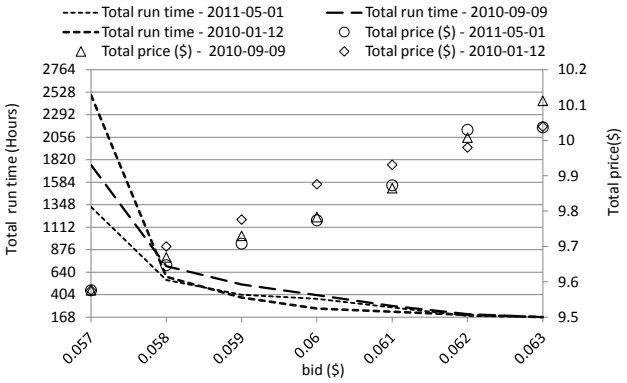


Fig. 3. Spot Instance running 168 hours job

Fig. 3 shows the following concepts: First, low bids lead to lower price but longer run time. Second, high bids lead to higher price but shorter run time. Moreover, the simulation of SI at 2010-09-09 shows a longer run time for most bid values compared with the other simulation dates. This is because of the short *inter-price* at epoch 2 (i.e., Band 3 is 1 to 2 hours). To study the influence of the *inter-price* time on the provider profit, we sum up the partial hour for each bid value. The result value is divided by the total run time to get the percentage of *Unpaid running time*. Results are illustrated in Fig. 4.

In Fig. 4, for the simulation date 2010-01-12, the provider's

 ISBN: 978-1-61208-216-5

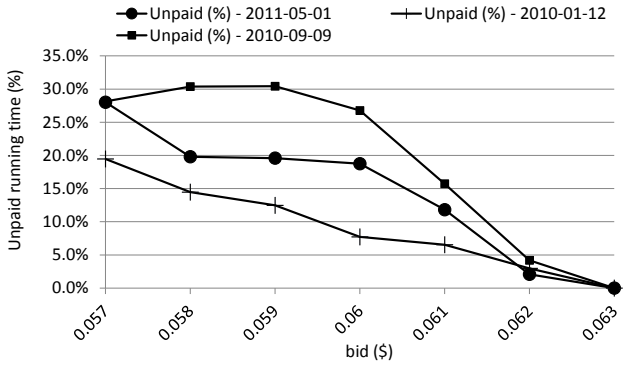


Fig. 4. Unpaid running time(%) - A Spot Instance running 168 hours job

loss (i.e., *Unpaid running time*) at low bids could be 20% of the total run time. However, the loss decreases with the higher bids values. The simulation date 2010-09-09 shows the highest loss for the provider (i.e., from 25% to 30% at low to medium bids) according to the short *inter-price* time at this epoch. Simulation at 2011-05-01 shows a reduction in provider loss compared with that done at 2010-09-09. However, it is higher than that at 2010-01-12. By this observation, we could explain the goal behind the appearance of the Bands 1 to 3 again starting from 2011-02-09. However, we cannot find any reasonable explanation behind the appearance of the Band 4 again in us-east region.

C. ESI simulation

In this section we simulate running the same described job but on ESI. The main goals of this experiment are the following: first, to observe the proposed approach consistency with the bidding concepts: 1-Low bids lead to lower price but longer run time. 2-High bids lead to higher price but shorter run time. Second, to study the proposed approach impact on the bidding strategies, and to suggest bidding strategies that boost the checkpointing technique to a level close to the optimum.

As described in Section III, when the market prices surpass the user bid, the provider reduces the VM capacity with a value proportional to the difference between user bid and the current price. However, a long run at higher price is inefficient, while it implies purchasing lower capacity with a higher price. It is user responsibility to find the best time to take a snapshot and turn off the running instance. To do that, in addition to the bid value, which is passed to the provider, the client should keep in mind another limit price value. Whenever the market price exceeds this limit, the user will take a checkpoint then terminate the instance.

We chose the same days of "US-East, Linux, c1.medium" instance price history to cover different pricing patterns as in the last experiment. The x-axis in Fig. 5(a) to Fig. 5(f) is the limit of the instance price. It is determined by the user to avoid long running of the instance at a high price. We assume that the user will start checkpointing process once the spot market price exceeded this limit value. In our simulator, it is implemented as 2 minutes delay for having a checkpoint then turning off the VM. User will be charged for this running time. Moreover, we consider that the job execution is paused during checkpointing time.

As well as the bid value, the limit value significantly affect the total price and total run time, as seen in Fig. 5. For example, in Fig. 5(a) and Fig. 5(d), if the user bid is 0.057 and the limit value is 0.058, then the total cost for running the job is 9.726$, while the total run time is 1988 hours and 12 minutes. For the same bid, if the limit value is raised to 0.059, the total cost will increase to 9.821$, while the total run time decreases to 1881 hours and 30 minutes. On the other hand, the same total price could be achieved by the bid value 0.058 and limit value 0.059 with a significant reduction in the total run time (i.e., time reduced from 1881 hour and 30 minutes to 528 hours and 18 minutes). This leads us to conclude that the optimal bid and limit values are those which satisfy the following relation: $Limit = Bid + 0.001$. It is consistent with the bidding concepts shown at the beginning of this section. Moreover, Fig. 5 shows that biding according to the relation $Limit = Bid + 2 * 0.001$ could be a good strategy, especially for average values (e.g., bid: 0.059 and limit: 0.061). However, this behavior is not consistent with all bid values because it depends on the prices distribution.

To compare the performance of the ESI with the SI, we consider Optimal checkpointing strategy as a reference. For the three simulation dates, we select the lowest bid (i.e., 0.057), the mean bid value (i.e., 0.060), and the highest bid value (i.e., 0.063). In our comparison, we consider two metrics, the total price and the total run time, where the lower normalized $Price \times Time$ is the better.

TABLE I
NORMALIZED $Price \times Time$ FOR EXECUTION ON MIN-BID, MEAN-BID, AND MAX-BID. THE REFERENCE IS A SI WITH OPTIMAL CHECKPOINTING STRATEGY SHOWN IN FIG. 3

	low-bid (0.057)	mean-bid (0.060)	max-bid (0.062)
2010-01-12	1.016 × 0.796	1.026 × 0.944	1.054 × 1.000
2010-09-09	1.026 × 0.812	1.050 × 0.797	1.047 × 1.000
2011-05-01	1.016 × 0.737	1.042 × 0.817	1.054 × 1.000

If we compare the results in Table I with what obtained by [7], the lowest normalized $Price \times Time$ of the instance us-east.c1.medium with low bid (i.e., 0.058) was 1.266. However, with our approach, even for a lower bid value (i.e., 0.057), the normalized $Price \times Time$ values were 0.809, 0.833, and 0.749 for the simulation dates in consequence, which means 34% to 40% reduction in normalized $Price \times Time$. Moreover, for the mean bid value (i.e., 0.060), the lowest normalized $Price \times Time$ of the same instance was 1.332. However, with our approach it is reduced to 0.969, 0.837, and 0.851 for the three simulation dates in consequence, which means 27% to 37% reduction in normalized $Price \times Time$.

Finally, we should remind that the *Unpaid running time* in case of ESIs is zero, which means that the provider will not lose any computation power.

 ISBN: 978-1-61208-216-5

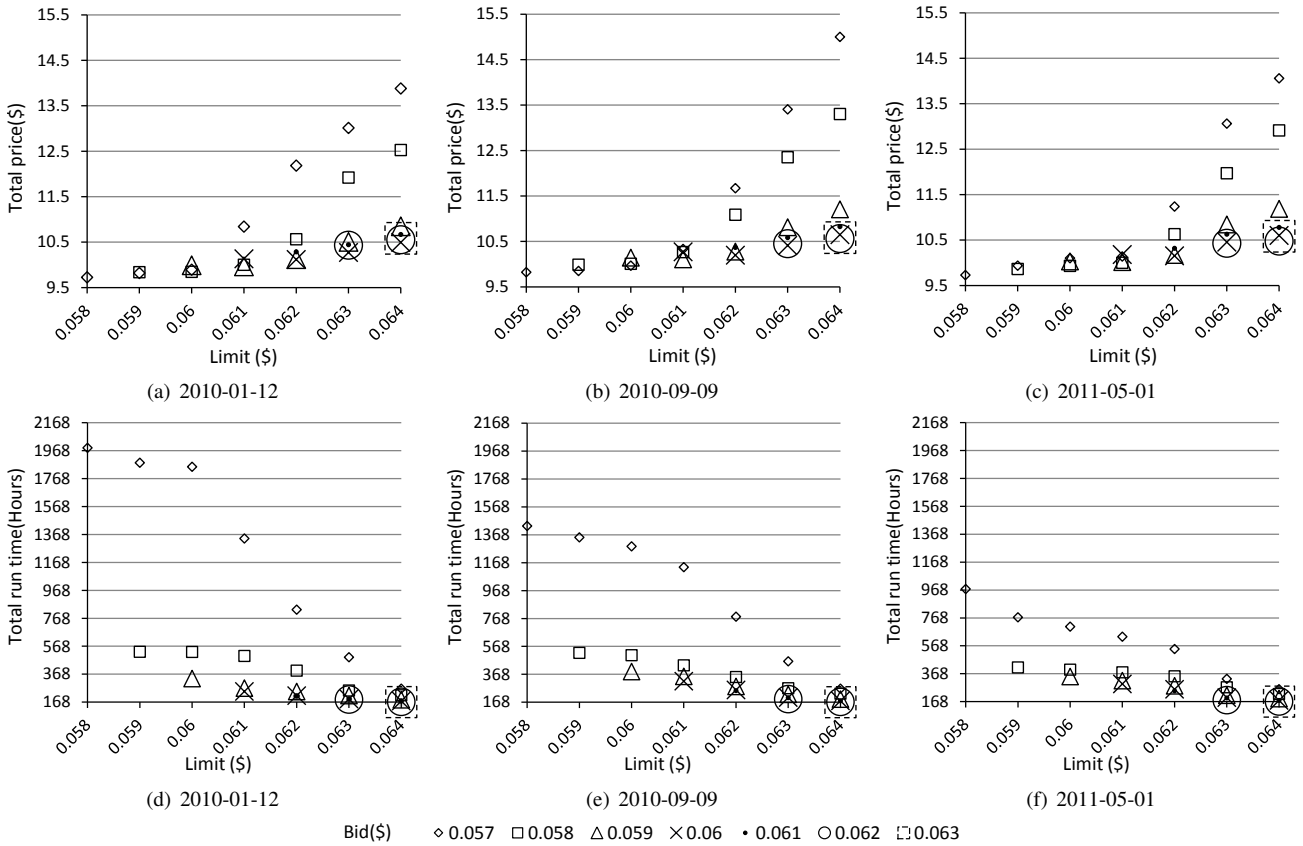


Fig. 5. Running 168 hours job on ESI with different limit prices

V. ESIs' INFLUENCE ON THE OTHER SERVICES' PERFORMANCE

In the following section, we study the influence of the ESIs on the other hosted instances (i.e., *On-Demand* and *Reserved* instances). As shown in Section IV-C, when the provider is overloaded, ESIs users may pay more money for fewer resources. It is a strong reason for ESIs users to terminate their instances, which free more resources at provider side. However, the provider should be aware of the users who choose high limit values or never attempt to terminate ESIs according to misunderstanding of ESIs concept.

In our analysis, we assume that *On-Demand* instances are hosted together with ESIs on Xen Hypervisor running *Credit Scheduler*. We started by running one *On-Demand* instance with n ESIs to understand the influence of ESIs on the performance of *On-Demand* instances. All instances are running two virtual cores. The workload is the CPU-intensive workload described in Section IV. *On-Demand* instances run with full capacity. However, ESIs' capacity is started with full capacity (i.e., 100%), then is reduced 10 percent with each step. The throughput of the *On-Demand* instance is measured with each reduction in the capacity.

As shown in Fig. 6, the *On-Demand* instance throughput is a function of both the number of ESIs and the capacity of each instance. By analyzing the curves in Fig. 6, we can notice three cases of *On-Demand* instance throughput: First, no throughput degradation. Second, throughput as a function of number of co-located ESIs only. Third, throughput as a function of number and capacity of co-located ESIs.

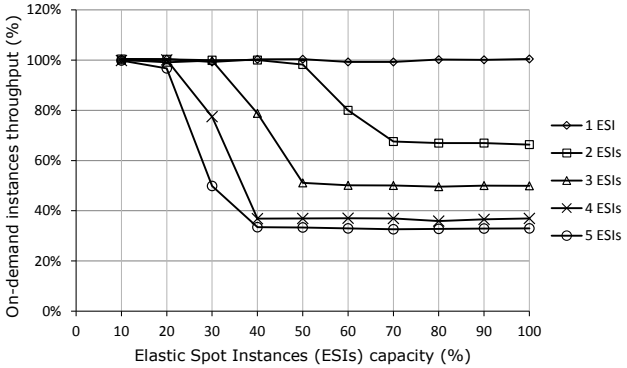


Fig. 6. *On-Demand* instances utilization with variant number of ESIs

To generalize the relation, assume that we are running n ESIs on the same host with m *On-Demand* instances. We would like to determine the capacity of the ESIs that reduce the influence of the ESI on the other hosted instances (i.e., *On-Demand* instance in our example). To formalize the models shown in Fig. 6, we define the following parameters:

- Free capacity on the host: C^h_{free}
- Number of *On-Demand* instances: n

- Requested capacity by *On-Demand* instance: C^d_{req}
- Assigned capacity to *On-Demand* instance: $C^d_{assigned}$
- Number of ESIs: m
- Requested capacity by ESI i: C^{si}_{req}
- Total requested capacity by ESIs: $\sum_{i=0}^{m} C^{si}_{req}$
- Assigned capacity to ESI i: $C^{si}_{assigned}$
- Total assigned capacity to ESIs: $\sum_{i=0}^{m} C^{si}_{assigned}$

In our case, we consider that the *On-Demand* instance will consume the full capacity of the CPU, so we will consider $C^d_{req} = 100$. According to our observations, we consider that a provider hosts VMs instances of the same size on one physical host. Therefore, the number of virtual cores is the same for both *On-Demand* and ESIs. In the following analysis, to isolate the other *On-Demand* instances influence, we consider hosting only one *On-Demand* instance on the physical host. However, the ESIs' impact will be the same for each *On-Demand* instance hosted on the same physical server. To calculate the allocated capacity for the *On-Demand* instance, we consider the following cases:

Case 1: The host is able to fulfill *On-Demand* instance required capacity while

$$(C^h_{free} - C^d_{req}) >= \sum_{i=0}^{m} C^{si}_{req} \quad (2)$$

In this case, the ESIs do not influence the *On-Demand* instance performance, and the *On-Demand* instance throughput is very close to 100%.

Case 2: The ESIs requested capacity is high to a level that influences the *On-Demand* instance's assigned capacity. However, the average requested capacity by an ESI is still lower than the requested capacity by *On-Demand* instance; this could be formulated as the following:

$$((C^h_{free} - \sum_{i=0}^{m} C^{si}_{req}) > C^d_{req}) \&\& ((\sum_{i=0}^{m} C^{si}_{req})/m < C^d_{req}) \quad (3)$$

In this case the *On-Demand* instance throughput is calculated by (1). Where $x = C^d_{assigned} = (C^h_{free} - \sum_{i=0}^{m} C^{si}_{req})$

Case 3: The ESIs requested capacity is very high. Moreover, the average requested capacity by an ESI is higher than or equal to the requested capacity by an *On-Demand* instance. In this case, the *Credit Scheduler* will employ its fairness to give the same capacity for each running instance on the hypervisor. In this case, the *On-Demand* instance throughput is calculated by (1), where $x = C^h_{total}/(n+m)$ and $n = 1$ in our example.

A region overloading will be reflected as high increase in the SIs' price, probably double the price of an On-Demand instance. In such a case, the ESIs should be scaled down as described in Algorithm 1 to satisfy the condition at (2). For example, if one *On-Demand* instance with two virtual cores running on the same host with 5 ESIs each with 2 cores, limiting the ESI's capacity to 20% will isolate any influence of the SIs. However, to prevent negative values of capacities in case of very high increase in the prices, we determine a static limit that cannot be exceeded. In our experiment, it was 10% of the CPU capacity. This value implies that ESIs will not influence the other hosted instances until the number of ESIs exceeds 20 instances on the same physical host.

Finally, we should remind that a very high price and low allocated capacity at overloaded time is a good reason for the customers of ESIs to have a checkpoint or/and turn off instances safely, which reduces ESIs's number and consequently reduces their influence.

VI. Related work

In this section, we classify the research towards improving the trades off between the total price, the reliability, and the total run time of SIs into two categories: first, work directed to find the best bid prices by analyzing and modeling prices history statistically (i.e., modeling price history). Second, work directed to manage SIs interruption by using fault tolerant architectures (i.e., managing the interruption). Moreover, some researches from the second category integrate history analysis techniques with fault tolerant architectures for a better performance.

A. Modeling price history

Javadi et al. [1] analyzed SIs history in terms of Spot price, *inter-price* time, but not the user bid. Andrzejak et al. [6] have proposed probabilistic decision model that considers user bid, budget, and the job deadline. The proposed model can suggest a bid value that meets a given budget or a deadline considering a specified level of confidence. Zhang et al. [9] adopted an auto-regressive model (AR) that depends on the historical values of demand to predict the next price of an instance type.

However, actual demand can neither be disclosed by the provider nor be inferred from the current price. It has been shown by Mazzucco et al. [22] that there is no correlation between SI prices and the time. Moreover, as shown in Fig. 1, the artifact changes in the SIs price make it difficult to build consistent models that describe the SIs' market behavior for the long run.

B. Managing interruption

Although MapReduce is designed as a fault tolerant architecture, it cannot tolerate a potential massive failure of instances in the SI's market. Therefore, Liu [23] extended the Cloud MapReduce (CMR) [24] implementation of map phase to stream intermediate results to a Cloud storage (i.e., SimpleDB). Their MapReduce implementation supports partial commit to keep track of the map process. In case of failure, system is able to determine the location at which the next map task should resume processing. Mattess et al. [25] examined many polices to run a Grid workload from DAS-2 [26]. Their local cluster is integrated with SI's market to cope with workload spikes, which results in reduction in the prices without degrading the performance.

Taifi et al. [27] studied running high performance computing (HPC) applications on SIs environment. To this end, they proposed SpotMPI architecture. One of SpotMPI architecture component is checkpoint-restart (CPR) calculator. Depending

 ISBN: 978-1-61208-216-5

on price history and the estimated total processing times, the CPR component determines the best checkpointing intervals. Jain et al. [28] developed an algorithm that dynamically adapts the resource allocation policy, which decides between *On-Demand* or SIs allocation. The allocation policy is adapted by learning from system performance on prior job execution while incorporating history of Spot prices and workload characteristics. However, the uncertainty of job time estimation is one of the problems that could approach [27] and [28] algorithms.

Yi et al. [7] employed checkpointing and migration as fault tolerance techniques. They examined many checkpointing strategies on the light of normalized $Price \times Time$ for different bid values and different types of instances. Moreover, after each instance's interruption, their approach decides the new SI's type, location, and price that reduces the total running time. However, as seen in Section IV-C, our approach showed outperforming results compared with their approach.

In addition to checkpointing and migration techniques, Voorsluys et al. [11] integrated job duplication technique. This integration increases the probability that jobs finish within their deadlines. However, as concluded by the authors, job duplication yields much higher costs.

VII. Conclusion and Future work

The proposed ESIs architecture does not require many modifications to the current Cloud Computing Infrastructure. However, it has benefits for both of the provider and the customer. On the provider side, our approach increases the provider's revenue where it eliminates the concept of the partial hours. For the customer, the proposed approach boosts the checkpointing strategy to the optimal level. However, clients' applications should be aware of our proposed bidding strategies.

We evaluated our approach against CPU-intensive applications where the CPU is the real player in power consumption. However, in the future, we will consider other resources and different combinations of the real workload. We will implement the techniques that work on I/O isolation like SEDF-DC. Furthermore, our extended work will include evaluating ESIs approach with the other SIs types.

References

[1] B. Javadi and R. Buyya, "Comprehensive Statistical Analysis and Modeling of Spot Instances in Public Cloud Environments," The University of Melbourne, Melbourne, Tech. Rep., 2011.

[2] R. Lämmel, "Google's MapReduce programming model; Revisited," *Sci. Comput. Program.*, vol. 68, no. 3, pp. 208–237, Oct. 2007.

[3] Amazon, "Amazon EC2 Spot Instances." [Online]. Available: http://aws.amazon.com/ec2/spot-instances/, Retrieved: May, 2012

[4] E. Park, B. Egger, and J. Lee, "Fast and space-efficient virtual machine checkpointing," in *Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, ser. VEE '11. New York, NY, USA: ACM, 2011, pp. 75–86.

[5] S. Yi, D. Kondo, and A. Andrzejak, "Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud," in *2010 IEEE 3rd International Conference on Cloud Computing*. IEEE, Jul. 2010, pp. 236–243.

[6] A. Andrzejak, D. Kondo, and S. Yi, "Decision Model for Cloud Computing under SLA Constraints," in *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, Aug. 2010, pp. 257–266.

[7] S. Yi, A. Andrzejak, and D. Kondo, "Monetary Cost-Aware Checkpointing and Migration on Amazon Cloud Spot Instances," *IEEE Transactions on Services Computing*, Jul. 2011.

[8] Amazon, "Amazon EC2." [Online]. Available: http://aws.amazon.com/ec2/, Retrieved: May, 2012

[9] Q. Zhang, E. Gürses, R. Boutaba, and J. Xiao, "Dynamic resource allocation for spot markets in clouds," p. 1, Mar. 2011.

[10] O. A. Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and T. Dan, "Deconstructing Amazon EC2 Spot Instance Pricing," Technion Israel Institute of Technology, Haifa, Tech. Rep., 2011.

[11] W. Voorsluys and R. Buyya, "Reliable Provisioning of Spot Instances for Compute-intensive Applications," *Computing Research Repository*, vol. abs/1110.5, 2011.

[12] H. Kim, H. Lim, J. Jeong, H. Jo, and J. Lee, "Task-aware virtual machine scheduling for I/O performance." in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, ser. VEE '09. New York, NY, USA: ACM, 2009, pp. 101–110.

[13] J. N. Matthews, W. Hu, M. Hapuarachchi, T. Deshane, D. Dimatos, G. Hamilton, M. McCabe, and J. Owens, "Quantifying the performance isolation properties of virtualization systems," in *Proceedings of the 2007 workshop on Experimental computer science*, ser. ExpCS '07. New York, NY, USA: ACM, 2007.

[14] J. Liu, W. Huang, B. Abali, and D. K. Panda, "High performance VMM-bypass I/O in virtual machines," in *Proceedings of the annual conference on USENIX '06 Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2006, p. 3.

[15] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, *Xen and the art of virtualization*. New York, New York, USA: ACM Press, Oct. 2003, vol. 37, no. 5.

[16] L. Cherkasova, D. Gupta, and A. Vahdat, "Comparison of the three CPU schedulers in Xen," *SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 2, pp. 42–51, Sep. 2007.

[17] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforcing performance isolation across virtual machines in Xen," in *Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, ser. Middleware '06. New York, NY, USA: Springer-Verlag New York, Inc., 2006, pp. 342–362.

[18] C. A. Waldspurger, "Memory resource management in VMware ESX server," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 181–194, Dec. 2002.

[19] T. Lossen, "Cloud exchange." [Online]. Available: http://cloudexchange.org/, Retrieved: May, 2012

[20] NASA, "NAS Parallel Benchmarks (NPB)." [Online]. Available: http://www.nas.nasa.gov/Resources/Software/npb.html, Retrieved: May, 2012

[21] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Resource Leasing and the Art of Suspending Virtual Machines," *High Performance Computing and Communications, 10th IEEE International Conference on*, vol. 0, pp. 59–68, 2009.

[22] M. Mazzucco and M. Dumas, "Achieving Performance and Availability Guarantees with Spot Instances," in *13th International Conference on High Performance Computing and Communications (HPCC-2011)*, Banff (Canada).

[23] H. Liu, "Cutting MapReduce Cost with Spot Market," in *USENIX HotCloud'11*, Portland, USA, 2011, p. 5.

[24] H. Liu and D. Orban, "Cloud MapReduce: A MapReduce Implementation on Top of a Cloud Operating System," *Cluster Computing and the Grid, IEEE International Symposium on*, vol. 0, pp. 464–474, 2011.

[25] M. Mattess, C. Vecchiola, and R. Buyya, *Managing Peak Loads by Leasing Cloud Infrastructure Services from a Spot Market*. IEEE, Sep. 2010.

[26] Das-2, "The Distributed ASCI Supercomputer 2 (DAS-2)." [Online]. Available: http://www.cs.vu.nl/das2/, Retrieved: May, 2012

[27] M. Taifi, J. Shi, and A. Khreishah, "SpotMPI: A Framework for Auction-Based HPC Computing Using Amazon Spot Instances," in *Algorithms and Architectures for Parallel Processing*, ser. Lecture Notes in Computer Science, Y. Xiang, A. Cuzzocrea, M. Hobbs, and W. Zhou, Eds. Springer Berlin / Heidelberg, 2011, vol. 7017, pp. 109–120.

[28] N. Jain, I. Menache, and O. Shamir, "On-demand or Spot? Learning-Based Resource Allocation for Delay-Tolerant Batch Computing," in *Microsoft Research*, 2011, p. 10.

 ISBN: 978-1-61208-216-5

Enabling the Deployment of Virtual Clusters on the VCOC Experiment of the BonFIRE Federated Cloud

Raul Valin, Luis M. Carril, J. Carlos Mouriño, Carmen Cotelo, Andrés Gómez, and Carlos Fernández
Supercomputing Centre of Galicia (CESGA)
Santiago de Compostela, Spain
Email: rvalin,lmcarril,jmourino,carmen,agomez,carlosf@cesga.es

***Abstract*—The BonFIRE project has developed a federated cloud that supports experimentation and testing of innovative scenarios from the Internet of Services research community. Virtual Clusters on federated Cloud sites (VCOC) is one of the supported experiments of the BonFIRE Project whose main objective is to evaluate the feasibility of using multiple Cloud environments to deploy services which need the allocation of a large pool of CPUs or virtual machines to a single user (as High Throughput Computing or High Performance Computing). In this work, we describe the experiment agent, a tool developed on the VCOC experiment to facilitate the automatic deployment and monitoring of virtual clusters on the BonFIRE federated cloud. This tool was employed in the presented work to analyse the deployment time of all possible combinations between the available storage images and instance types on two sites that belong to the BonFIRE federated cloud. The obtained results have allowed us to study the impact of allocating different requests on the deployment time of a virtual machine, showing that the deployment time of VM instances depends on their characteristics and the physical infrastructure of each site.**

Keywords-Cloud computing; Federated clouds; Virtualization; Cloud platforms; Virtual clusters; IaaS; SaaS.

I. INTRODUCTION

The BonFIRE Project [1] supports experimentation and testing of innovative scenarios from the Internet of Services research community, specifically focused on the convergence of services and networks. BonFIRE operates a Cloud facility based on an Infrastructure as a Service delivery model with guidelines, policies and best practices for experimentation. A federated multi-platform approach is adopted, providing interconnection and interoperation between novel service and networking testbeds. BonFIRE currently comprises of 6 geographically distributed testbeds across Europe, which offer heterogeneous Cloud resources, including compute, storage and networking. Each testbed can be accessed seamlessly with a single experiment descriptor, using the BonFIRE API that is based on the Open Cloud Computing Interface (OCCI). Figure 1 shows details about resource offering on the different testbeds, which include on-demand resources.

The BonFIRE project is also studying the possible federation of the BonFIRE testbeds with a variety of external cloud facilities, such as those provided by FEDERICA or OpenCirrus. BonFIRE offers an experimenter control of available resources. It supports dynamically creating, updating, reading and deleting resources throughout the lifetime of an experiment. Compute resources can be configured with application-specific contextualisation information that can provide important configuration information to the virtual machine (VM); this information is available to software applications after the machine is started. BonFIRE also supports elasticity within an experiment, i.e., dynamically create, update and destroy resources from a running node of the experiment, including cross-testbed elasticity.

INRIA currently offers on-request compute resources in BonFIRE, allowing experimenters to reserve large quantities of physical hardware (162 nodes/1800 cores available). This gives experimenters flexibility to perform large-scale experimentation, as well as providing greater control of the experiment variables as exclusive access to the physical hosts is possible. Further control of network performance between testbeds is anticipated through future interconnection with Federica and GÉANT AutoBAHN. BonFIRE gives you control of your experiment, which is treated as a concrete entity in BonFIRE to manage your resources.

Some additional features implemented on the BonFIRE project are:

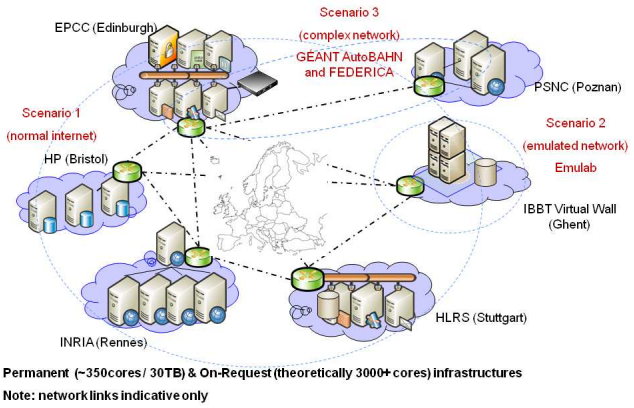


Figure 1. Representation of the BonFIRE infrastructure. Image obtained from [1].

 ISBN: 978-1-61208-216-5

- Saving compute disk images with your personal software stack, as well as storage resources.
- Sharing saved compute and storage resources.
- Sharing access to experiments with colleagues.
- Repeating experiments and sharing experiment descriptions for others to set up.
- Aggregated monitoring metrics at both resource level (e.g., CPU usage, packet delay, etc.) and application level for your VMs.
- Aggregated monitoring metrics at infrastructure level at selected testbeds.

Virtual Clusters on federated sites (VCOC) is one of the supported experiments of the BonFIRE Project [2]. Its main objective is to evaluate the feasibility of using multiple Cloud environments to deploy services that need the allocation of a large pool of CPUs or virtual machines to a single user (as High Throughput Computing or High Performance Computing). This experiment considers the deployment of virtual clusters with the propose of executing a radiotherapy application, developed in the eIMRT project [3], which calculates the dose for radiotherapy treatments based on Monte Carlo methods.

The VCOC experiment tried to answer different questions related to the usage of virtual clusters in distributed Cloud environments, analysing the advantages of deploying virtual clusters in a federated cloud. As part of the VCOC experiment a set of experiments have been proposed related to the time that the deployment and enlargement of a virtual cluster need to be operational as well as the influence that other simultaneous operations have on the process. The final objective is to get a better understanding about how to manage these virtual clusters to guarantee a reasonable time to solution or latency. A set of application probes have been chosen and they will help us to study the elasticity. The information provided by these probes will be used to monitor the performance of the application and to trigger the change in the size of the cluster. This information will be also combined with the information provided by the monitoring tools of BonFIRE.

The results and data acquired during the VCOC experiment should permit to develop policies and business rules to include in the applications under development at the institution which use the Software as a Service model.

In this paper, we introduce a description of the experiment agent developed by the VCOC experiment to manage the deployment of virtual clusters to the BonFIRE federated cloud as well as the required time to deploy individual instances with different configurations and images. The structure of the paper is as follows: In section II we describe the experiment agent and its main functionalities. A description of the error and log manager implementation is also shown in this section. The required time to deploy individual instances of the available virtual machine images in BonFIRE infrastructure is shown in Section III. Finally, the main conclusions of the paper are drawn in Section IV.

II. Description of the experiment agent

The BonFIRE project provides several ways to submit an experiment depending on the user preferences.

- The BonFIRE Portal (GUI) in a step-by-step manner [4].
- A command line client tool, such as Restfully, to interact with BonFIRE [5].
- A script for your experiment deployment, which can be automatically executed by, e.g., Restfully.
- The BonFIRE experiment descriptor, currently based on JavaScript Object Notation (JSON) [6].
- Raw HTTP commands via cURL [7].

In the VCOC experiment, we have developed an experiment agent to deploy, control and monitor the deployed experiments through a single interface. This agent was developed in Python and communicates with the experiment manager API using the httplib2 [8] library. Experiments are described in a JSON file specifying the necessary resources of the virtual clusters. After the submission of the experiment, the XML response is processed to obtain the basic information of the deployed resources. This information enables us to monitor the status of each virtual compute node and it is saved into a local file for future analysis. The main functionalities of the experiment manager are as follow.

- Experiment controller, controls the repetition of the experiments and communicates with the Experiment Manager using the API.
- Experiment status, measures accurately the time for each step in the work–flow and store this information.
- Experiment failures, detects the fail of an experiment and its resubmission.
- Experiment descriptors, accepts an experiment description in JSON and the number of repetitions as input.
- Multi-experiment, supports the sequential execution of several experiment descriptions.
- Simultaneous experiment, controls several experiment descriptions simultaneously.
- Random Experiment, generates random deployment requests which follow a defined time pattern to introduce load on the infrastructure.
- Sequential experiments, executes and controls two experiment descriptions with a predefined delay between them.
- Experiment accounting, records the BonFIRE accounting units for each experiment description.

A work–flow diagram of the experiment agent is depicted in Figure 2. This figure shows three main levels after a experiment submission. First, we evaluate the status of the experiment until the experiment has been deployed; this means the JSON experiment description has been processed

 ISBN: 978-1-61208-216-5

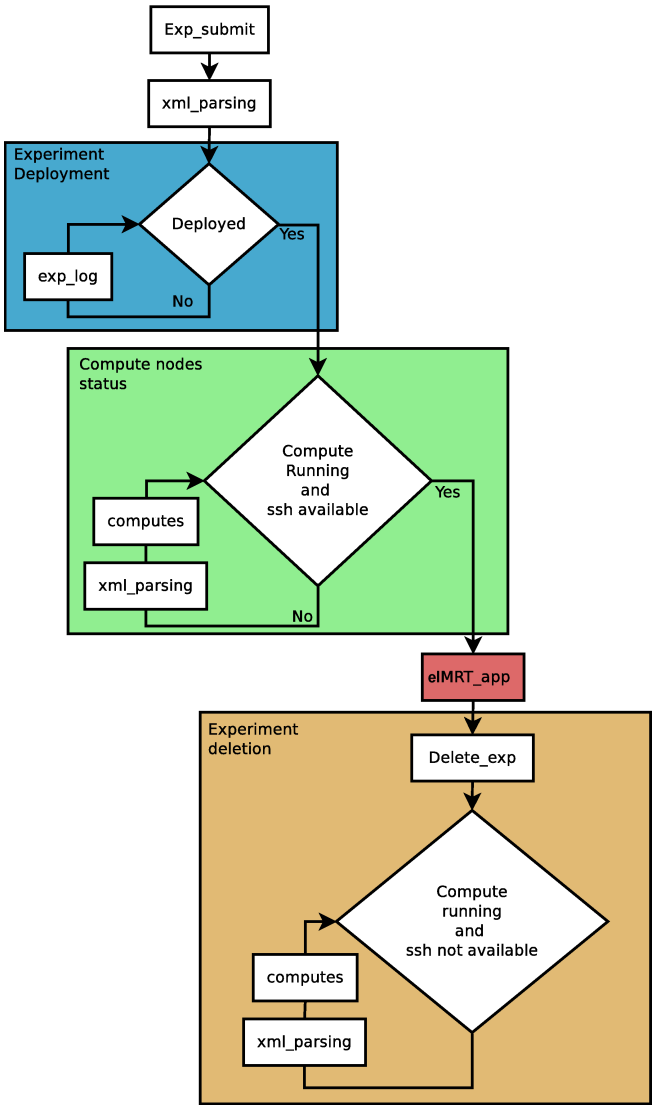


Figure 2. Workflow diagram of the experiment agent.

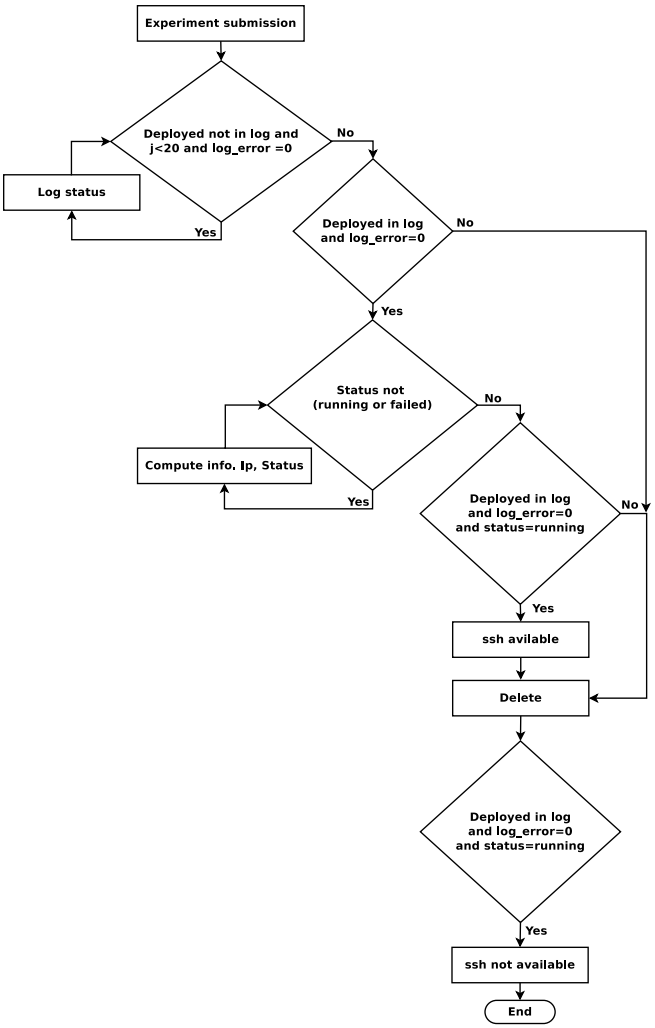


Figure 3. Workflow diagram of the error management implemented in the experiment agent.

by the Broker interface. After that, we have to wait for the allocation of the requested resources in the Broker layer of the infrastructure. This is the second level of the experiment agent, which evaluates if the required resources are running and ssh–available. When the requested resources are ssh–available, the eIMRT application will be executed. The last level of the experiment submission is the experiment deletion, which is carried out when the eIMRT application has finished. The experiment agent considers the deletion completed when the experiment is not ssh–available and the deleted resources reached the status DONE.

Two important functionalities implemented on the experiment agent are the error management and the measurement of the required time to complete each level described above. The error management has been developed to perform unattended deployment of experiments, taking into account possible errors or delays during the deployment of the experiment. Figure 3 shows the flow diagram of the error management that has been implemented in the experiment agent. First, after the experiment submission we have to evaluate if the experiment has been correctly deployed or any error has occurred. The experiment is deployed when the log status information provided by BonFIRE, after checking the experiment definition, returns the status deployed. If some error exists, the experiment will be resubmitted again. Otherwise, if there are not errors the experiment agent will evaluate the status of the requested VMs until they are ssh–available. When the VMs are available, the eIMRT application is executed and the experiment is destroyed when the eIMRT application is completed. are not ssh–available.

The measured times implemented on the experiment agent script try to provide information about the necessary time to deploy/destroy the computational resources available from

 ISBN: 978-1-61208-216-5

the submission/destruction of the experiments. Therefore, the experiment agent saves a timestamp value when the experiment is submitted and after its submission, a new timestamp value is saved when the computational resources are ssh–available. The difference between these two timestamp values provides the necessary time to deploy the resources requested on the JSON file. From the obtained times a first idea of the quality of service can be sketched if we want to use the infrastructure as a service. Finally, the experiment agent also measures the necessary time to destroy the experiment from the destruction request until the VMs are not ssh-available and the deleted resources reached the status DONE.

The last functionality implemented on the experiment agent, that we think must be highlighted in this description, is the possibility of deploying random experiments which follow a predefined pattern obtained from the accounting system of the Finisterrae supercomputer hosted in CESGA [9]. This functionality has been developed to introduce random noise into the BonFIRE infrastructure when we have exclusive access and therefore, we need to evaluate the impact of scheduling new experiments simultaneously.

This functionality returns a histogram with a discrete experiment submission probability such as is depicted in Figure 4, where the experiment submission probability for each 30 minutes of a day of the week is represented. Therefore, if we want to deploy random experiments during one day each 30 minutes at the same time that we are deploying our experiments, we will need to indicate the desired number of random experiments that they will be deployed during one day and the experiment agent will distribute the desired number of experiments taking into account the probability distribution. Figure 5 shows an example of the distribution of 100 random experiments taking into account the histogram depicted in Figure 4.

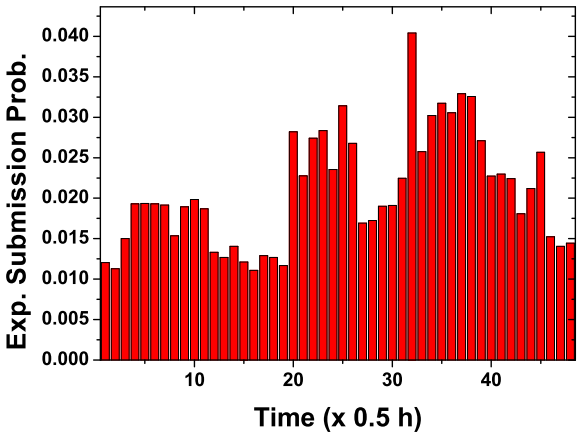


Figure 4. Experiment submission probability each 30 minutes of a day of the week.

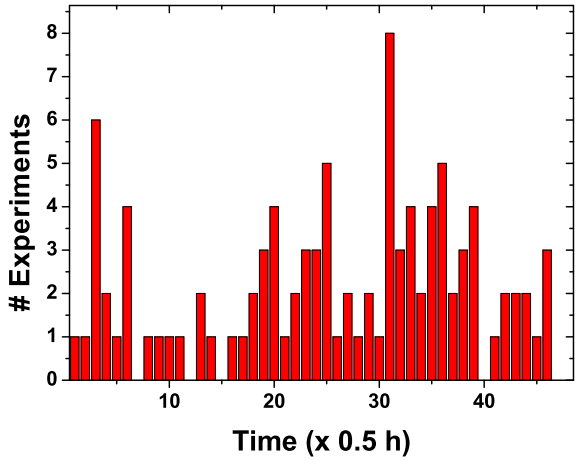


Figure 5. Example of the distribution of 100 random experiments taking into account the histogram depicted in Figure 4.

III. Results of the deployment time on EPCC and INRIA BonFIRE sites

The BonFIRE federated cloud has predefined resources, storage images, instances and networks, which are available for users. Table I and Table II show the resources, virtual machine images and instance types, available on EPCC and INRIA BonFIRE sites. Furthermore, two network resources are also available in these two sites enabling us choosing between either an Internet connection or a WAN connection. In this work, we have studied the deployment time of all possible combinations between the available storage images and instance types on EPCC and INRIA BonFIRE sites. The main goal of this study is to analyse the impact on the deployment time of allocating different requests.

The methodology adopted to carry out the experiment was based on the deployment on each site of each combination of storage-instance type. Each one of these combinations was deployed 10 times using the experiment agent. The final value to calculate the deployment time is equal to the mean of the 10 values provided by the experiment agent from the experiment submission until the VM is ssh-available, such as it was described in Section II.

Figure 6 depicts the deployment time for each storage-instance type combination on EPCC BonFIRE site. The obtained results show that DebSqV3 and DebSq2GV3 images have similar deployment times between 50-100 seconds, independently of the instance type. The deployment time increases in a rate similar to the size of disk image of the VM, represented on the figure by a red line. Therefore, the 10 GB storage image has the largest deployment time higher than 300 seconds independently of the instance type.

The deployment time for each storage-instance type com-

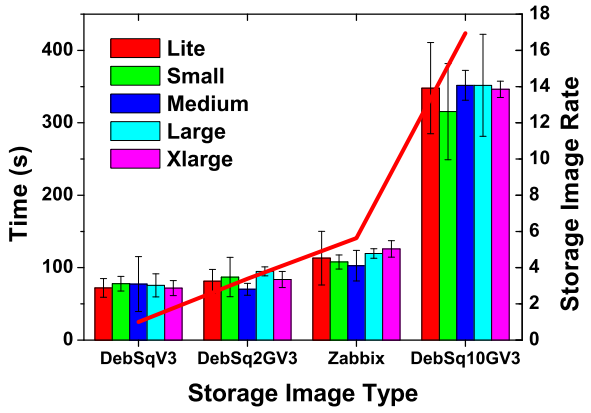


Figure 6. Deployment time for each storage-instance type combination on EPCC BonFIRE site. The size rate of the storage images with respect to the DebSqV3 image (red line) is also depicted for comparison reasons.

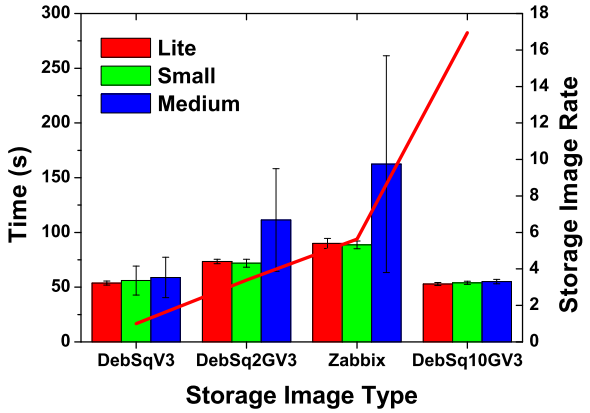


Figure 7. Deployment time for each storage-instance type combination on INRIA BonFIRE site. The size rate of the storage images with respect to the DebSqV3 image (red line) is also depicted for comparison reasons.

bination on INRIA BonFIRE site is depicted in Figure 7. The deployment time of DebSqV3 storage images on INRIA is around 50 seconds, independently of the instance type. For DebSq2GV3 and Zabbix storage images, the deployment time depends on the instance type with larger values for Medium instances around 100 seconds for DebSq2GV3 and 150 seconds for Zabbix. However, similar values, slightly higher than 50, can be observed for Lite and Small instances with both storage images.

A comparison between these results with the previous ones shown in Figure 6, highlights that the deployment time of VMs on INRIA is almost independent on the storage image and its dependence is higher with the instance type. This situation is remarkable for the DebSq10GV3 storage image. The main reason of this difference on the behaviour of deployment time between EPCC and INRIA is due to the configuration of the physical infrastructure. Both sites base its cloud infrastructure on the OpenNebula [10] platform but EPCC copies the storage images to the compute node using NFS. However, INRIA has implemented a system that makes a snapshot of the storage image after its first copy to the compute node via NFS. Therefore, one VM can be deployed faster on INRIA than EPCC if a snapshot of the storage image exists on the compute node.

Storage	Disk Size (MB)	Description
Zabbix	3400	Debian Squeeze with Zabbix monitoring
DebSqV3	604	Debian Squeeze
DebSq2GV3	2048	Debian Squeeze
DebSq10GV3	10240	Debian Squeeze

Table I
STORAGE RESOURCES AVAILABLE ON EPCC AND INRIA BONFIRE SITES.

Instance		vcpu	vmemory (MB)
EPCC	INRIA		
Lite	Lite	0.5	256
Small	Small	1	1024
Medium	Medium	2	2048
Large		2	4096
Xlarge		4	8192

Table II
INSTANCE TYPES AVAILABLE ON EPCC AND INRIA BONFIRE SITES.

Other difference, which rises from the comparison between sites, is the dependence with the instance type. Results on EPCC show that the deployment time is almost independent of the instance type however, medium instances on INRIA have larger deployment times. The origin of this difference is the way we are measuring the times, we do not have exclusive access to the infrastructure and the measured time also includes the scheduling time. Therefore, depending on the computational load of physical resources is possible to observe delays when instances with high computational requirements are requested.

These results rise several questions about the best way to design a quality of service model on federated clouds, since in some cases could not be possible to guarantee the same configuration among sites or the same overload of the infrastructure when users can choose the location of their experiments [11]. From a global point of view, a scheduling system with information about the load of the physical resources of each site belonging to the federated cloud may provide better allocation of the requested virtual machines. Furthermore, this scheduler might take into account other factors such as configuration differences of each site that can affect the deployment time of virtual resources [12].

 ISBN: 978-1-61208-216-5

IV. Conclusion and future work

This work described the experiment agent developed on the VCOC experiment supported by the BonFIRE project in order to facilitate the automatic deployment and monitoring of virtual clusters on the BonFIRE federated cloud. The experiment agent implements several functionalities such as accept the description of virtual clusters on a JSON file, error management, recording the deployment times of virtual resources or deployment of random experiments following a predefined pattern. The first use of the experiment agent was to study the deployment time of all possible combinations between the available storage images and instance types on EPCC and INRIA BonFIRE sites. The main objective of this study was to analyse the impact on the deployment time of allocating different requests. The obtained results show that the deployment time of a VM instance is around 50 seconds and 300 seconds depending on the size of the storage image, the instance type and the deployment type. These results were obtained without an exclusive access to the infrastructure and the measured time includes the scheduling time. The obtained differences between sites rise several questions about the best way to design a quality of service model on federated clouds. For instance, it is difficult to guarantee the same configuration among sites or the computational load of the physical infrastructure when users are able to choose the experiment location. The obtained results, together with other data from other experiments will help EPCC to modify their infrastructure in order to reduce deployment times for large images.

Acknowledgment

The work leading to this publication was funded by the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 257386, "BonFIRE". The authors would like to thank Mr. Kostas Kavoussanakis and Mr. Ally Hume from EPCC, and Mr. Florian Schreiner from Fraunhofer FOKUS, for fruitful discussions about the VCOC experiment. Finally, we must highlight that this work has been possible thanks to the technical support provided by Mr. Maxence Dunnewind from INRIA, Mrs. Eilidh Troup from EPCC, Mr. Roland Kübert from University of Stuttgart and Mr. Gareth Francis from EPCC.

References

[1] "BonFIRE project." [Online]. Available: http://www.bonfire-project.eu/ [retrieved: May, 2012]

[2] "VCOC: Virtual Clusters on Federated Cloud Sites ." [Online]. Available: http://www.bonfire-project.eu/innovation/virtual-clusters-on-federated-cloud-sites [retrieved: May, 2012]

[3] L. Carril, Z. Martín-Rodríguez, C. Mouriño, A. Gómez, R. Díaz, and C. Fernández, "Advanced Computing Services for Radiotherapy Treatment Planning," in *Cloud Computing*. CRC Press, Oct. 2011, pp. 529–551. [Online]. Available: http://dx.doi.org/10.1201/b11149-27

[4] "BonFIRE portal." [Online]. Available: http://www.bonfire-project.eu/innovation/virtual-clusters-on-federated-cloud-sites [retrieved: May, 2012]

[5] *Restfully*. [Online]. Available: https://github.com/crohr/restfully [retrieved: May, 2012]

[6] "JSON." [Online]. Available: http://www.json.org/ [retrieved: May, 2012]

[7] cURL. [Online]. Available: http://curl.haxx.se/ [retrieved: May, 2012]

[8] "httplib2." [Online]. Available: http://pypi.python.org/pypi/httplib2 [retrieved: May, 2012]

[9] "Finisterrare Supercomputer." [Online]. Available: https://www.cesga.es/en/infraestructuras/computacion/finisterrae [retrieved: May, 2012]

[10] *OpenNebula*. [Online]. Available: http://opennebula.org [retrieved: May, 2012]

[11] A. Andrzejak, D. Kondo, and S. Yi, "Decision Model for Cloud Computing under SLA Constraints," in *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, Aug. 2010, pp. 257–266.

[12] R. Buyya, S. K. Garg, and R. N. Calheiros, "SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions," in *2011 International Conference on Cloud and Service Computing*. IEEE, Dec. 2011, pp. 1–10.

 ISBN: 978-1-61208-216-5

Towards a Scalable Cloud-based RDF Storage Offering a Pub/Sub Query Service

Laurent Pellegrino, Françoise Baude, Iyad Alshabani
INRIA-I3S-CNRS, University of Nice-Sophia Antipolis
2004 route des lucioles, Sophia Antipolis, France
lpellegr@inria.fr, fbaude@inria.fr, ialshaba@inria.fr

***Abstract*—Recently, Complex Event Processing engines have gained more and more interest. Their purpose consists in combining realtime and historical data in addition to knowledge bases to deduce new information. Also, RDF is now commonly used to make information machine-processable. In this short paper we propose to leverage existing research about distributed storage and realtime filtering of RDF data with the intention of helping Complex Event Processing engines to reach their goal at large scale. Towards this objective, we identify and discuss the challenges that have to be addressed for providing a solution that supports RDF data storage and a pub/sub retrieval service on a cloud-based architecture. Then, we explain how to meet these challenges through a solution based on structured Peer-to-Peer networks. Finally, we discuss the status of our ongoing work whose implementation is realized thanks to ProActive, a middleware for programming cloud-based distributed applications.**

***Index Terms*—*RDF (Resource Description Framework)*; *Publish/Subscribe*; *RDF data management*; *Cloud Computing*.**

I. Introduction

In the past years, Cloud Computing gained a great interest in academic and industrial solutions. Its goal is to provide users with more flexible services in a transparent way. All services and applications are allocated in the cloud which is an internet-scale collection of connected devices. Aside, the exponential growing of information exchanged over the internet leads to the emergence of the Semantic Web [1] whose realization is brougth into existence thanks to RDF (Resource Description Framework) [2]. RDF is a W3C standard aiming to improve the World Wide Web with machine processable semantic data. It provides a powerful data model for structured knowledge representation and is used to describe semantic relationship among data. Statements about resources are in the form of $(subject, predicate, object)$ expressions which are known as *triples* in the RDF terminology (each element of a triple is dubbed RDF term). The subject of a triple denotes the resource that the statement is about, the predicate denotes a property or a characteristic of the subject, and the object presents the value of the property. RDF is increasingly used due to its interoperability [3], its good properties in data exchange and its potential use of inferencing to contextually broaden search, retrieval and analysis.

The traditional way of querying RDF data is a blocking *get* operation. However, applications need an asynchronous query mode to be more responsive on arrival of RDF data. Publish/subscribe (pub/sub) is a messaging pattern where publishers and subscribers communicate in a loosely coupled fashion. Subscribers can express their interests in certain kinds of data by registering a subscription (continuous query) and be notified asynchronously of any information (called an *event*) generated by the publishers that matches those interests. Notifications are made possible thanks to a matching algorithm that puts in relation publications and subscriptions.

Our goal is to provide a system, deployed in a cloud environment, that stores RDF events persistently, filter and notify them as soon as they arrive. For example, Complex Event Processing (CEP) [4] systems have a need to mix realtime, past events and existing knowledge bases to deduce new patterns [5]. However, the system we envisage is not limited to the integration with CEP engines. More generally, it could be used to take advantage of its distributed storage and pub/sub layer.

This short paper identifies some challenges that have to be addressed in order to build a distributed system that combines RDF data storage and pub/sub. In Section II, we highlight some of the challenges to take up when this type of system has to be built. Section III motivates and defines our retrieval model. Section IV explains how we expect our system to meet the challenges in line with our model, and how it differs from existing systems. Section V presents our conclusions.

II. Challenges

Proposing solutions for or against the following requirements or difficulties constitutes the challenges:

1) Scalability: In our context, a scalable system must be able to support a large number of data, publications and subscriptions. This is the key property to fulfill when a distributed system is built. Also, in pub/sub systems, expressivity and scalability are closely related [6], [7]. Expressivity implies that events with different formats and semantics are supported in addition to a powerful subscription language (i.e. a subscription language that offers the possibility to consumers to subscribe precisely to the information they are interested in). But the more expressive a pub/sub system is, the more complex the matching algorithm becomes. Thus, the efficiency of the matching algorithm significantly affects both performance and scalability.

2) Fault-tolerance: Depending on the type of the application, there might be the need to ensure different level of reliability. For instance, in a financial system such as the

 ISBN: 978-1-61208-216-5

New York Stock Exchange (NYSE) or the French Air Traffic Control System, reliability is critical [8]. Ensuring a correct dissemination of events despite failures requires a particular form of resiliency. Pub/sub systems which consider event routing based on reliability requirements are rare schemes [9], which proves this challenge has not been sufficiently tackled.

3) Skewed distribution of RDF data: The frequency distribution of terms in RDF data is highly skewed [10], [11]: many triples may share the same predicate (e.g., *rdf:type*). This distribution prevents scalability from algorithms that base their partitioning on these values.

In addition to the challenges which have been previously introduced, some orthogonal properties related to QoS (Quality of Service), such as data delivery semantics, notifications ordering, security aspects, etc. constitute also major open research challenges that are naturally present in RDF-based pub/sub systems [12].

III. RETRIEVAL MODEL

Nowadays, datasets grow so large that they become awkward to work with. This idea is really well captured by the notion of big data from which knowledge acquisition has to be extracted. To make it feasible, information have to be analyzed and correlated. A solution is CEP engines that recently proposed to leverage information which stem from realtime data, contextual and past information. At high scale, a step towards this direction consists in helping CEP engines to reach their goal by providing both storage and realtime filtering of data of interest in order to minimize the amount of information they have to work with. For that, we propose a retrieval model of the stored data based on both *pull* and *push* mechanisms. The pull mode refers to one-time queries; an application formulates a query to retrieve data which have been already stored. In contrast, the push mode refers to pub/sub and is used to notify applications which register long standing queries and push back a notification each time an event that matches them occurs. Contrary to RACED [13], that also proposes a push mode, the result is not the output of a previous subscription matching but it is aimed for getting past and perennial information.

A. Data model

The data model we introduce hereafter is valid for both the pull and push retrieval modes. It is built on top of quadruples. A quadruple extends the concept of RDF triple by adding a fourth element (usually named context or graph value) to indicate or to identify the data source and the event itself. Indeed, the notion of provenance is essential when integrating data from several sources and more generally to classify data on the web. Finally, each quadruple represents a potential event that may be delivered to a subscriber but also a data that is stored.

However, the number of elements contained by an event (quadruple) is limited. To overcome this drawback, we have introduced the notion of compound event: an event that is made of a non-limited number of quadruples. Supposing that a quadruple is modeled by a 4-tuple $q = (c, s, p, o)$ and a compound event by a set $C = \{q_0, q_1, ..., q_n\}$ then each q of C shares the same context value c in order to allow to identify the quadruples that form this compound event. Moreover, thanks to this abstraction, our content-based pub/sub system can support multi-attribute values, still in compliance with RDF data model.

B. Filter model

Both retrieval modes have their filter model based on SPARQL (SPARQL Protocol and RDF Query Language) [14], another W3C specification that is usually used to retrieve and manipulate data stored in RDF format with one time queries. This language is suitable to build a very expressive filter model. Even if it could be used as a pull retrieval model, for the push retrieval model some restrictions are required (e.g. we only allow SELECT query form, a pattern applies to one graph value at a time, see below).

SPARQL provides the possibility to formulate a subscription by associating several filter constraints to a quadruple (event), but also to a set of quadruples that belong to the same compound event. This means that several events that are published at different times and that belong to a same compound event may participate to the matching of a subscription by using their common constraints.

```
1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2 SELECT ?user ?name ?age WHERE {
3     GRAPH ?g {
4         ?user foaf:name ?name .
5         ?user foaf:age ?age
6     } FILTER (?age >= 18 && ?age <= 25)
7 }
```

Listing 1. Legal SPARQL subscription indicating that only events about users whose age is between 18 and 25 have to be notified.

Listing 1 shows an example of a subscription that is used to deliver a notification each time two events that belong to the same compound event (represented by the graph pattern and its associated variable `?g`) match the constraints. This subscription depicts two types of constraints that may be uttered. The first one is a join constraint. It consists in computing an equi-join condition on the variable `?user` with the events of the same graph that match the triple patterns (a triple that may contain variables for querying unknown values), see lines 3, 4 and 5. The second type of constraints that may be formulated are filter constraints. Filter constraints are shown in the example by using the `FILTER` keyword on line 6. This second type of constraint may contain several logically related predicates.

Here, we introduced joins because unlike in traditional publish/subscribe systems [15] (where the constraints are matched for each event which is asynchronously published) we want to apply the matching on a set of events (compound event), where each event has been published independently. This condition is fundamental because our push retrieval mode is not supposed to act as a CEP engine correlating several compound events. However, our system has to handle two constraints. First, the

 ISBN: 978-1-61208-216-5

quadruples that form a compound event are not stored at the exact same time, due to the fact that the system is distributed, i.e. each quadruple can be indexed on any distributed node of the system in an asynchronous way. Second, a join constraint is limited to a set of quadruples that belong to the same compound event.

IV. Addressing These Challenges

Our aim is to provide an Internet wide system that fulfill challenges introduced in section II while respecting the model presented in section III. There are already some approaches experimenting how to store and query RDF data using popular cloud technology, but along the pull model mainly. Recently, CumulusRDF [16] proposed to rely upon the Cassandra [17] key-value store, by leveraging its two levels indexing model in order to store RDF triples. The choices they make in CumulusRDF are driven by the need to retrieve RDF data by triple patterns only and not the full expressivity of SPARQL. Their solution requires to build more than one index for each triple to be stored. Even if the lookup performances seem reasonably good, they do not support conjunctive queries (joins), nor simple or complex queries that contain some filter conditions. Also, some solutions combining Map-Reduce and distributed data storage systems (e.g., HDFS, BigTable) [18] have been proposed but they require a configuration phase, and then involve several large data sets movements. Moreover, none of the introduced solutions are well adapted when extreme scalability is expected. Indeed, Peer-to-Peer (P2P) systems have been recognized as a key communication model to build platforms for distributed applications at very large scale [19]. For that reason, our system model is based on the original idea of Content Adressable Network (CAN) [20].

A CAN is a structured P2P network (structured in opposition to unstructured, another category of P2P networks better suited to high churn, which is thus less necessary for private/public cloud environments) based on a d-dimensional Cartesian coordinate space labeled $\mathcal{D}$. This space is dynamically partitioned among all peers in the system such that each node is responsible for indexing and storing data in a *zone* of $\mathcal{D}$ thanks to a standard RDF datastore such as Jena [21]. According to our data model, we use a 4-dimensional CAN in order to associate each RDF term of a quadruple to a dimension of the CAN network. A quadruple to index is a point in a 4-dimensional space.

Distributed pub/sub systems have been extensively studied [6], [15], [22] over the last two decades. Recently, some works such as BlueDove [23] revives this field in a cloud context. However, among these works only a few are concerning pub/sub with RDF data and none are combining storage and pub/sub [24].

Most of the proposed solutions use consistent hashing to map data onto nodes for RDF pub/sub systems or RDF data storage. This means that data have to be indexed several times in order to be retrieved and be handled by the pub/sub matching algorithm. For example, CSBV [25] is a matching algorithm for RDF triples that would imply to index each quadruple 15 times in our case: one indexation for each possible combination without repetition of the RDF terms contained by a quadruple. Our approach which relies on CAN, replaces consistent-hashing by lexicographic ordering in order to use only one index and to support range queries efficiently. In return, subscriptions have to be indexed several times. However, we prefer to trade data duplicates with subscriptions duplicates due to the huge foreseen volume of data.

As for systems which use consistent-hashing, we also have to confront the challenge II-3. But in addition, due to lexicographic order, RDF terms which are lexicographically close may be handled by the same peer and unbalance the load between peers. We propose three solutions based on static and dynamic adaptation to overcome this load balancing issue. The first one simply consists in removing prefixes. Suppose that we have to index two quadruples which differs only by one RDF term. On one hand we have *http://example.org/animal* and on the other hand *http://example.org/jacket*. Also, in the network there are two peers: one managing the range $[a, e)$ and one $[e, j)$. In such a case, if we remove the prefix *http://example.org* the former data will be indexed on the first peer and the latter by the second peer. An additional solution is to have an approach similar to the one proposed in BlueDove. Event sources are aware of the type of data and the range values they will publish. Hence, it is possible to take advantage of this information to preconfigure our P2P network. For example, if we know that RDF data published are about weather in Europe and the value of the key event of the published compound events is between $[-20; 40]$, we can leverage this knowledge to increase the number of peers managing this range of values. Finally, the third solution related to elasticity and thus also tackling challenge II-1 will be to balance the load of peers by using the standard join and leave [1] operations of our P2P system. Indeed, by considering the unpredictable and fluctuating amount of information that may be produced (or removed) by any entity, the system has to be elastic. In contrary to an always-on infrastructure for which the institutions refrain to pay for, the idea is to rely on the notion of Cloud Computing to scale horizontally by adding more nodes (peers) on-demand and to release them whenever possible. But also to scale vertically by offering the possibility to deploy several peers on the machines that are underloaded.

To meet challenge II-1 we expect to develop a matching algorithm that parallelizes and balances as much as possible the matching of compound events. A few algorithms have been proposed to balance the matching but the execution to perform a join between several conjunctions is done sequentially by creating a chain [25]. To improve scalability and performances we also intend to manage burst of new subscriptions and the placement of peers according to geographic information. The former case implies to adapt the number of computing agents in charge of the matching process in each peer. The later, such as proposed in [26], consists in improving latency perceived by Internet wide-users (specially subscribers) as CEP(s) by a

[1] In the future, we wish to offer an RDF data garbage collection operation.

 ISBN: 978-1-61208-216-5

geographical mapping of P2P nodes and proxies[2] on cloud hosts.

Finally, challenge II-2 implies to take into account the replication of RDF data but also the states of the matching algorithm. In the former case, we can replicate the data by using the neighbors of a peer. However, in the latter case, it is less obvious because subscriptions do not have to be lost and duplicate notifications have to be avoided. In our system indexing a subscription (set of related patterns) ends up in duplicating it on several peers. We think about leveraging this behavior to come back into a consistent state for the pub/sub layer in case of failure. In addition we are interested to build our system such that user requested QoS properties may be easily addressed. To make it feasible we introduce configurable proxies lying out of the P2P network which will only store and compute the matching between subscriptions and publications. All messages (requests, responses, notifications, etc.) go through these proxies where they can be handled according to requested QoS properties.

V. Conclusion

In this short paper, we have identified and discussed the challenges which need to be addressed in order to build a scalable cloud based RDF storage offering a pub/sub query service. We currently have a first prototype of our extended version of CAN [27], implemented by using ProActive/GCM technology. ProActive is an asynchronous active-object based middleware offering the notion of asynchronous calls with futures (a promise to get back a response) among distributed objects, extended with the possibility to transparently handle groups of objects and security (e.g., authentication, encryption) for inter-object communications [28]. In addition it provides the notion of multi activity to handle requests concurrently. Also, thanks to ProActive abstraction, any peer or proxy needed to access the CAN network can easily be deployed on any host, be it on a private/public cloud, grid or cluster, desktop machines offering elaborated support to address firewall issues, and more generally issues that may be encountered in such a distributed infrastructure.

Acknowledgment

This work was in part supported by the EU FP7 STREP project PLAY and French ANR project SocEDA.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific American*, vol. 284, no. 5, pp. 28–37, 2001.

[2] O. Lassila and R. Swick, "Resource description framework (rdf) model and syntax," *World Wide Web Consortium, http://www. w3. org/TR/WD-rdf-syntax*, 1999.

[3] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks, "The semantic web: The roles of xml and rdf," *Internet Computing, IEEE*, vol. 4, no. 5, pp. 63–73, 2000.

[4] D. Luckham, *The power of events: an introduction to complex event processing in distributed enterprise systems*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[2]Proxies are representing publishers, subscribers and users of the pull mode.

[5] D. Anicic, P. Fodor, S. Rudolph, and N. Stojanovic, "Ep-sparql: a unified language for event processing and stream reasoning," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 635–644.

[6] A. Carzaniga, D. Rosenblum, and A. Wolf, "Achieving scalability and expressiveness in an internet-scale event notification service," in *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*. ACM, 2000, pp. 219–227.

[7] J. Wang, B. Jin, and J. Li, "An ontology-based publish/subscribe system," *Middleware 2004*, pp. 232–253, 2004.

[8] K. Birman, "A review of experiences with reliable multicast," *Software: Practice and Experience*, vol. 29, no. 9, pp. 741–774, 1999.

[9] S. Mahambre and U. Bellur, "Reliable routing of event notifications over p2p overlay routing substrate in event based middleware," in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. IEEE, 2007, pp. 1–8.

[10] S. Kotoulas, E. Oren, and F. Van Harmelen, "Mind the data skew: distributed inferencing by speeddating in elastic regions," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 531–540.

[11] A. Harth, J. Umbrich, A. Hogan, and S. Decker, "Yars2: A federated repository for querying graph structured data from the web," *The Semantic Web*, pp. 211–224, 2007.

[12] S. Mahambre, M. Kumar, and U. Bellur, "A taxonomy of qos-aware, adaptive event-dissemination middleware," *Internet Computing, IEEE*, vol. 11, no. 4, pp. 35–44, 2007.

[13] G. Cugola and A. Margara, "Raced: an adaptive middleware for complex event detection," in *Proceedings of the 8th International Workshop on Adaptive and Reflective Middleware*. ACM, 2009, p. 5.

[14] E. Prud'Hommeaux and A. Seaborne, "Sparql query language for rdf," *W3C working draft*, vol. 4, no. January, 2008.

[15] P. Pietzuch and J. Bacon, "Hermes: A distributed event-based middleware architecture," in *Distributed Computing Systems Workshops*. IEEE, 2002, pp. 611–618.

[16] G. Ladwig and A. Harth, "Cumulusrdf: Linked data management on nested key-value stores," in *The 7th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2011)*, 2011, p. 30.

[17] A. Lakshman and P. Malik, "Cassandra: A structured storage system on a p2p network," in *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*. ACM, 2009, pp. 47–47.

[18] J. Sun and Q. Jin, "Scalable rdf store based on hbase and mapreduce," in *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, vol. 1. IEEE, 2010, pp. V1–633.

[19] M. Jelasity and A. Kermarrec, "Ordered slicing of very large-scale overlay networks," in *Peer-to-Peer Computing, 2006. P2P 2006. Sixth IEEE International Conference on*. IEEE, 2006, pp. 117–124.

[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 161–172, 2001.

[21] J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: implementing the semantic web recommendations," in *World Wide Web conference*. ACM, 2004, pp. 74–83.

[22] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, 2003.

[23] M. Li, F. Ye, M. Kim, H. Chen, and H. Lei, "A scalable and elastic publish/subscribe service," in *Parallel & Distributed Processing Symposium (IPDPS), IEEE International*. IEEE, 2011, pp. 1254–1265.

[24] I. Filali, F. Bongiovanni, F. Huet, and F. Baude, "A survey of structured p2p systems for rdf data storage and retrieval," *Transactions on Large-Scale Data-and Knowledge-Centered Systems III*, pp. 20–55, 2011.

[25] E. Liarou, S. Idreos, and M. Koubarakis, "Continuous rdf query processing over dhts," in *Proceedings of the 6th international semantic web conference*. Springer-Verlag, 2007, pp. 324–339.

[26] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated data placement for geo-distributed cloud services," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation*. USENIX Association, 2010, pp. 2–2.

[27] F. Baude, F. Bongiovanni, L. Pellegrino, and V. Quema. (2011) D2.1 requirements eventcloud. Project Deliverable PLAY. [Online]. Available: http://play-project.eu/documents/viewdownload/3/20

[28] L. Baduel, F. Baude, D. Caromel, A. Contes, F. Huet, M. Morel, and R. Quilici, "Programming, composing, deploying for the grid," *GRID COMPUTING: Software Environments and Tools*, pp. 205–229, 2006.

 ISBN: 978-1-61208-216-5

Datanode Optimization in Distributed Storage Systems

Xiaokang Fan, Shanshan Li, Xiangke Liao, Lei Wang, Chenlin Huang, Jun Ma
School of Computer Science and Technology
National University of Defense Technology
Changsha, Hunan, P.R.China

{fanxiaokang, shanshanli, xkliao, wanglei, huangchenlin, majun}@nudt.edu.cn

***Abstract*—Distributed storage systems designed for small files have been developing rapidly, like Facebook's hayStack, Twitter's Cassandra and so on. But, under our observation, there are still some drawbacks in these systems. For example, they do not have cache specified for files and have not taken the relationship inherent in application-specific knowledge between files into consideration. We propose a file-level cache on datanode and co-location of affinitive files based on application-specific knowledge. We use a synthetic data set and a real world trace to evaluate our optimization. The file-level cache and co-location of affinitive files together can improve system's throughput by 20%-50%.**

Keywords-distributed storage system; file-level cache; co-location.

I. INTRODUCTION

Distributed storage systems have been widely used in large datacenters. These systems are designed to provide efficient, reliable access of data using clusters of commodity hardware [22]. So far, applications specified for small files have been increasing rapidly. For example, micro blogging, facebook, twitter, and so on. These applications generate enormous amounts of small files to store in the storage systems. For example, Facebook have stored over 260 billion images and more than 20 petabytes of data so far [11]. Many systems have been developed to support these applications, like fastDFS [18], Facebook's Haystack [11], and so on.

Under our observation, these systems are not perfect. As we know, most distributed systems are deployed as userspace libraries on large clusters of commodity machines. Each node in the cluster has local operating system and file system. Local system will load popular data into its cache. Usually cache unit is block. But in systems specific for small files, block may not be an appropriate choice as for the cache unit. Since a block may contain many files and quite often only a small part of them are frequently accessed. In fact, cache space has not been made fully use of. In distributed storage systems, files are usually randomly distributed in the whole system for load balancing. Little attention has been paid on file's inner relationship when files are written into disk. In web applications, when an image file is accessed, the images that make up the same hypertext document will also be accessed. The relationship between these related files have not been made use of.

In order to avoid these two drawbacks, we have proposed two optimizations on datanode: first, we build up a file-level cache which can make full use of the cache space. Second, we propose co-location of related files that store related files close to each other on disk which can take advantage of the disk technology trend that is toward improved sequential bandwidth [28]. In our evaluation, we find that with only file-level cache, we can improve the system's throughput by maximally 40%. With only co-location of related files, we can improve the throughput by maximally 20%. With both file-level cache and co-location of related files, we can improve the throughput by maximally 50%.

The rest of this article is organized as follows: Section 2 reviews the related works. Section 3 provides detailed motivation for our optimization. Section 4 describes our file-level cache in detail and Section 5 explains our co-location strategy of related files in detail. Section 6 describes our implementation on TFS [27]. We evaluate our optimization in Section 7 and draw a conclusion in Section 8.

II. RELATED WORK

With the rapid development of data-intensive applications, traditional file systems could no longer meet the demand for mass data storage. Many distributed storage systems have been developed to support applications with enormous amounts of data. For example, Amazon has designed Dynamo [8] to power parts of Amazon Web Services. Google has developed GFS [9] for its core data storage and usage needs.

Some peer-to-peer (P2P) systems have also looked at the problem of data storage and distribution [10, 12]. But, they are generally used as file sharing systems. Distributing data for performance, availability and durability has been widely studied in the file system and database system community. Compared with P2P storage systems that only support flat namespaces [29], distributed file systems typically support hierarchical namespaces [8, 24, 25, 26].

 ISBN: 978-1-61208-216-5

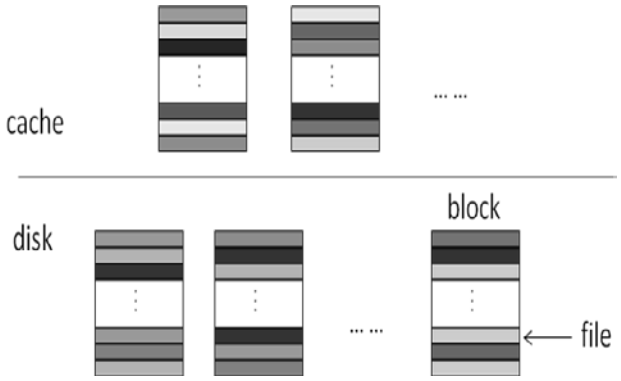


Figure 1. Cache organized in the unit of block

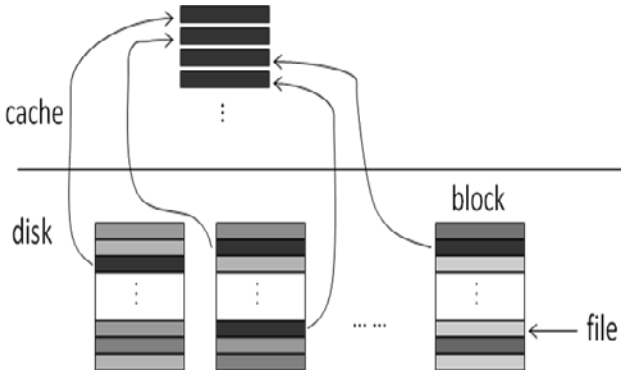


Figure 2. Cache organized in the unit of file

Studies of distributed file systems specified for small files have become a key component of storage systems research as applications specific for small files like images and micro bloggings have been increasing rapidly [21, 31, 32]. FastDFS is designed to meet the requirement of the website whose service based on files such as photo sharing site and video sharing site. Facebook has designed Haystack to serve its Photos application where data is written once, read often, never modified, and rarely deleted.

To increase the efficiency of the enormous small disk requests that characterize accesses to small files, much work have been done on the disk layout of small files. Localizing logically related objects is the choice of many file systems. Some researchers have investigated the value of breaking file system's disk storage into cylinders and moving the most popular data to the centermost cylinders in order to reduce disk seek distances [1, 2, 3]. From the same perspective of reducing disk seek distances, immediate files, an idea proposed by [4], moves inode to the first block of the file. This approach puts inodes together with their file data, which can improve the performance of read operations that read file data. Several other works [4, 5, 7] have proposed how to group related files together intelligently.

III. PRELIMINARY

Modern distributed file systems' topology can usually be divided into two kinds: master-slave structure and ring structure. Systems like GFS, HDFS [17], fastDFS usually organize machines in master-slave structure. In these systems there are two types of machines: one namenode with a large number of datanodes. Namenode handles metadata while datanode handles real data. This structure frees namenode from the data flow, so it can significantly reduce workload on namenode. Systems of the ring structure are often decentralized systems. There is no masternode, which only deals with metadata, in these systems. All machines act as the same role. All nodes can be called datanodes.

The growth and diffusion of applications specific for small files have led to systems that support efficient, secure and durable access of small files. Systems of the ring structure, like Dynamo and Cassandra [19], are intended to store relatively small objects. In systems of the master-slave structure, TFS is developed by Taobao [6] to store its enormous amounts of online commodity images.

As one part of the whole system, datanode plays a very important role. But so far there is hardly any optimization specified for datanode.

As we know, most distributed storage systems are not implemented in the kernel of operating systems, but are instead provided as userspace libraries, which means that they are all based on local file systems [20]. In systems that store small files, local file systems can hardly have any sense of single files; so they usually organize cache in the unit of block. Studies show that in most web service applications the file access pattern applies the Pareto principle, which means that only a small part of all files are frequently accessed while most files are rarely accessed [13, 14, 15, 16]. Since files are evenly distributed in the whole system, it is still true that of all the files that consist a block, only a small part are frequently accessed while others are rarely accessed. This strategy may result in low efficiency of cache and the waste of cache space.

Fig. 1 illustrates the case how cache space is wasted. Darker files are more frequently accessed while lighter files are less frequently accessed. When one block is loaded into cache as some popular files in it are accessed, files in the same block that are rarely accessed will also be loaded into cache. It stands a good chance that those unpopular files will have never been accessed before this block is replaced by other blocks. Cache space occupied by those unpopular files is wasted.

Modern distributed systems usually distribute files randomly for load balancing [30]. But this has not taken the logical relationship of files into consideration. In most cases, files can be partitioned into small groups based on application-specific knowledge. Files in the same group are closely related with each other that if one file is accessed most probably the others will be accessed too. For example, in online business like Amazon, all images stored in the system can be partitioned based on the commodities they describe. Once a commodity is skimmed by some customer, all images that describe this commodity will be accessed.

IV. FILE-LEVEL CACHE

Since cache space in datanode has not been fully made use of and much of it has been wasted. In order to make better use of cache space, we proposed a file-level cache on datanode. Fig. 2 illustrates the basic idea of our file-level

 ISBN: 978-1-61208-216-5

cache. We organize cache in the unit of single files rather than blocks. Each time we load a single file into cache rather than a block.

Studies of cache replacement strategy have been a topic for many years. Many replacement algorithms have been proposed, such as FIFO [35], LRU [36], OPT [37] and so on. FIFO is too simple to make the most of cache while OPT is just an ideal model, which has not been put into practice. The LRU strategy discards the least recently used items when new items need to be loaded into cache. This strategy fits our applications well so we choose LRU as our replacement strategy. Since files are not in the same size, we organize files in the cache as a list. As enormous accesses to small files may generate great amounts of cache misses while all these cache misses would search the entire list, which results in great amounts of searching time. In order to avoid these unnecessary searches of the list generated by cache misses, we introduce a bloom filter.

For an incoming read request, our algorithm does the following:

- Check the bloom filter to see whether this file is in the cache. If the bloom filter indicates that it is not in the cache, load this file from disk and add a node that represents this file to the head of the list. Reply with the data loaded from disk.
- If bloom filter indicates that this file may be in the cache, search the list to see whether this file is in the cache.
- If it is in the cache, move the node that represents this file to the head of the list and reply with data in the cache.
- If it is not in the cache, load this file from disk and add a node that represents this file to the head of the list. Reply with the data loaded from disk.

Each time we load a new file into the cache, if the total size of all the data in the cache is larger than the cache size, we recursively remove the node in the tail until the total size is smaller than the cache size.

Compared with the time of loading files from disk, the time of searching file-level cache can be ignored. With bloom filter avoiding most of the unnecessary searches of cache. We believe that our file-level cache can greatly reduce the overall file access time.

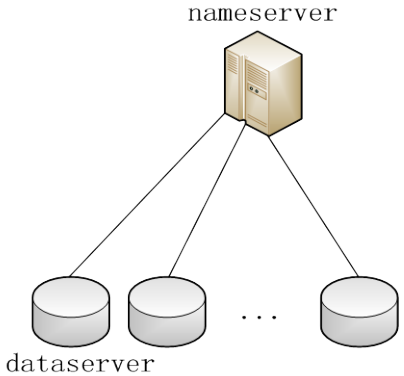


Figure 3. Stucture of TFS.

V. CO-LOCATE FILES BASED ON AFFINITY

We borrowed a concept of affinity to describe the inner relationship between files based on application-specific knowledge. We say several files are affinitive in case that if one of them is accessed, the rest are very likely to be accessed. As modern disk technology trend is toward improved sequential bandwidth. To better exploit bulk data bandwidth and avoid frequent reposition to new locations, we use co-location to place affinitive files at adjacent disk locations.

In modern distributed storage systems, a write process usually consists of the following steps:

- System randomly chooses a datanode based on the current workload on every datanode.
- Client contacts and sends data to the chosen datanode directly.
- After all data have been received, datanode commits.

Each write request goes through the entire write process. Since datanode is chosen randomly, files are distributed randomly in the whole system.

In order to co-locate affinitive files, we do not write disk for single write request, instead we collect files in a buffer and write them to disk in batches. Each time the client receives a write request, it puts the file in the buffer. Files in the buffer are divided into groups based on application-specific knowledge. When buffer is full or a time limit is met, system begins the write process.

In our approach, we have made some modifications to modern write process to achieve co-location of affinitive files. The write process consists of the following steps:

- System randomly chooses a datanode for the first group of files in the buffer based on current workload on every datanode.
- Client contacts and sends data of files in the first group to the chosen datanode directly.
- After all data have been received, datanode commits.
- Check whether the buffer is empty. If the buffer is not empty, go to the first step; otherwise ends the write process.

We believe that co-locating files based on the relationship inherent in application-specific knowledge can be exploited to successfully realize the performance potential of modern disks' bandwidth.

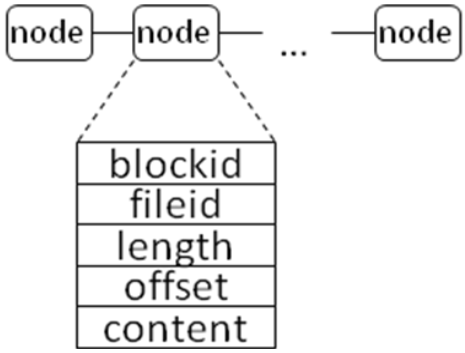


Figure 4. Structure of cache

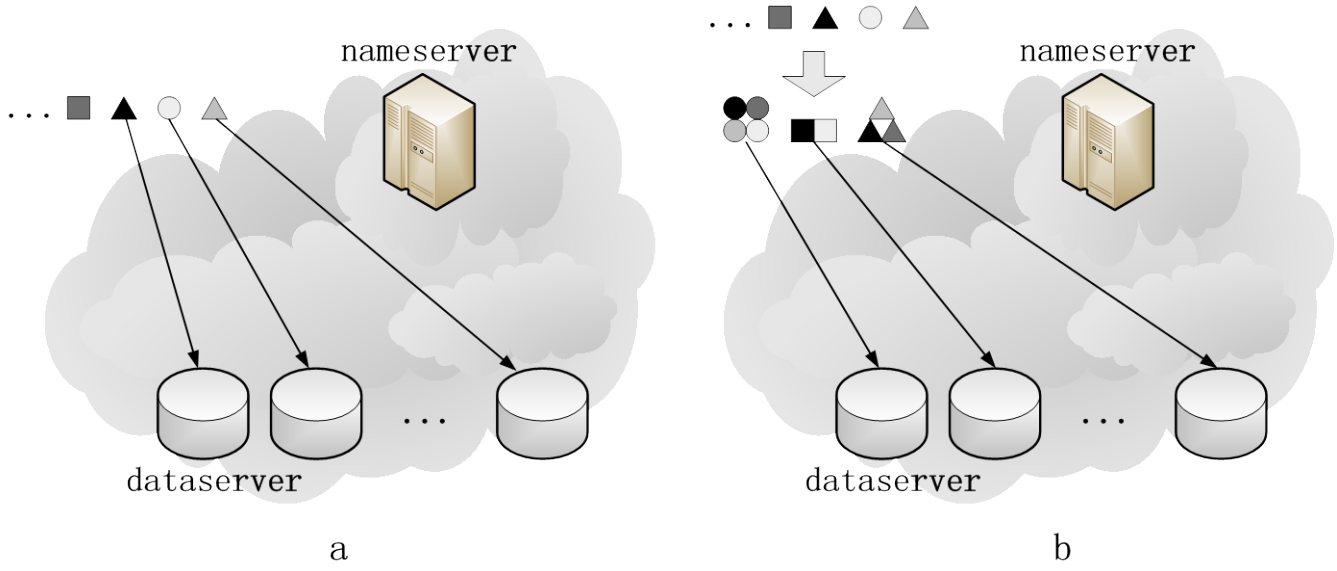


Figure 5. Write process.

VI. IMPLEMENTATION

This section describes our detailed implementation on TFS. TFS is the abbreviation of Taobao File System. It is a distributed file system and is designed to store great amounts of small images. The average file size is 16.7 kilobytes. Fig. 3 depicts the basic structure of TFS. Like GFS, a TFS cluster consists of multiple nodes which can be divided into two types: one nameserver and a large number of dataservers. Files are organized as blocks which are usually 64 megabytes in size. Dataserver stores blocks while nameserver maintains the logical mappings of blocks to dataservers. TFS also makes some optimizations on the filename to reduce metadata stored on nameserver. For each file, TFS encodes the id of the block that contains the file to the file's filename. So when a client gets the filename, it can get the id of the block that contains the file by decoding the filename. Each block is replicated several times throughout the network.

A read process in TFS goes like this: client sends a read request to nameserver. Nameserver replys with the cooresponding location (i.e., the dataserver). The client contacts the corresponding dataserver. Dataserver replies with the file's content.

A write process in TFS goes like this: client sends a write request to nameserver. Nameserver chooses a dataserver based on the current workload on every dataserver and replys with the dataserver. Client sends data to dataserver. After all data have been received, dataserver commits to nameserver. Dataserver replys to client that write process completes.

In our file-level cache implemented on datanode, files are organized as a list. As Fig. 4 shows, each node in the list represents a file which records the file's blockid, fileid, file length, offset in the block and the file's content. We use a standard bloom filter in our implementation. Each bit in the array is set to 0 when service starts. In order to improve our bloom filter's accuracy, i.e., to make the false positive [33] rate as low as possible, we use three hash functions [34].

Fig. 5a shows the writing mechanism of TFS: a random dataserver is chosen to store the first file in the write request queue. Fig. 5b shows the writing mechanism in our approach: files are stored in the buffer and grouped according to which hypertext document they make up. Then a dataserver will be allocated for each group of files. Files in the same group will be written into the same dataserver and the storage mechanism in dataserver will guarantee that these files will be placed in adjacent locations on disk. In our implementation, since file size is relatively small, we believe that 4 megabytes is enough for the buffer size and our experiments have proved that this is an appropriate choice.

VII. EVALUATION

This section reports measurements of our implementation on TFS, which shows that it can dramatically improve the system's performance.

Our TFS cluster contains 22 nodes, with 2 of them act as nameservers and the rest act as dataservers. Each node runs 64-bit ubuntu10.04 and uses an Intel Core 2 Duo E6850 3GHz CPU, 4GB RAM, and a 500GB 7200rpm hard disk.

We use two main data sets for testing. Our synthetic data set simulates the trace in applications in which files show a certain clustering effect, which means that each file may have a strong relationship with several other files. So we can use this characteristic to co-locate files that have strong relationships with each other. Files' average size is 20 kilobytes and total size is 2 terabytes. Our second data set is a real world data set which comes from our college's online teaching system. Teachers use this system to share slides with students, while students use this system to submit homework. This system is also used as a communication platform for students and teachers to discuss with each other. So a great number of small files are stored in this system.

 ISBN: 978-1-61208-216-5

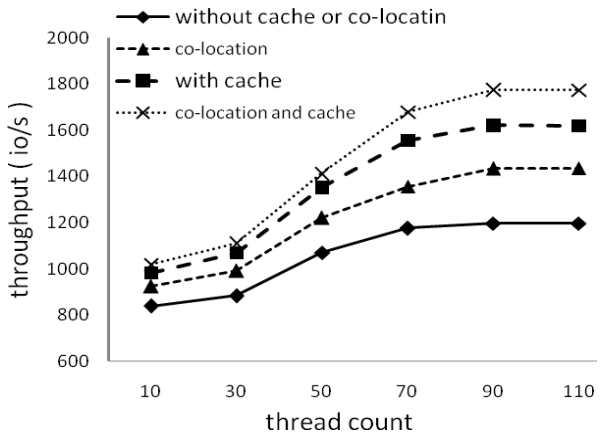


Figure 6. Read throughput under different numbers of thread.

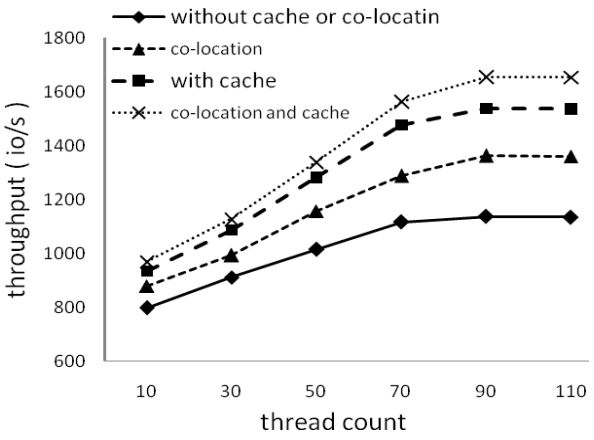


Figure 8. Throughput under different numbers of thread.

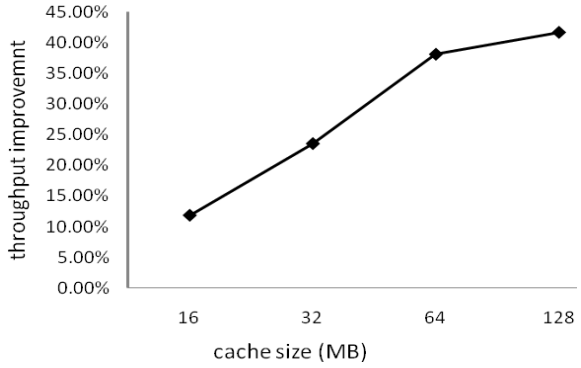


Figure 7. Throughput improvement of different cache size.

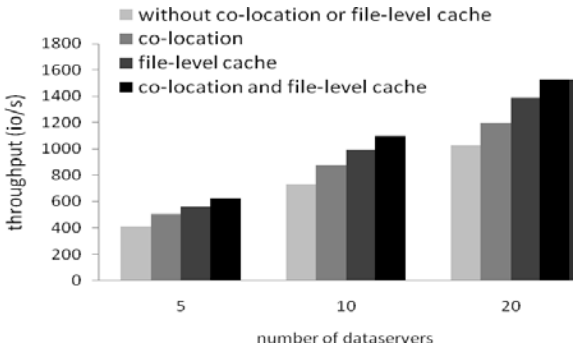


Figure 9. Throughput of different numbers of datanode.

Since the disk throughput is hard to evaluate and is not suitable for the evaluation of the whole system, we use the I/O throughput of the whole system, i.e., the read/write requests complete per second, as our criterion.

Results with synthetic data

Fig. 6 shows the throughput of only read requests under different thread counts. Co-location improves performance by 10%-20%. File-level cache improves performance by 17%-35%. Co-location and file-level cache together improve performance by 20%-49%.

Fig. 8 shows the throughput of read and write requests together under different thread counts. Read write ratio is 20:1. The performance improvement is nearly the same as the performance improvement in conditions with only read requests.

Results with real world data

First, we conduct an experiment with our real world data set in 2 scenarios: without co-location or file-level cache, with only file-level cache. Fig. 7 shows the throughput improvement with varying cache size. When cache size is smaller than 64 megabytes, throughput increases almost linearly as cache size increases. But, it increases much slower after cache size reaches 64 megabytes. Since the file list gets longer as cache size increases, the searching time becomes larger.

Fig. 9 shows the throughput of the 4 scenarios with varying numbers of datanode. We can see that averagely co-location can improve throughput by 20%, file-level cache can improve throughput by 35%, co-location and file-level cache together can improve throughput by 50%.

VIII. CONCLUSION AND FUTURE WORK

Distributed storage systems designed for small files are widely used in large datacenters to power today's popular applications specific for small files. Aiming at drawbacks that exist in these systems, we proposed two optimizations on datanode in distributed storage systems. By co-locating related files, we can increase the system's throughput by maximally 20%. By implementing a file-level cache on datanode, we can increase the system's throughput by maximally 40%. By implementing the two optimizations together on datanode, we can increase the system's throughput by maximally 50%.

We believe that, for most applications, page information provides useful information about relationships between files that can be exploited by grouping. So, in this paper, we investigate the approach of grouping files that make up a single hypertext document. Other approaches based on application-specific knowledge are worth investigating. In

 ISBN: 978-1-61208-216-5

our implementation, we have provided interfaces to support other application-specific knowledge.

ACKNOWLEDGMENT

We would like to thank our program committee shepherd Petre and the anonymous reviewers for their insightful comments and constructive suggestions. This research is supported by NSFC 61133005.

REFERENCES

[1] P. Vongsathorn and S. Carson, "A System for Adaptive Disk Rearrangement", *Software – Practice and Experience*, Vol. 20, No. 3, March 1990, pp. 225–242.

[2] C. Ruemmler and J. Wilkes, "Disk Shuffling", Technical Report HPL-CSP-91-30, Hewlett-Packard Laboratories, October 3, 1991.

[3] "Smart Filesystems", *Winter USENIX Conference*, 1991, pp. 45–51.

[4] S. J. Mullender and A. S. Tanenbaum, "Immediate Files", *Software–Practice and Experience*, 14 (4), April 1984, pp. 365–368.

[5] G. R. Ganger and M. F. Kaashoek, Embedded inodes and explicit grouping: exploiting disk bandwidth for small files, In ATEC '97: Proceedings of the annual conference on USENIX Annual Technical Conference, pages 1–1, Berkeley, CA, USA, 1997. USENIX Association.

[6] http://www.taobao.com: June, 2012

[7] Z. Zhang and K. Ghose, hfs: a hybrid file system prototype for improving small file and metadata performance, SIGOPS Oper. Syst. Rev., 41(3):175–187, 2007.

[8] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 205-220, 2007.

[9] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, pp. 29-43, 2003.

[10] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The Bittorrent P2P File-Sharing System: Measurements and Analysis" Peer-to-Peer Systems IV. vol. 3640, M. Castro and R. van Renesse, Eds., ed: Springer Berlin / Heidelberg, 2005, pp. 205-216.

[11] D. Beaver, S. Kumar, H. C. Li, J. Sobel, and P. Vajgel, "Finding a needle in Haystack: facebook's photo storage," presented at the Proceedings of the 9th USENIX conference on Operating systems design and implementation, Vancouver, BC, Canada, 2010.

[12] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Looking up data in P2P systems," *Commun. ACM*, vol. 46, pp. 43-48, 2003.

[13] M. E. J. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemporary Physics*, vol. 46, pp. 323-351, 2005/09/01 2005.

[14] A. Charnes, W. W. Cooper, B. Golany, L. Seiford, and J. Stutz, "Foundations of data envelopment analysis for Pareto-Koopmans efficient empirical production functions," *Journal of Econometrics*, vol. 30, pp. 91-107, 1985.

[15] S. Joseph E, "Self-selection and Pareto efficient taxation," *Journal of Public Economics*, vol. 17, pp. 213-240, 1982.

[16] T. M. Tripp and H. Sondak, "An evaluation of dependent variables in experimental negotiation studies: Impasse rates and pareto efficiency," *Organizational Behavior and Human Decision Processes*, vol. 51, pp. 273-295, 1992.

[17] http://hadoop.apache.org/hdfs/: May, 2012

[18] http://code.google.com/fastdfs/: April, 2012

[19] http://cassandra.apache.org/: June, 2012

[20] M. K. Aguilera, A. Merchant, M. Shah, A. Veitch, and C. Karamanolis, "Sinfonia: a new paradigm for building scalable distributed systems," presented at the Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, Stevenson, Washington, USA, 2007.

[21] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni, "PNUTS: Yahoo!'s hosted data serving platform," *Proc. VLDB Endow.*, vol. 1, pp. 1277-1288, 2008.

[22] J. Dean, "Evolution and future directions of large-scale storage and computation systems at Google," presented at the Proceedings of the 1st ACM symposium on Cloud computing, Indianapolis, Indiana, USA, 2010.

[23] S. A. Weil, K. T. Pollack, S. A. Brandt, and E. L. Miller, "Dynamic Metadata Management for Petabyte-Scale File Systems," presented at the Proceedings of the 2004 ACM/IEEE conference on Supercomputing, 2004.

[24] D. Hitz, J. Lau, and M. Malcolm, "File system design for an NFS file server appliance," presented at the Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference, San Francisco, California, 1994.

[25] A. W. Leung, M. Shao, T. Bisson, S. Pasupathy, and E. L. Miller, "Spyglass: fast, scalable metadata search for large-scale storage systems," presented at the Proccedings of the 7th conference on File and storage technologies, San Francisco, California, 2009.

[26] H. C. Lim, S. Babu, and J. S. Chase, "Automated control for elastic storage," presented at the Proceedings of the 7th international conference on Autonomic computing, Washington, DC, USA, 2010.

[27] http://code.taobao.org/p/tfs/wiki/index/: May, 2012

[28] L.W. McVoy and S.R. Kleiman, "Extent-like Performance from a UNIX File System", in Proc. USENIX Winter, 1991, pp.33-44.

[29] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," presented at the Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, 2001.

[30] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: a scalable, high-performance distributed file system," presented at the Proceedings of the 7th symposium on Operating systems design and implementation, Seattle, Washington, 2006.

[31] Z. Zhang and K. Ghose, "hFS: a hybrid file system prototype for improving small file and metadata performance," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 175-187, 2007.

[32] M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff, and J. K. Ousterhout, "Measurements of a distributed file system," SIGOPS Oper. Syst. Rev., vol. 25, pp. 198-212, 1991

[33] http://en.wikipedia.org/wiki/Type_I_and_type_II_errors: June, 2012

[34] http://en.wikipedia.org/wiki/Bloom_filter: June, 2012

[35] http://en.wikipedia.org/wiki/FIFO: June, 2012

[36] http://en.wikipedia.org/wiki/Cache_algorithms#Least_Recently_Used : June, 2012

[37] http://en.wikipedia.org/wiki/Page_replacement_algorithms#The_theoretically_optimal_page_replacement_algorithm: June, 2012

 ISBN: 978-1-61208-216-5

ERHA: Execution and Resources Homogenization Architecture

Guilherme Galante, Luis Carlos Erpen de Bona
Department of Informatics
Federal University of Paraná
Curitiba, PR – Brazil
{ggalante,bona}@inf.ufpr.br

Paulo Antonio Leal Rego, José Neuman de Souza
Master and Doctorate in Computer Science
Federal University of Ceará
Fortaleza, CE – Brazil
pauloalr@lia.ufc.br, neuman@ufc.br

***Abstract*—In this paper, we present the Execution and Resources Homogenization Architecture (ERHA). The architecture aims to provide mechanisms for submitting and executing batch applications in private IaaS clouds using homogeneous virtual environments created over heterogeneous physical infrastructure. With ERHA it is possible to deploy and execute applications in IaaS clouds in an automatic and easy way. The architecture creates homogeneous virtual environments and manages the entire execution process, from source code submission to results collection phase. The results confirmed the architecture efficiency in deploying parallel applications in clouds and reducing significantly the disparities in execution time using different machine models.**

***Index Terms*—scientific applications; application execution; homogeneous environments.**

I. Introduction

The rapid provisioning of independent and isolated resources, hardware and software customization, quick access to resources as well as on-demand scalability, have made cloud computing an attractive model for the scientific community. In fact, many scientists have adopted this new paradigm, moving their data and performing *in silico* experiments in the cloud [1]–[4].

However, the deployment and execution of scientific applications are generally not straightforward, comprising a set of complex hardware and software configurations. Other issues must be considered in a cloud scenario, e.g., particular control aspects of different clouds, interfaces to be used, number of virtual machines (VMs) to be deployed and what resources will be needed by them.

According to [5], running a scientific application in the cloud presents three different challenges: initial application deployment, subsequent application execution, and data transfers to/from the cloud. Generally, the application deployment in a cloud requires that all software (and possibly data) be stored in a VM image, which is sent to and stored in the cloud. Thus, the user can create a new VM from this image, access it and run applications. At the end, the application results must be copied to a non-volatile storage using a network protocol. These steps seem extremely simple for a computer scientist, but may pose a challenge to people from other areas. The challenge may be even greater when it comes to parallel applications.

Another related problem is the provisioning of computing resources in heterogeneous physical infrastructures. The VMs performance is directly related to the physical machines (PMs) where they are allocated. As the cloud computing environment is commonly highly heterogeneous, there may be machines with different processors. This difference between the CPUs can directly influence the performance of VMs and consequently the applications encapsulated within. An example is presented in [6], where the performance results of a MapReduce execution on Amazon EC2 show fluctuations up to 24%. Another issue related to performance disparities due to heterogeneity is that the slowest instance will be the bottleneck for an entire execution of the application, thus, under-utilizing faster machines.

Considering these issues, this paper presents the Execution and Resources Homogenization Architecture (ERHA), a solution to automate the deployment and execution of sequential and parallel batch applications in clouds, providing homogeneous computing resources. This type of application was focused because it is very common in scientific computations. The architecture provides a language to describe the resources and the execution parameters, a standard method to deploy and execute the application in clouds, and a mechanism that enables the allocation of VMs using processing units (PUs), a metric that represents the effective processing power of each machine (physical or virtual).

Four experiments show the architecture efficiency in deploying applications in clouds in an easy way and reducing significantly the disparities in execution time using different PMs models.

The remainder of the paper is organized as follows. Section II presents the related works. Section III introduces the proposed architecture. In Section IV, the experiments are presented and the results are discussed. Finally, in Section V, we present our conclusion and potential future research topics.

II. Related Work

Several studies [2], [7], [8] have assessed the aspects of running scientific applications on public clouds. The feasibility of the current cloud services (Amazon and GoGrid) for the execution of scientific applications is explored in [7]. A performance comparison of eight applications (CAM, Gamess, GTC,

 ISBN: 978-1-61208-216-5

IMPACT-T, MAESTRO, MILC, Paratec and HPCC) running in a private virtual cluster and in Amazon EC2 is presented in [8]. In a similar study, the impact of using different MPI libraries in an atmospheric model running on EC2 is analyzed in [2].

All these studies present some concerns about applications performance. Large fluctuations of high-performance computing workloads on cloud infrastructure were reported in [9]. In [7], the authors have found that many Amazon and Google services exhibit large performance changes over time.

The studies which use techniques for limiting the CPU usage of VMs are generally related to the dynamic provisioning of resources. In [10], Xen's performance isolation for I/O intensive applications is studied and two mechanisms are proposed to improve CPU and network resource isolation. In [11], an architecture for dynamic resources management upon the hypervisor Xen is presented. The authors dynamically adjust the amount of CPU and memory of VMs to reduce service level objectives (SLOs) violation. In [12], an adaptive control system which automates the task of tuning the resources allocation for the maintenance of SLOs is presented. The work uses KVM hypervisor and focuses on maintaining the expected response time for Web applications, by tuning the CPU usage.

Unlike aforementioned works, the hypervisors KVM and Xen are used in ERHA and the techniques for limiting the CPU usage are leveraged to handle the problem of providing homogeneous resources despite being in a heterogeneous infrastructure. In addition, besides Amazon EC2 (with the Elastic Compute Unit), no other public cloud provider or research uses an abstraction like our PU for representing CPU resources.

About the creation of virtual environments and the execution of applications, the three most similar works are highlighted: Neptune [13], DADL [14] and Nimbus Context Broker [15]. Neptune is a domain specific language that automates configuration and deployment of existing HPC software in AppScale clouds. DADL (Distributed Application Description Language) is a language for describing hardware requirements, behavior and architecture of distributed applications. Finally, Nimbus Context Broker is a tool used to create and provide virtual clusters.

Cloud computing is taking larger proportions in the scientific field and there are several studies about scientific applications performance in cloud. Despite this, to the best of our knowledge, there is no published research addressing the support for running sequential and parallel batch applications in clouds, especially considering the virtual environments homogenization, as we show with ERHA.

III. ERHA Architecture

ERHA is an architecture that aims to provide mechanisms to submit and execute batch applications in private IaaS clouds (e.g., OpenNebula and Eucalyptus) using homogeneous virtual environments created over heterogeneous physical infrastructure. With ERHA it is possible to deploy and execute applications in clouds in an automatic and easy way. The architecture creates the virtual environment and manages the entire execution process, from source code submission to results collection phase.

Furthermore, the architecture allows users to configure their VMs' processing power using an uniform metric, the Processing Units (PU). A PU is a value in terms of GFLOPS (billion floating-point operations per second), or other metric that represents the processing power of each physical or virtual machine. The system administrator defines a value to the PU, which is used in the VMs allocations. All virtual instances requested by the user will have equivalent computing power, regardless the underlying PM and its processor model. For example, considering a PM with 30 GFLOPS and the PU set as 10 GFLOPS, it is possible to allocate three VMs with 1 PU, or one VM with 1 PU and another with 2 PUs.

To implement the execution environment and the resources homogenization, ERHA uses three layers, as shown in Figure 1: (1) *Resource Description Layer*, (2) *Execution Management Layer* and (3) *Allocation Layer*.

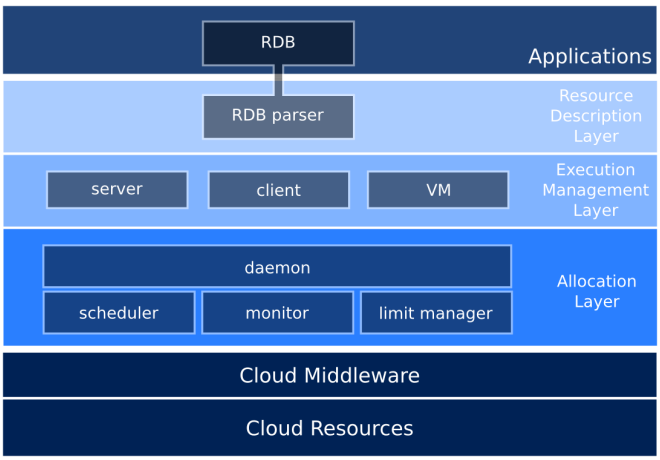


Fig. 1. Architecture layers

The *Resource Description Layer*, *Execution Management Layer* and *Allocation Layer* are presented in Section III-A, III-B, III-C, respectively. A complete example of running an application envolving all ERHA layers is presented in Section III-D.

A. Resource Description Layer

The *Resource Description Layer* (RDL) is responsible for receiving the resources demands and informing it to the other layers. The resources needed by the application are informed in the *Resource Description Block* (RDB). The block is made up of a set of attribute-value pairs and it must be inserted into the source code, marked by the reserved word *#neb_config* and delimited by braces ({}). An example of RDB is shown in Figure 2.

In this example, a VM named OMPTest is requested in an OpenNebula cloud. The VM has four virtual CPUs (VCPU), eight PUs and 256 MB RAM, no disk attachment is needed

 ISBN: 978-1-61208-216-5

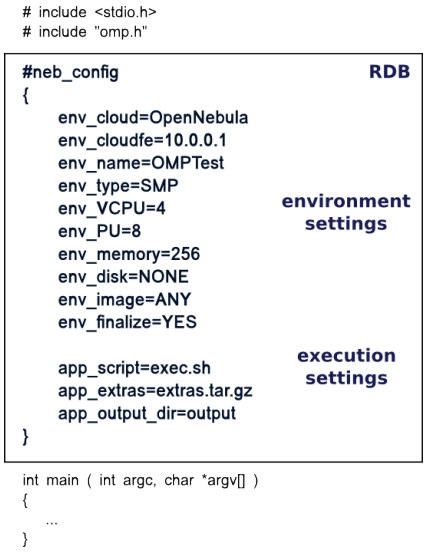


Fig. 2. RDB example

and a default image is used. When the execution ends, the VM will be finalized. Note that in RDB is informed the total amount of PUs of a VM, and this processing power is divided by all VCPUs (in this example, each VCPU has 2 PUs).

The rest of RDB describes the execution script name, the *extras.tar.gz* file, which contains the applications dependencies, and the output directory, where results will be saved.

The information is obtained by the *RDB Parser*, which parses all fields and converts them to the specific cloud middleware format (OpenNebula, in this example). The parser sends the formatted data to the *Execution Management Layer*, which will use them to request the resources to the cloud via *Allocation Layer*.

B. Execution Management Layer

The *Execution Management Layer* (EML) handles the execution process, controlling all needed interactions with the cloud during the application execution. EML is a client-server application composed of three components: 1) *Client*, 2) *Server* and 3) *VM-Daemon*.

The *Client* is installed in the user machine and provides a command-line interface that is used to submit the applications to the cloud. This interface is also used to track the execution progress and receive error messages. It is also responsible for receiving the resources description from *RDB Parser* and sending it to *Server* module, deploying the application in the cloud, managing the execution and collecting the results.

The *Server* runs in the cloud front-end and is responsible for receiving the requests and sending the performing the actions related to VMs creation and finalization. It receives the resources description from the *Client*, converts this information to the appropriate format and makes the requests to the *Allocation Layer*. The same process is performed to finalize the virtual environment. It is necessary just one *Server* instance to serve all *Clients*.

The last component is the *VM-Daemon*. This module must be inserted into VM image and is initialized on VM boot process. When started, the *VM-Daemon* sends and receives information to/from the *Client*, including IP address, VM identification, authorization keys, error warnings and commands that must be executed in VM.

C. Allocation Layer

The *Allocation Layer* (AL) was designed to perform the VMs allocation based on PUs. AL ensures uniform allocation of computing power to VMs and standardizes the representation of the processing power of a cloud infrastructure through the use of PUs. The PU is the abstraction used for representing the processing power, similar to ECU (Elastic Computing Unit) for Amazon EC2.

Despite the VM performance being directly related to the underlying PM, the use of PUs metric makes it possible to guarantee VM processing power without worrying about the infrastructure heterogeneity. Four modules make up the AL: (1) *Limit Manager*, (2) *Monitor*, (3) *Scheduler* and (4) *Daemon*.

Limit Manager is the module responsible for applying the limits on CPU usage. When a new VM is created, for example, the *Daemon* invokes the *Limit Manager*, which sets the CPU resource that can be used by the VM, considering the amount of PUs allocated to it.

This feature is implemented limiting the CPU cycles used by each VM. To achieve the desired results, it is necessary that the hypervisors allow a way to set the percentage of CPU that will be used by a given VM. Xen has this functionality natively implemented through Credit Scheduler algorithm [16]. An alternative for limiting the CPU usage for the Kernel-based Virtual Machine (KVM) with the *cpulimit* tool is evaluated in [17] and is used in this work.

Monitor is the module responsible for capturing information about the entire infrastructure. This module can directly access the infrastructure data or make requests to the cloud computing middleware. The monitored information is related to CPU, memory, storage and network for each PM and VM.

All decisions about the VMs allocation on PMs are taken by *Scheduler*. Unlike the schedulers present in the main cloud computing middleware, like Eucalyptus and OpenNebula, the proposed scheduler must not consider the raw information about CPU, memory and hard drive to define where the VMs will be allocated. Once the architecture is based on PUs, all allocations must be made based on the number of PUs required by the VM and the amount of free PUs in the PMs. The architecture provides some basic scheduling policies, e.g., Random, Round Robin and First Fit. New scheduling policies can be easily created and added to the architecture.

The last module, the *Daemon*, manages all the other modules and controls the infrastructure. It communicates with the cloud computing middleware and maintains information about the PMs and running VMs (captured by the Monitor). In addition, it has access to all the requests that come to the middleware. The *Daemon* receives the requests from

 ISBN: 978-1-61208-216-5

Execution Layer and invokes the *Scheduler* when there are VMs at pending state. When the *Daemon* receives the VM-PM mapping from *Scheduler*, it allocates the VMs. The *Daemon* is also responsible for storing information about the PUs and invoking the *Limit Manager* to apply the limit of CPU usage when a VM is started or migrated.

D. Running Applications with ERHA

To demonstrate how ERHA works, this section presents an example of using the architecture to run the scenario described in the RDB shown in Figure 2. The environment creation and the application execution is performed in thirteen steps, as illustrated in Figure 3 and explained in sequence.

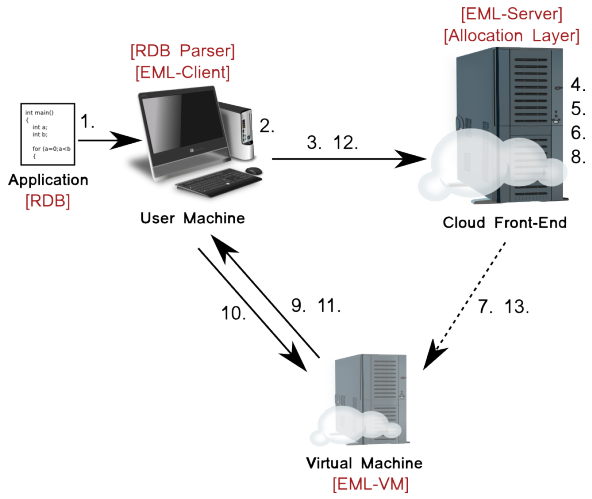


Fig. 3. Application Execution Steps

The process starts when (1) the user adds the *RDB* to the application, provides the scripts and other dependencies and submits the application through *EML-Client*. The *Client* (2) uses the RDB Parser to get information about the requested environment and (3) sends the VM template to *EML-Server*, which (4) requests a VM to the *Allocation Layer*. In the *Allocation Layer*, (5) the *Daemon* detects a pendency and calls the *Scheduler*, that (6) returns a VM-PM mapping. Then, (7) the *Daemon* sends the VM creation request to the cloud middleware, and the VM is created. Next, the *Daemon* calls the *Limit Manager* to (8) apply the CPU limitation to the VM, based on the PU configuration established in RDB.

After VM operating system booting process, (9) the *EML-VM* gets the IP and the VM public-key and sends them to *EML-Client*. The IPs are used to identify the VMs during the execution process, and the public-key is used to allow passwordless remote access. In the following step, (10) *EML-Client* uploads the application and its dependencies to the VM and sends the compilation and execution commands. After finishing the application execution, the (11) *EML-VM* collects the results and sends them back to the *Client*. If the machines are no longer needed, (12) *EML-Client* requests the VM destruction to the *Allocation Layer* and (13) the *Daemon* sends the command for VM destruction to the cloud middleware.

IV. Tests and Results

To evaluate the proposed architecture, a prototype was implemented using Python, Ruby and Shell Script. OpenNebula 2.2.1 was used as cloud middleware and KVM and Xen as virtualization technology. The choice of OpenNebula was made based on the flexibility for VMs creation, which allows the configuration of resources according to application needs, unlike Eucalyptus and OpenStack, in which a pre-configured class must be chosen.

TABLE I
Physical Machines Configurations

	Ci7	Ci5	X4	C2D
Processor	Intel Core i7-930 2.8 GHz	Intel Core i5-750 2.66 GHz	Intel Xeon X3430 2.4 GHz	Intel Core 2 Duo E7400 2.8 GHz
PUs (Total)	10	9	8	5
Memory	24 GB	4 GB	8 GB	4 GB
OS	Ubuntu Server 11.04 64 bits			Debian 6.0 64 bits
Hypervisor	KVM			Xen

A heterogeneous private cloud with OpenNebula was used as testbed, consisting of 2 machines model Ci5, 1 machine model Ci7, 1 machine model X4 and 1 machine model C2D, all connected by a Gigabit Ethernet network. The configurations of the physical machines are presented in Table I. All VMs use Ubuntu Server 11.04 32 bits as operating system. All source code and applications used in experiments are available in the project site [18].

The Intel Linpack benchmark was used to collect the VMs' computing power running inside all PMs. With this information, it is possible to find the relation between CPU usage (%) and the amount of PUs for all PMs. For this work, the PU was set to 3 GFLOPS (50% of one core of X4), and the relation %CPU/PU is presented in Table II. This specific configuration is more suitable for CPU-intensive applications because only the Linpack benchmark is used in the process of PU definition.

TABLE II
Relation between the CPU usage and the amount of PUs for each physical machine.

	CPU usage			
	Ci7	**Ci5**	**X4**	**C2D**
1 PU	39%	41%	50%	44%
2 PUs	78%	81%	100%	83%
3 PUs	119%	118%	150%	130%
4 PUs	157%	162%	200%	165%
5 PUs	196%	218%	250%	200%
6 PUs	234%	260%	300%	N/A
7 PUs	274%	305%	350%	N/A
8 PUs	320%	354%	400%	N/A
9 PUs	368%	400%	N/A	N/A
10 PUs	400%	N/A	N/A	N/A

We conducted four experiments with two parallel OpenMP applications: a 2D heat transfer problem [19] and a LU decomposition algorithm [20]. The heat transfer problem consists in

 ISBN: 978-1-61208-216-5

solving a partial differential equation to determine the variation of the temperature within the heat conducting body. LU decomposition is a method to factorize a matrix as the product of a lower triangular matrix and an upper triangular matrix. This algorithm is a key step in several fundamental numerical algorithms in linear algebra such as solving a system of linear equations, inverting a matrix, or computing the determinant of a matrix.

The experiments were grouped in two sets, each set examining different issues. In Set A, three experiments test the ERHA efficiency in providing a homogeneous environment. In Set B, the impact of PMs load in application performance is evaluated. For each test, the applications were run 10 times and the results were combined to calculate mean, in a 95% confidence interval.

A. ERHA Efficiency

In this section, three different test cases were used to validate the ERHA architecture and to analyze its efficiency using different configurations.

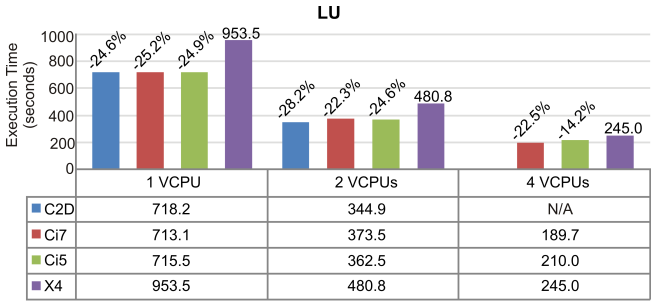
1) Experiment 1: In the first experiment we tested the architecture with the Limit Manager module disabled. The purpose of this experiment is to evaluate the difference in the PMs processing power. The LU decomposition and heat transfer applications were executed on a virtual environment allocated on one PM of each model (Ci5, Ci7, X4 and C2D). The tests were performed using VMs with 1, 2 and 4 VCPUs, except for C2D, which supports VMs with just 2 VCPUs.

Figure 4 shows the applications execution time for each configuration. It can be seen that both applications take longer in C2D for all configurations. The difference reaches 30.3% in the case of heat transfer application with 1 VCPU and 28,2% in the case of LU decomposition with 2 VCPUs. These results emphasize the need for a solution which considers the infrastructure's heterogeneity once they confirm the influence of the underlying PM on virtual machines performance and its applications.

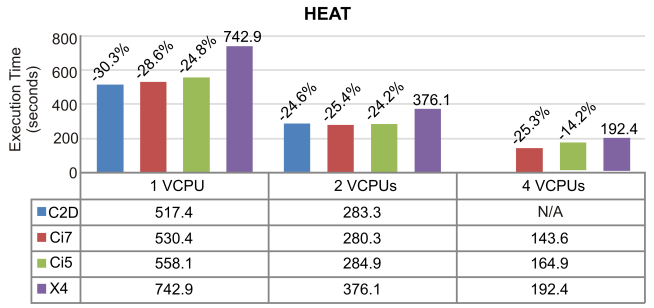
2) Experiment 2: In the second experiment, the Limit Manager features were evaluated. The LU decomposition and heat transfer applications were executed in VMs with 1, 2 and 4 VCPUs and setting the processing power to 1 PU per VCPU, totaling respectively 1, 2, and 4 PUs. The results are presented in Figure 5.

As expected, the execution time is greater than in Experiment 1, due the CPU limitation imposed by Limit Manager. Taking the PM X4 as basis, in LU decomposition the largest difference is 1.7% in processing time, while in the heat transfer problem the largest difference is 6%. The results show that performance variability is reduced if compared with Experiment 1.

3) Experiment 3: In the third experiment, the previous experiment we repeated using 2 PUs per VCPU. The VMs with 1, 2 and 4 VCPUs had processing power of respectively 2, 4 and 8 PUs. The results can be observed in Figure 6. The most relevant difference in execution time is 3.5% for LU

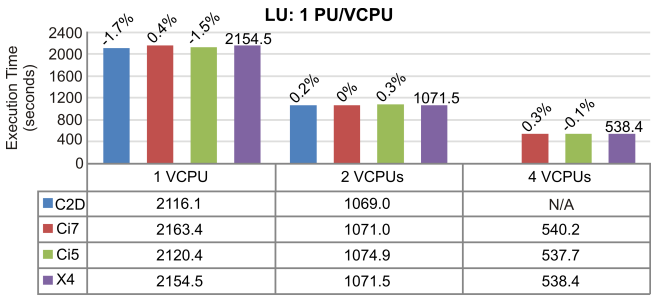


	1 VCPU	2 VCPUs	4 VCPUs
C2D	718.2	344.9	N/A
Ci7	713.1	373.5	189.7
Ci5	715.5	362.5	210.0
X4	953.5	480.8	245.0

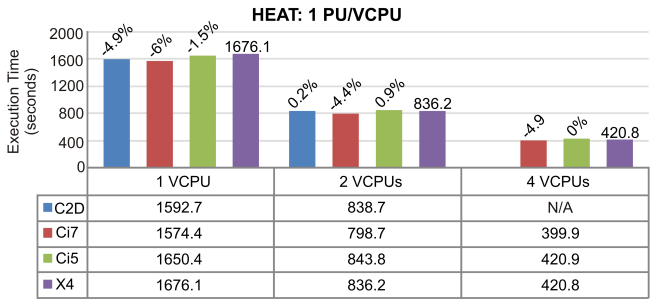


	1 VCPU	2 VCPUs	4 VCPUs
C2D	517.4	283.3	N/A
Ci7	530.4	280.3	143.6
Ci5	558.1	284.9	164.9
X4	742.9	376.1	192.4

Fig. 4. ERHA Efficiency - Execution time without CPU limiting



	1 VCPU	2 VCPUs	4 VCPUs
C2D	2116.1	1069.0	N/A
Ci7	2163.4	1071.0	540.2
Ci5	2120.4	1074.9	537.7
X4	2154.5	1071.5	538.4



	1 VCPU	2 VCPUs	4 VCPUs
C2D	1592.7	838.7	N/A
Ci7	1574.4	798.7	399.9
Ci5	1650.4	843.8	420.9
X4	1676.1	836.2	420.8

Fig. 5. ERHA Efficiency - Execution time with CPU limiting enabled: 1 PU per VCPU

decomposition problem in case with 2 VCPUs and 5.9% for heat transfer problem in case with 1 VCPU.

The test confirmed the results of the second experiment, by proving the efficiency of the proposed solution to reduce the VMs performance variability commonly imposed by different PMs models. Furthermore, despite the execution time being larger when executing the applications with 4 VCPUs and 2 PUs per VCPU than in the case without CPU limitation, it is important to highlight that the more powerful processors still

 ISBN: 978-1-61208-216-5

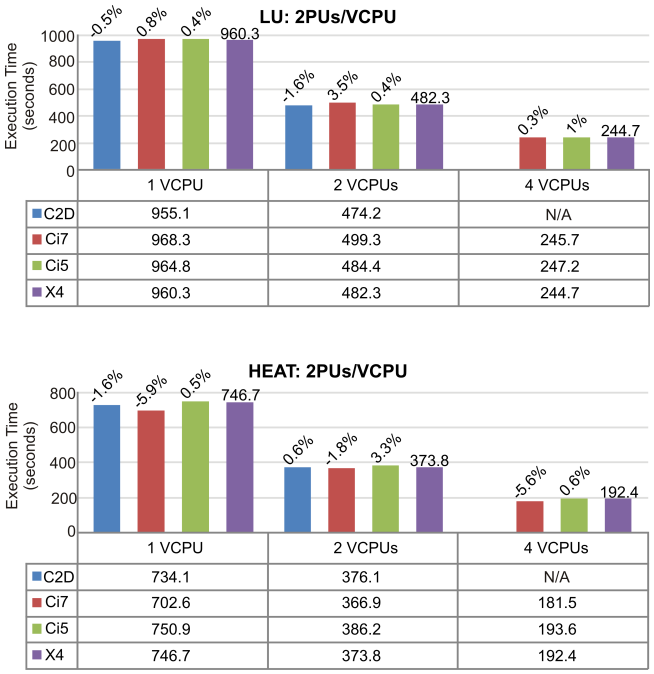


Fig. 6. ERHA Efficiency - Execution time with CPU limiting enabled: 2 PUs per VCPU

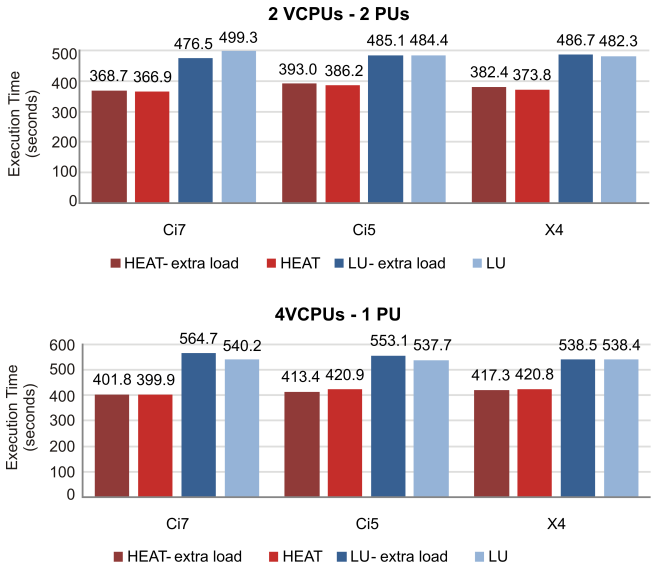


Fig. 7. Execution time using different loads in physical machines

have resources enough to run more VMs (2 PUs free in Ci7), allowing other applications to run in that PM.

B. Impact of Physical Machines Load

The objective of this experiment is to evaluate the efficiency of the architecture to provide performance isolation while running more than one VM per PM. In this experiment, two VMs were executed at the same time in each PM (Ci5, Ci7 and X4), one running the heat transfer application and the other running the LU reduction problem.

Two different configurations of VMs were used: VMs with 2 VCPUs and 4 PUs (2 PUs for each VCPU) and VMs with 4 VCPUs and 4 PUs (1 PU for each VCPU). Considering that two VMs will run on each PM, a total of 8 PUs will be used. The results can be observed in Figure 7, where the cases marked with *-extra load* represent the tests where 2 VMs are used in the same PM (competing for CPU resources). The results were compared with the previous experiments results (Figures 5 and 6), where a single VM per PM was used.

It can be observed that the largest differences between the execution time were 4,6% and 4,5% in the VMs with 2 and 4 VCPUs, respectively, both running the LU decomposition application in host Ci7. These results show that the solution provides a good performance isolation for 2 VMs running on the same PM. Further experiments must be performed to prove the ERHA's efficiency regardless the number of VMs per PM.

To sum up, the presented experiments were executed and configured easily with ERHA. The results demonstrate the solution's efficiency to deploy applications in clouds and to reduce the performance fluctuations despite being in a dynamic and heterogeneous environment.

V. Conclusion and Future Work

This paper presented a solution to automate the deployment and execution of batch applications in clouds, providing mechanisms to create homogeneous virtual environments over private cloud middleware. The results described in Section IV confirmed the ERHA's efficiency in deploying applications in clouds and reducing the disparities in execution time using different PMs.

Although the presented tests have just used OpenMP applications, ERHA allows to run sequential, shared-memory and distributed memory parallel applications. For example, it is possible to run MPI applications in a homogeneous virtual cluster with all credentials for SSH communications, automatically created and managed by the architecture.

The architecture is useful in the cases where the researchers expect comparable performance for their applications, independent of the physical resources used. This feature is quite important for repeatability of experiments and results. Furthermore, considering the uniform processing power allocations provided by ERHA, it is possible to reduce the performance variability in VMs' migrations between different physical machine types.

To conclude, the main contribution of this paper was the reduction of the impact of the data center heterogeneity in the VMs performance. In addition, we proposed mechanisms to enable running applications in clouds in an easy and uniform way, abstracting the infrastructure and middleware complexities.

The next step in our research is to extend the architecture implementation to work with other cloud middleware and new application types such as MapReduce. We also intend to create new scheduling policies and implement an interface to permit users to dynamically change the amount of PUs and VCPUs allocated to VMs.

 ISBN: 978-1-61208-216-5

ACKNOWLEDGMENTS

This work is supported by FUNCAP, CAPES and INCT-MACC (CNPq process 573710/2008-2).

REFERENCES

[1] G. V. Mc Evoy, B. Schulze, and E. L. M. Garcia, "Performance and deployment evaluation of a parallel application on a private cloud," *Conc. and Comp.: Prac. and Exp.*, vol. 23, pp. 2048–2062, Dec. 2011.

[2] C. Evangelinos and C. N. Hill, "Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazon's ec2." *October*, vol. 2, no. 2.40, pp. 2–34, 2008.

[3] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl, "Scientific cloud computing: Early definition and experience," in *Proc. of the 2008 10th IEEE Intl Conf. on High Perf. Comp. and Comm.*, ser. HPCC '08. IEEE Computer Society, 2008, pp. 825–830.

[4] J.-S. Vöckler, G. Juve, E. Deelman, M. Rynge, and B. Berriman, "Experiences using cloud computing for a scientific workflow application," in *Proc.of the 2nd Intl Workshop on Scientific cloud computing*, ser. ScienceCloud '11. ACM, 2011, pp. 15–24.

[5] Y. Simmhan, C. Van Ingen, G. Subramanian, and J. Li, "Bridging the gap between desktop and the cloud for escience applications," *IEEE 3rd Intl Conf. on Cloud Computing*, pp. 474–481, 2010.

[6] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: observing, analyzing, and reducing variance," *Proc. VLDB Endow.*, vol. 3, pp. 460–471, September 2010.

[7] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *IEEE/ACM 11th Intl Symposium on Cluster, Cloud and Grid Computing*, may 2011, pp. 104–113.

[8] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance analysis of high performance computing applications on the amazon web services cloud." in *Proc. of IEEE Intl Conf. on Cloud Computing Technology and Science*, 2010, pp. 159–168.

[9] Y. El-Khamra, H. Kim, S. Jha, and M. Parashar, "Exploring the performance fluctuations of hpc workloads on clouds," in *Proc. IEEE Second Intl Conf. on Cloud Computing Technology and Science*, 30 2010-dec. 3 2010, pp. 383–387.

[10] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforcing performance isolation across virtual machines in xen," in *Proc. of the ACM/IFIP/USENIX 2006 Intl Conf. on Middleware*, ser. Middleware '06. Springer-Verlag New York, Inc., 2006, pp. 342–362.

[11] W. Dawoud, I. Takouna, and C. Meinel, "Elastic vm for cloud resources provisioning optimization," in *Advances in Comp. and Comm.*, ser. Comm. in Comp. and Information Science, A. Abraham, J. Lloret Mauri, J. F. Buford, J. Suzuki, and S. M. Thampi, Eds. Springer Berlin Heidelberg, 2011, vol. 190, pp. 431–445.

[12] A. Sangpetch, A. Turner, and H. Kim, "How to tame your vms: an automated control system for virtualized services," in *Proce. of the 24th Intl Conf. on Large installation system administration*, ser. LISA'10. USENIX Association, 2010, pp. 1–16.

[13] C. Bunch, N. Chohan, C. Krintz, and K. Shams, "Neptune: a domain specific language for deploying hpc software on cloud platforms," in *Proc. of the 2nd Intl workshop on Scientific cloud computing*, ser. ScienceCloud '11. ACM, 2011, pp. 59–68.

[14] J. Mirkovic, T. Faber., P. Hsieh, G. Malayandisamu, and R. Malavia, "Dadl: Distributed application description language," USC/ISI, Tech. Rep. ISI-TR-664, 2010.

[15] Nimbus Project, http://www.nimbusproject.org/, [retrieved: june, 2012].

[16] T. Deshane, Z. Shepherd, J. Matthews, M. Ben-Yehuda, A. Shah, and B. Rao, "Quantitative comparison of Xen and KVM," in *Xen summit*. Berkeley, CA, USA: USENIX association, Jun. 2008.

[17] P. A. L. Rego, E. F. Coutinho, D. G. Gomes, and J. N. de Souza, "Faircpu: Architecture for allocation of virtual machines using processing features," in *Proc. of Fourth IEEE Intl Conf. on Utility and Cloud Computing*, Dec 2011, pp. 371 –376.

[18] ERHA Project, http://www.inf.ufpr.br/ggalante/erha, [retrieved: june, 2012].

[19] J. H. Lienhard and J. H. Lienhard, *A Heat Transfer Textbook - 3rd ed.* Phlogiston Press: Cambridge, Massachusetts, 2008.

[20] D. Poole, *Linear Algebra: A Modern Introduction*. Cengage Learning, 2006.

 ISBN: 978-1-61208-216-5

Cloud based Dynamically Provisioned Multimedia Delivery: An Elastic Video Endpoint

Alistair Blair, Gerard Parr, Philip Morrow, Bryan Scotney and Aaron McConnell
India-UK Centre of Excellence for Next Generation Networks
Faculty of Computing and Engineering, University of Ulster,
Coleraine, Northern Ireland
blair-a6@email.ulster.ac.uk,{gp.parr, pj.morrow, bw.scotney,a.mcconnell}@ulster.ac.uk

Steve Appleby and Mike Nilsson
Video Delivery Research,
BT Innovate & Design, Adastral Park,
Ipswich, England
{steve.appleby, mike.nilsson}@bt.com

***Abstract*—Content Delivery Networks are commonplace in today's Internet and are an important technique in the distribution of multimedia content to the plethora of Internet Protocol enabled devices. However, it has been recognised that current networks are many times over provisioned server side for peak demand and therefore greatly under utilised at other times. The emergence of cloud computing as a commercial reality has created the opportunity where content delivery networks can leverage the resources of existing cloud providers to increase capacity when required. In this paper, we propose an Elastic Video Endpoint (EVE), a virtualised multimedia distribution resource, which can utilise cloud resources to dynamically provision capacity in real time. Initial results have shown that the system can respond to increased load and provide extra bandwidth capacity on demand.**

Keywords-cloud; elastic; content delivery network; dynamic provisioning.

I. INTRODUCTION

It has been predicted that video traffic will account for around 90% of the 966 exabytes of global Internet Protocol traffic that will cross the globe in 2015 [1]. The high bandwidth and strict Quality of Service (QoS) requirements such as lower start up delay, reduced end-to-end delay and higher continuity of multimedia, creates many challenges in the area of content management and content delivery across the Internet. Degradation of any of the above factors can have adverse effects on a user's Quality of Experience (QoE), which in turn can lead them to complain or change the service provider they are using.

In an attempt to combat these challenges, clusters of machines connected to the Internet, containing replica data can be strategically placed at various geographic locations to improve dissemination performance. These Content Delivery Networks (CDN) [2], [3] offer a good way to decrease core network bandwidth, reduce network latency and lower delivery costs.

Typically a CDN will carry out the following functions [4]:

- Performs redirection of connection requests to the nearest suitable surrogate server, when a user attempts to download content;
- Provides the ability to deliver various content from a set of surrogate servers that are placed at various geographic locations;
- Perform content outsourcing to control the content that is stored on the surrogate servers that form the CDN and how it is replicated from the source server;
- Provides management services that monitor and store data on requests, cache hit/misses and accounting of content usage.

There are a number of variants of these commercially available content delivery networks and these can be categorised as:

- Highly distributed, e.g., Akamai [5], rent or place servers in the data centres of many Internet Service Providers (ISPs) around the world;
- Big Data, e.g., Limelight [6], build and run their own data centres around the world;
- P2P Assisted, e.g., Bittorrent [7], share content in a collaborative from many different sources, users, web caches and proxies;
- Cloud, e.g., Amazon CloudFront [8], enables content providers to provision capacity from Amazons cloud resources in a pay as you go manner;

However, it has been noted that content delivery networks, in any guise, are many times over provisioned and therefore under utilised [9]–[15]. This over provisioning means that that the CDN infrastructure is expensive to implement and manage, [16], [17]; however, it is required due to the lack of overload protection, so that flash crowds can be dealt with effectively. To reduce this provisioning, would increase the risk of lowering the end user experience. The work of Sun et al. [18] has shown that "10% of connections are server-limited at least 40% of the time." Cloud CDNs are a new and emerging approach [12], [19]–[22], that use cloud resources, namely cloud storage to reduce the cost associated with implementing content delivery services. However, again, these resources are created at different locations across multiple clouds and can lead to over provisioning due to the lack of overload protection.

 ISBN: 978-1-61208-216-5

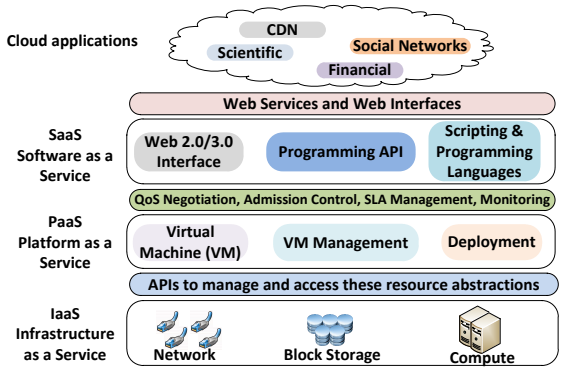


Fig. 1: Cloud Computing layered architecture

The system proposed in this paper uses metrics (CPU, Memory, Disk, Network), obtained from physical machines and hosted Virtual Machines (VMs) in an attempt to dynamically provision resources namely bandwidth when it is required, therefore increasing utilisation while minimising provisioning cost. The paper focuses mainly on Network and CPU metrics. The elastic nature of Cloud resources offer the ability to dynamically provision resources when they are required, this in turn enables resources at a single location to grow when needed to allow higher utilisation across any provisioned hardware.

The remainder of this paper is organised as follows: Section II gives a brief overview of cloud computing and its layered architecture. Section III documents related work in the area of Content Delivery Networks, populars CDNs, integrated cloud based content delivery and details some of the challenges that are being faced. Section IV details the proposed system, and finally, Section V gives some results, discussion and future work.

II. Cloud Computing

The emergence of the cloud computing paradigm as a commercial reality has created a new landscape for Internet based computing, whereby self owned IT resources can be reduced and replaced by the computation-as-a-service model that cloud providers can offer, enabling users to reduce the Capital Expenditure (CAPEX) and Operating Expenditure (OPEX). Virtualisation has allowed cloud providers to offer computing resources (compute, storage and network) as a service that can be dynamically provisioned at multiple geographical locations when required. A cloud platform is typically made up of four distinct layers: data storage, data management, data service, and user access [23].

Fig. 1 gives a representation of the layered structure of the cloud computing architecture, which consists of four layers:

- Infrastructure as a Service (IaaS) or data storage layer provides an abstract view towards the under lying compute, storage and network allowing virtual instances of mini data centres.
- Platform as a Service (PaaS) or data management layer combines infrastructure, operating systems and application software and offers it as a utility.
- Software as a Service (SaaS) or data service layer provides software products and services as a utility that can be used on demand.
- Cloud applications or user access provides the access point of applications to the Cloud.

Due to the high QoS of requirements of applications such as real-time multimedia, cloud computing has become particularly attractive for content delivery. Typically content delivery services are operated using dedicated servers that lack the dynamic nature of cloud computing or cloud storage and fail to fully utilise the elastic abilities that cloud can offer. While the properties described above can be seen as the advantages of cloud computing, there are also some disadvantages, namely resource contention, which occurs when VMs are oversubscribed, i.e., contending for the same physical resources, leading to poor application performance, causing user experience to deteriorate. This paper considers the option of utilising cloud computing resources to operate delivery endpoints that can grow/shrink in real time, when demand for their services changes while maintaining the high levels of QoS that multimedia data requires.

III. Related Work

A. General

Much research has been carried out in the area of data locality and the methods used to disseminate multimedia data to end-users [17], [24]–[27]. There are currently two key distribution techniques used to disseminate media across the Internet, namely CDN architectures and P2P architectures. Recent work has seen attempts to utilise both techniques. TopBT [24] is a topology aware Bittorrent client that can reduce download traffic by 25% while increasing download speeds by around 15%. Alessandria et al. [25] analysed some commercial P2P video applications and found that they were able to cope with impairments caused by delay, packet loss and insufficient bandwidth. However, their study did show that when all peers where affected by bottlenecks they failed to recover. Seyyedi et al. [26], compare connected and unconnected meshes and show that the connected mesh offers significant improvements in end-to-end delay and distortion, while Kang et al. [27] define a hybrid CDN-P2P architecture that allows the CDN network to take the load during quiet periods and when busy the P2P component allows the system to compensate by enabling neighbours to distribute content, relieving stress on the CDN, Tonget al. [17], propose a new web service P2PCDN architecture to help distribute content from under provisioned servers, they believe that their architecture could be provisioned using cloud computing.

The work documented above focuses on using client devices to help in the distribution process, however, these have their weak points, which include high background traffic [28] and energy tradeoffs [9].

 ISBN: 978-1-61208-216-5

B. Popular CDNs

Akamai is currently the market leader of content delivery services, with "nearly one hundred thousand servers, deployed in 72 countries" [5]. These servers are provisioned in many different ISP data centres around the world, whereas Limelight [6] has a few large data centres placed around the globe. However, research has shown that the performance of Akamai could be maintained even if the number of their servers was reduced [29]. Akamai have recently changed their status from being a CDN provider to a cloud provider. The integration of cloud computing is affecting all content providers and at present cloud storage is a significant topic in the research community.

C. Integrated CDN

Cloud computing has created a new concept of Cloud storage, whereby large data stores can be made available to users dynamically when they are required. Cloud storage is very different from traditional storage and whereas traditional storage was of a fixed size, cloud storage has the ability to grow if required. This service is offered on a pay-as-you go basis and so costs can be controlled and managed. In terms of functionality, it is able to deliver a variety of online services, as opposed to traditional storage systems that are aimed at large scale transactional processing and high performance computing. Wu et al. [30], and Huo et al. [31] detail the advantages of cloud storage and the challenges that will face the technology while Lin et al. [19], use cloud storage as the basis of a new content delivery network called CCDN. Simulations by Wang et al. [20], have also shown that cloud storage offers a highly scalable and fault tolerant platform that offers lower delay and higher bandwidth to end users.

D. CDN Challenges

Distribution servers must be over provisioned for peak demand, due to their lack of overload protection (each instance will have limited CPU, Memory, Storage and Network bandwidth). Sun et al. [18], argue that, "passively monitoring the transport-level statistics," of a server is a much better approach, as the ability to monitor conventional metrics is much too difficult, we would argue that with the advent of virtualisation the ability to monitor core performance metrics offered by the hypervisor, e.g., CPU, Disk, Network and Disk at twenty second intervals is a novel and promising technique (Twenty seconds is the smallest interval offered by VMware). No matter how large the content storage is, bandwidth is required to disseminate the content.

In summary, the above literature makes use of traditional physical resources, while cloud computing has enabled smaller content providers to utilise delivery services that would otherwise be out of their reach, the delivery services are still over provisioned, with multiple instances running at any one location. However, cloud storage is only a minor part of the flexibility that is offered by cloud computing, by utilising cloud computing the ability to dynamically add extra capacity and real-time monitoring of an instance is possible. The ability to monitor a VM and dynamically add/remove extra capacity in real-time presents an opportunity where the foot print of a system can be kept to a minimum, therefore further reducing the financial and data cost associated with cloud based content delivery.

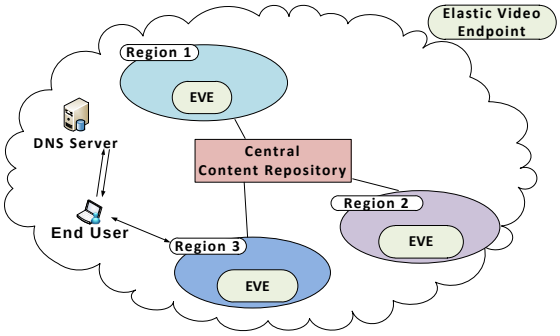


Fig. 2: A dissemination architecture consisting of multiple EVE resources

IV. System Overview

The literature review above has shown that research in the area of content delivery is a popular topic, with developments taking place in all aspects of the concept. However, more recently, the realisation of cloud computing has created a platform that removes the need of content providers to either pay for their own hardware or expensive third party distribution platforms. Content producers can now utilise cloud resources to store and distribute their content in a pay as you go manner. While this has reduced the expense for providers, it still leaves the problem of over provisioning due to the lack of overload protection within these systems. In an attempt to dynamically provision content, while still maintaining a high quality of service we propose a new delivery endpoint, based on cloud resources, "EVE," Elastic Video Endpoint. In Fig. 2, we can see an architectural overview of a distribution network consisting of multiple endpoints located in different geographical locations. A central content repository retains a copy of all the content that is available for dissemination.

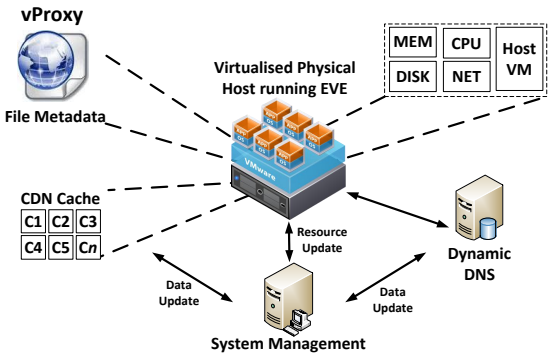


Fig. 3: Components of an Elastic Video Endpoint

From here content can be replicated to endpoints placed in different geographic regions when it is deemed necessary. Each region contains an instance of Elastic Video Endpoint that is adequately provisioned to disseminate content under current conditions. Each endpoint contains a subset of the content that is held at the central repository, which is determined by its current popularity. If the content delivery conditions change i.e., a flash crowd situation occurs where resources become contented, the endpoint can provision extra resources to facilitate the extra demand by utilising the elastic nature of cloud computing. An intelligent endpoint that can scale its resources depending on the load that the system is currently under is shown in Fig 3. The system takes a subset of content from a central master repository and stores it using cloud storage. Current CDNs are optimised for small file delivery [32], [33] and not the large files that are required to store High Definition (HD) and Super HD video. The content used is a mixture of HD and Super HD videos that range in size from a few hundred megabytes to a few gigabytes.

EVE is made up of five major components; these are the VMware Infrastructure, Dynamic DNS, a vProxy, a Cache and System Software.

(a) VMware Infrastructure
VMware infrastructure software has been chosen as the platform on, which to base the cloud platform as it is an industry standard with VMWare controlling eighty percent of the server virtualisation market [34]. The exposed API's of the software provide the ability to access all aspects of the hypervisor and its associated host enabling a custom monitoring system written in C# to be created. This monitoring system can access a large array of metrics. The specific performance metrics to be recorded are CPU, Memory, Disk and Network. These metrics cover many aspects of the system, including the physical host and all VMs running on that host. These metrics give an accurate insight into the current health of both the VM's and the host, using the VMware API's the metrics are available at twenty second intervals. Example metrics are percentage of CPU capacity and throughout of the network.

(b) Dynamic DNS
The request routing component determines the best end-point to facilitate the end users' request. If the content isn't cached locally then the user is redirected to another more suitable location by means of a DNS-based redirect. DNS based requests are highly efficient and help to reduce access time to the endpoint. This enables the system to carry out load balancing so that resource allocation at the endpoint doesn't become overloaded.

(c) vProxy
The proxy keeps a record of files that are stored locally and also of the files that are held remotely. When a cache hit or miss occurs the vProxy updates the relevant record with the information and then redirects the user the local or remote file. By keeping an accurate account of the content that is cached or needs to be cached, the system can fully utilise the space that it has.

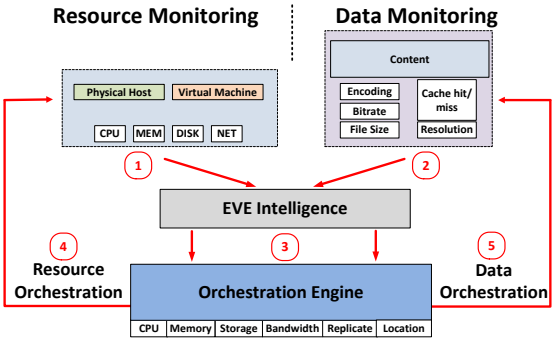


Fig. 4: EVE processes and flow

(d) Cache
The cache is an area of cloud storage that is used to store content for dissemination. The cache is mounted with in the system drive as a folder. This enables the Operating System partition to be minimal in size while the content drive can also be kept to a minimum. This allows the system footprint to remain at a minimum until such times as it needs to expand to hold extra content.

(e) System Software
The management component monitors the remote database to decide when corrective, (e.g., add an extra NIC, increase the cache size or create a new endpoint instance) should be taken to prevent an endpoint from failing. The system takes values from the monitoring database CPU, Memory, Disk and Network, when used along with the cache hits and other content metadata to determine, which file is placing the endpoint under pressure. Corrective action may include increasing capacity or temporarily redirecting future connections to another endpoint via the intelligent DNS.

In order for EVE to operate, there are a number of processes that occur in the general operation and maintenance of each instance. These processes are shown in Fig. 4:

1) Resource monitoring
Performs data acquisition for host and VM metrics for use by the optimiser of metrics such as CPU, Memory, Disk and Network.

2) Data monitoring
Performs data acquisition for use by the optimiser, about content that is hosted at the endpoint, e.g., popularity, cache hits/misses and bitrate.

3) System Intelligence
Performs analysis of the real-time metrics coming from both the data monitoring and resource monitoring processes. Using these metrics the system determines if content should be added/removed or deleted from endpoint instances. Also whether an endpoint instance requires more/less capacity, these decisions are then passed to the relevant orchestration engine for execution.

 ISBN: 978-1-61208-216-5

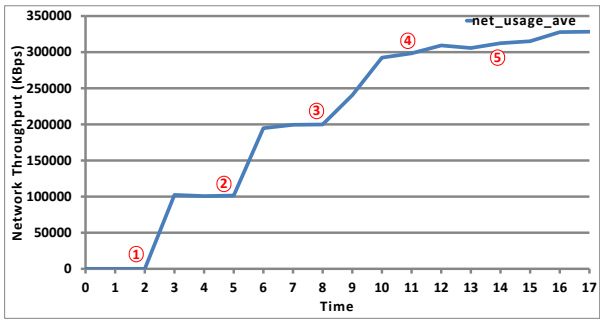


Fig. 5: Network throughput against time

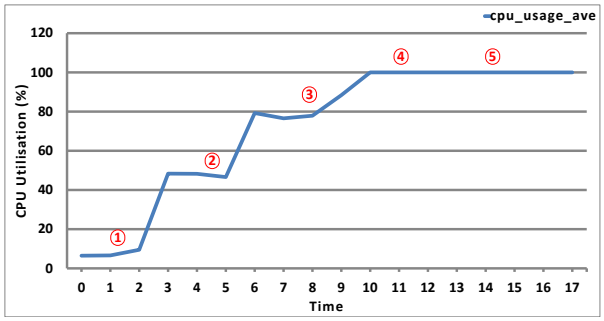


Fig. 6: VM CPU against time

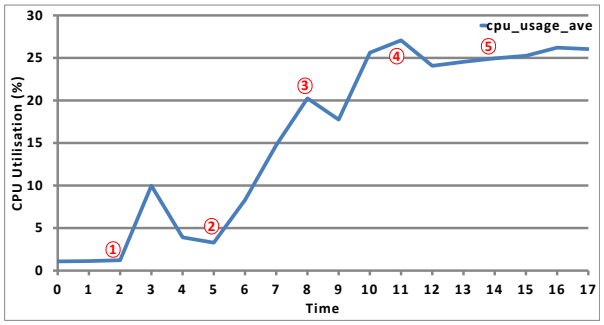


Fig. 7: Host CPU against time

4) Resource Orchestration
   Performs dynamic addition or removal of resources resources to a Virtual Machine when and if they are available. If resources are contented on the host then action to create a replica at the same or a different site can be initiated.
5) Data Orchestration
   Performs control over the content at the endpoint, to include data replication, deletion or migration, while also updating the dynamic DNS system to provide load balancing across the system.

V. Experiment and Discussion

Experimental Setup

Traditional physically hosted application servers are limited by the resources of that machine, cloud computing enables the dynamic provisioning of extra resources, e.g., bandwidth and storage when they are required. However, even with the ability to add these extra resources a point will arise when the VM is no longer able to meet demand.

The experiment is designed to show that as load on a Virtual Machine increases the resources associated with that VM will also increase until a threshold is reached where adding extra resources will have no or little effect.

In this paper we consider network bandwidth and its effect on CPU. Adding and removing NICs when required to alleviate bandwidth pressure on a VM to disseminate extra data. To demonstrate this a cloud distribution server was created using Windows 2008 R2 web edition with a configuration of 1vCPU, 1024 MB memory, one network interface card and a 40 GB thin provisioned system disk, a second 40GB is mounted for content storage, the entire VM is hosted on a DELL R515 blade server with two Opteron 8 core processors, 16GB of RAM and 12 gigabit ethernet cards, running VMware ESX 5.0.

In order generate a load on the media server, openload [35] was used generate http requests on the IP address assigned to the single NIC. As the load increased it would be expected to see an increase in the CPU utilisation of both the host and the VM. As a link becomes fully utilised, an extra virtual NIC is added and a new instance of openload created against its associated IP address. This process is repeated until a point is reached where the total throughput from the server reaches a maximum, it is expected that the graphs will show that the CPU is the limiting resource.

Discussion

The experiment results showing the network throughput are documented in Fig. 5. Baseline throughput occurs until point 1 where the first instance of openload is initiated. At this point the throughput increases until a maximum is reached, here the throughput levels out. The same point in Fig. 6 shows that the CPU follows a smiler trend. When a second NIC (2) is added and load placed on it, this again results in an increase of throughput with a corresponding increase in the VM CPU. When a third NIC (3) is added the increase achieves a similar addition to the total but takes slightly longer to reach its maximum. This increase in time can be attributed to the CPU reaching 100% utilisation Fig. 6 point (4). Fig. 5 shows extra vNICs being added at point (4) and (5), however, the total throughput levels out with little or no change, due to resource contention on the CPU. At points (2) and (3) we can notice a slight dip in the CPU, we believe this can be attributed to resource discovery as the new vNIC is added to the endpoint.

Fig. 5 and Fig. 6 deal with the Virtual Machine; however, we believe that the host must also be considered, Fig. 7 shows the physical host CPU over the same time period. the graph shows that there is some increase in CPU utilisation on the host as the VM CPU increases. Others spikes in the host CPU could be attributed to other VM's and processes that are running on the host, this information is important has it has an influence on

 ISBN: 978-1-61208-216-5

determining on whether or not resources can be dynamically added to the endpoint at times of VM contention.

VI. Conclusion and Future Work

In this paper, we implemented an initial version of an intelligent multimedia delivery endpoint based on Cloud computing infrastructure called EVE. The results show that the endpoint can provision extra capacity in real time when required, however, it can be seen that resources namely CPU have a limiting factor on the total bandwidth that can be provisioned.

Further work will aim at enhancing the endpoint, to allow the addition or deletion of extra capacity when required. The ability to implement an expanding cache when required and some prediction algorithms to predict provisioning will be developed. This work it is hoped will be detailed in future publications

Acknowledgment

The authors wish to acknowledge the funding received for this project from BT-EPSRC CASE award as part of the India-UK Advanced Technology Centre of Excellence in Next Generation Networks Systems and Services.

References

[1] Cisco visual networking index. [retrieved: April, 2012]. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf

[2] Edgecast. [retrieved: April, 2012]. [Online]. Available: http://www.edgecast.com

[3] Cachefly. [retrieved: April, 2012]. [Online]. Available: http://www.cachefly.com/

[4] Hao, "Content delivery networks: a bridge between emerging applications and future IP networks," *Network, IEEE*, vol. 24, no. 4, pp. 52–56, 2010.

[5] Akamai technologies. [retrieved: April, 2012]. [Online]. Available: http://www.akamai.com

[6] Limelight networks. [retrieved: April, 2012]. [Online]. Available: http://www.limelightnetworks.com/

[7] Bittorrent. [retrieved: April, 2012]. [Online]. Available: http://www.bittorrent.com.

[8] Amazon cloudfront. [retrieved: April, 2012]. [Online]. Available: http://aws.amazon.com/cloudfront/

[9] A. Feldmann, A. Gladisch, M. Kind, C. Lange, G. Smaragdakis, and F.-J. Westphal, "Energy trade-offs among content delivery architectures," *Telecommunications Internet and Media Techno Economics (CTTE), 2010 9th Conference on*, pp. 1–6, 2010.

[10] D. Niu, B. Li, and S. Zhao, "Understanding demand volatility in large VoD systems," in *NOSSDAV '11: Proceedings of the 21st international workshop on Network and operating systems support for digital audio and video*. ACM Request Permissions, Jun. 2011.

[11] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. H. Andrew, "Greening geographical load balancing," in *SIGMETRICS '11: Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*. ACM Request Permissions, Jun. 2011.

[12] V. Aggarwal, X. Chen, V. Gopalakrishnan, R. Jana, K. Ramakrishnan, and V. Vaishampayan, "Exploiting virtualization for delivering cloud-based IPTV services," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, 2011, pp. 637–641.

[13] F. Ramos, R. Gibbens, F. Song, P. Rodriguez, J. Crowcroft, and I. White, "Reducing energy consumption in IPTV networks by selective pre-joining of channels," *Green Networking '10: Proceedings of the first ACM SIGCOMM workshop on Green networking*, Aug. 2010.

[14] N. Xu, J. Yang, M. Needham, D. Boscovic, and F. Vakil, "Toward the Green Video CDN," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom*, 2010, pp. 430–435.

[15] I. Vaishnavi, P. Cesar, D. Bulterman, and O. Friedrich, "From IPTV services to shared experiences: Challenges in architecture design," *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pp. 1511–1516, 2010.

[16] Z. Lu, J. Wu, and W. Fu, "Towards a Novel Web Services Standard-Supported CDN-P2P Loosely-Coupled Hybrid and Management Model," in *Services Computing (SCC), 2010 IEEE International Conference on*, 2010, pp. 297–304.

[17] J. Tong, K. Xu, and R. Pi, "A new Web Service Structure of Combining P2P and CDN Technologies," *Web Society (SWS), 2010 IEEE 2nd Symposium on*, pp. 475–479, 2010.

[18] P. Sun, M. Yu, M. J. Freedman, and J. Rexford, "Identifying performance bottlenecks in CDNs through TCP-level monitoring," in *W-MUST '11: Proceedings of the first ACM SIGCOMM workshop on Measurements up the stack*. ACM Request Permissions, Aug. 2011.

[19] C.-F. Lin, M.-C. Leu, C.-W. Chang, and S.-M. Yuan, "The Study and Methods for Cloud Based CDN," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*, 2011, pp. 469–475.

[20] Y. Wang, X. Wen, Y. Sun, Z. Zhao, and T. Yang, "The Content Delivery Network System Based on Cloud Storage," in *Network Computing and Information Security (NCIS), 2011 International Conference on*, 2011, pp. 98–102.

[21] H. A. Tran, A. Mellouk, and S. Hoceini, "QoE Content Distribution Network for Cloud Architecture," *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*, pp. 14–19, 2011.

[22] Y. Wang, C. Huang, J. Li, and K. Ross, "Estimating the performance of hypothetical cloud service deployments: A measurement-based approach," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 2372–2380.

[23] R. Xue, Z.-S. Wu, and A.-N. Bai, "Application of Cloud Storage in Traffic Video Detection," in *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, 2011, pp. 1294–1297.

[24] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Zhang, "TopBT: A Topology-Aware and Infrastructure-Independent BitTorrent Client," *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, 2010.

[25] E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, and M. Meo, "P2P-TV Systems under Adverse Network Conditions: A Measurement Study," *INFOCOM 2009, IEEE*, pp. 100–108, 2009.

[26] S. Seyyedi and B. Akbari, "Hybrid CDN-P2P architectures for live video streaming: Comparative study of connected and unconnected meshes," *Computer Networks and Distributed Systems (CNDS), 2011 International Symposium on*, pp. 175–180, 2011.

[27] S. Kang and H. Yin, "A Hybrid CDN-P2P System for Video-on-Demand," *Future Networks, 2010. ICFN '10. Second International Conference on*, pp. 309–313, 2010.

[28] P. Shi, H. Wang, Y. Gang, and X. Yuan, "ACON: Adaptive construction of the overlay network in CDN-P2P VoD system," in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, 2011, pp. 182–187.

[29] S. Triukose, Z. Wen, and M. Rabinovich, "Content delivery networks: how big is big enough?" *SIGMETRICS Performance Evaluation Review*, vol. 37, no. 2, Oct. 2009.

[30] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud Storage as the Infrastructure of Cloud Computing," in *Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on*, 2010, pp. 380–383.

[31] Y. Huo, H. Wang, L. Hu, and H. Yang, "A Cloud Storage Architecture Model for Data-Intensive Applications," in *Computer and Management (CAMAN), 2011 International Conference on*, 2011, pp. 1–4.

[32] X. Guan and B.-Y. Choi, "Push or Pull?: Toward Optimal Content Delivery," in *Communications (ICC), 2011 IEEE International Conference on*, 2011, pp. 1–5.

[33] S. Borst, V. Gupta, and A. Walid, "Distributed Caching Algorithms for Content Distribution Networks," *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, 2010.

[34] S. D. Burd, G. Gaillard, E. Rooney, and A. F. Seazzu, "Virtual computing laboratories using vmware lab manager," in *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, ser. HICSS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1–9. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2011.482

[35] P. Johnsen. Openload. [retrieved: April, 2012]. [Online]. Available: http://freecode.com/projects/openload

 ISBN: 978-1-61208-216-5

"cocoBox": A Social File Cloud System for Collaboration

Ki-Sook Chung, Hyunjoo Bae

Future Internet Service Research Team,
Electronics and Telecommunications Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, KOREA
{kschung, hjbae}@etri.re.kr

Abstract— **File cloud provides file storages on the Internet and manages them for people to access and manipulate their contents like documents, pictures, and movies anywhere and anytime. There are various smart devices such as tablets, smart phones, and smart pads, which can utilize file cloud services. In this paper, we introduce a social file cloud system as one of tools for collaboration between people in the workplace. We extend basic file cloud service by adding social factors such as tags, score, and comments to a file into a social file cloud service, that is, "cocoBox", which means a file box for communication and collaboration. CocoBox provides the basic functionalities of file cloud service such as file upload and file download. In addition, we focus on social factors of files to help collaboration among people who share same files. Whoever shares a specific file with others can add tags, give score, and add/remove his/her opinion on that file. Therefore, cocoBox enhances communication and collaboration among people with these social factors.**

Keywords-file cloud; REST; service component; score; tag; comment; collaboration

I. INTRODUCTION

Cloud computing is defined as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]". There are several service models for cloud computing such as SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service).

File cloud provides file storages on the Internet and manages them for people to access and manipulate their files anywhere and anytime. Recently, we have various kinds of mobile devices such as laptops, mobile pads, and smart phones to use file cloud services. File cloud services enable us to access same files on any devices and share various contents like pictures, movie, and music to other people. As the examples of file cloud, dropbox [3], Amazon S3 [4], and iCloud [5] are popular. These services provide file storage service with user friendly interfaces on desktops, web browsers, and mobile internet devices and enable file sharing among people.

We extend file cloud service by adding social features such as tags, score, and comments to ordinary content repository into a social file cloud service. Based on the Java Content Repository (JCR) 170 [6], we develop "cocoBox", which means a file box for communication and collaboration. CocoBox provides the basic functionalities of file cloud service such as file sharing, uploading and downloading. But, in addition to it, cocoBox has social features for collaboration between people who share the same files. Whoever shares a specific file with others can add tags, give score, and add/remove his/her opinion on that file. These social factors of the file give people additional information of it; therefore, these social values can help to promote the collaboration among colleagues who share common files.

This paper is organized as follows. In Section 2, we overlook recent popular file cloud services such as iCloud, S3, dropbox. Section 3 shows the architecture, data models of cocoBox including social features. In addition, REST APIs of cocoBox and cocoBox applications implemented by using the APIs are introduced. Finally, we summarize and describe further works of this study in Section 4.

II. RELATED WORK

With the bombing growth of number of smart phone users and mobile internet devices, the need to share contents such as pictures, movies, and music with other people also grows. It is required to provide file cloud services for users to access their files on various devices. A lot of file cloud services are developed and provided. In this section, we outlook on worldwide popular file cloud services, e.g., iCloud, Amazon S3, dropbox.

A. iCloud

iCloud is a cloud storage and cloud computing service from Apple Inc. announced on June 6, 2011 at the Apple Worldwide Developers Conference (WWDC). The service allows users to store data such as music files on remote computer servers for download to multiple devices such as iOS-based devices, and personal computers running Mac OS X or Microsoft Windows. It also replaces Apple's MobileMe service, acting as a data syncing center for email, contacts, calendars, bookmarks, notes, to-do lists, and other data. As of 2012, the service has over 100 million users [2].

iCloud stores music, photos, documents, and more and wirelessly pushes them to devices. iCloud is said to makes it quick and effortless to access just about everything on the devices people use every day. iCloud automatically and securely stores content so it's always available to iPhone, iPad, iPod touch, Mac, or PC and gives people access to their music, movies, apps, latest photos, and more from whichever device people happen to be using. It also keeps

 ISBN: 978-1-61208-216-5

email, contacts, and calendars up to date across all devices without explicit syncing and management.

B. *Dropbox*

Dropbox is a web-based file hosting service operated by Dropbox, Inc. that uses networked storage to enable users to store and share files and folders with others across the Internet using file synchronization [2].

Dropbox is a free service that lets users bring photos, docs, and videos anywhere. This means that any file users save to Dropbox will automatically save to their computers, phones and the Dropbox website [3].

C. *Amazon S3*

Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers [2]. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

III. COCOBOX: A SOCIAL FILE CLOUD

Other file cloud services do not focus on social features, which can help people express their opinion on the sharing files or rank them. In this section, we describe the architecture of cocoBox and social features, which cocoBox provides to promote collaboration while sharing files.

A. *System Architecture*

"cocoBox" means a file box for communication and collaboration. In the previous section, most popular used services don't have social features which help people to collaborate, that is, to communicate their opinions on the files and evaluate them.

Figure 1 shows the overall architecture of cocoBox system. CocoBox is implemented based on the JCR (java content repository) 170 as a repository. The cocoBox server manipulates requests of users which are called from web browser on user's desktop or mobile internet device like a smart phone. Since cocoBox is focusing on collaboration between people, the main target domain would be small or middle – size enterprise. Within a closed group such as divisions, teams and departments, people share files and contents with their colleagues. To support this closed group collaboration, cocoBox interacts with a directory server which manages organization chart and member information using LDAP protocol.

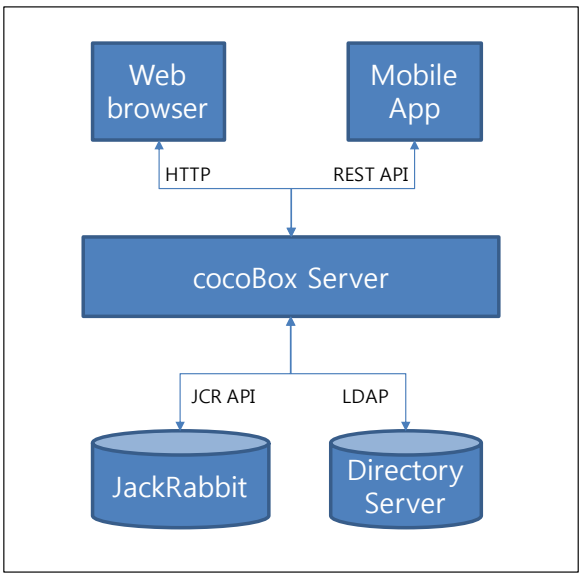


Figure 1. System Architecture of cocoBox

CocoBox also provides service components in the format of REST and enables application developers make their own applications easily. We developed a mobile application for cocoBox by using these components.

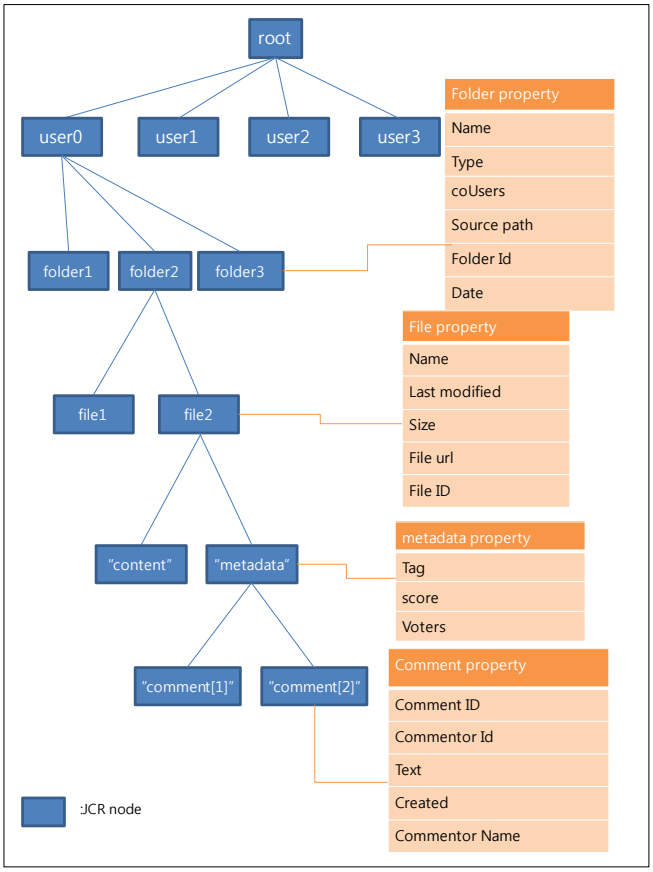


Figure 2. Data model of cocoBox

The data model of cocoBox is shaped into a tree structure as shown in Figure 2. This model supports a parent-child relationship between folders and files. Each file and folder is a node, which has properties respectively.

- Folder node
  This represents a folder, and includes basic folder properties and sharing information, and has subfolders and file nodes as its child. The folder node can have one of three types: sharing folder, shared folder, and personal folder type. If the folder is sharing folder, this means that its owner is the user, and it has coUsers properties ,which mean co-workers who share this folder together. If the folder is shared folder type, it has no children and it has source path property which is original owner's folder path. Otherwise, the folder is a personal folder, which nobody can share
- File node
  File node has contents of the file and metadata as it children and has general information as its properties.
- Metadata node
  Metadata and its children represent the social features of cocoBox. Metadata include tags, score, comments and related information.
- Comment node
  Comment node represents a comment and it has content of comment and commentator information. Only the writer of the comment can remove that comment.

To summarize, the social features in the cocoBox system are follows;

- Score
  The quality of document would be estimated using this score. People can score each file on a scale of 0 to5.
- Tag
  This tag information could be used as keywords. Since social data are in the data tree, people can search files which have the specific tag.
- Comment
  People can add their opinion about this file in the short sentence, share their thinking, and even discuss it.

With these social factors of files, file sharers can rank their files and express their opinion about sharing files. Therefore people can discuss on the shared documents and even share their knowledge.

B. cocoBox service components

We provide cocoBox service components in the form of REST API. Not only core functionalities to manipulate file storage, but also additional functionalities to manage social metadata of the file are provided. Using these APIs, people can develop their file cloud applications easily which use cocoBox. These components provide simple interfaces to create/delete/share folders, upload/download files, and add/remove social metadata of files such as comments, tags, score, as shown in the Table 1.

TABLE I. COCOBOX SERVICE COMPONENTS

REST API	Function	HTTP method
http://{serveRoot}/cbox/login	login	POST
http://{serveRoot}/cbox/logout	logout	POST
http://{serveRoot}/cbox/{userId}/userInfo	get user info	GET
http://{serveRoot}/cbox/{userId}/folderInfo	get folder info	GET
http://{serveRoot}/cbox/{userId}/moverFile	move file	POST
http://{serveRoot}/cbox/{userId}	delete file	DELETE
http://{serveRoot}/cbox/{userId}/filename	change filename	POST
http://{serveRoot}/cbox/{userId}/file	upload file	POST
http://{serveRoot}/cbox/{userId}/file	get file	GET
http://{serveRoot}/cbox/{userId}/folder	create folder	POST
http://{serveRoot}/cbox/{userId}/folder	delete folder	DELETE
http://{serveRoot}/cbox/{userId}/file/meta	get meta info	GET
http://{serveRoot}/cbox/{userId}/tag http://{serveRoot}/cbox/{userId}/score http://{serveRoot}/cbox/{userId}/comment http://{serveRoot}/cbox/{userId}/comment	modify meta info	POST POST POST DELETE
http://{serveRoot}/cbox/{userId}/fileUrl	get file URL	GET
http://{serveRoot}/cbox/{userId}/search	search file	GET
http://{serveRoot}/cbox/{userId}/history	get history	GET
http://{serveRoot}/cbox/{userId}/folderInfo	share folder	POST
http://{serveRoot}/cbox/{userId}/folder/addUsers	invite users	POST

As an example, we implemented a cocoBox mobile application using these service components.

C. cocoBox applications

We can use cocoBox services through web browser and mobile internet devices.

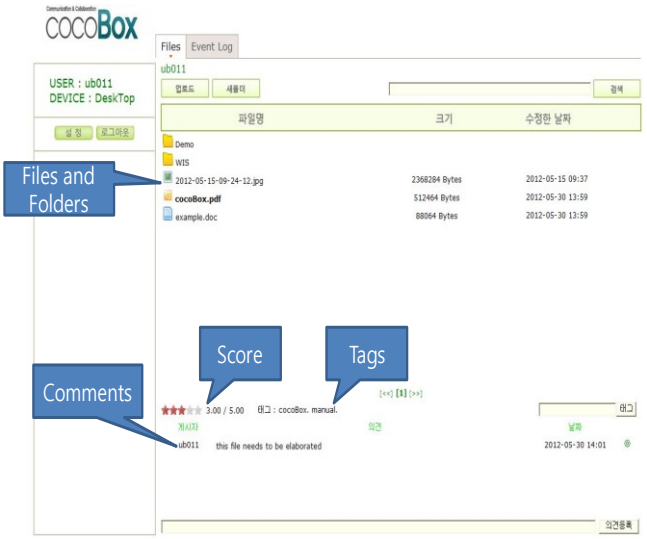


Figure 3. cocoBox web application

Figure 3 shows the cocoBox web user interface. When you select a file, the social data such as tag, average score, and comments are displayed in the page and you can add or modify them.

Figure 4 shows the home display of mobile cocoBox app. This application was developed using the cocoBox service components shown in Table 1and runs on smart mobile devices the Oss of which are Android 2.2.

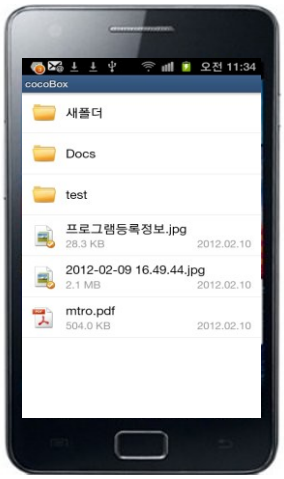


Figure 4. cocoBox mobile application

Figure 5 shows the social data of the mobile cocoBox app. People can handle social data of a file using this interface. By providing these social data in file cloud service, people can discuss their sharing contents and express their thought or preference about it. We expect these social data of file cloud could help the collaboration and even communicating their knowledge with other people.

Figure 5. Social data on the cocoBox app: score, tags, comments on the file

IV. Conclusion and Future Work

We introduce a social file cloud system, cocoBox. We extend this file cloud service by adding social features such as tags, score, and comments to ordinary content repository into a social file cloud service. The social features of cocoBox are tags, comments, score on the file. We expect these features can help collaboration and communication between people who share contents and have same interest. For further study, we continue to find more social features of file cloud such as e-mail notification of file changes and to develop desktop client which synchronizes files with cocoBox server. We expect improved cocoBox will help people to collaborate in the work environment.

Acknowledgments

This research was supported by the KCC (Korea Communications Commission), Korea, under the Development of Customer Oriented Convergent Service Common Platform Technology based on Network support program supervised by the KCA(Korea Communications Agency) (KCA-2012- (09913-05001)).

References

[1] "The NIST Definition of Cloud Computing". National Institute of Science and Technology. Retrieved 24 May 2012.

[2] http://en.wikipedia.org/wiki/ICloud , Retrieved 19 March 2012

[3] http://www.dropbox.com , Retrieved 24 May 2012

[4] http:// aws.amazon.com/s3/, Retrieved 24 May 2012

[5] https://www.icloud.com/, Retrieved 19 March 2012

[6] Content Respository API for Java Technology Specification, Java Specification Request 170, version 1.0

[7] http://jackrabbit.apache.org, Retrieved 19 March 2012

[8] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, and Thomas Sandholml, "What's Inside the Cloud? An Architectural Map of the Cloud Landscape", Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, pp. 23-31, 2009

[9] Ki-sook Chung and Sangki Kim, "A study on the application lifecycle management over SOA based application hosting platform", Proceedings of ICACT2010, pp. 310-314, 2010

[10] Ki-sook Chung and Young-mee Shin, " Service Components for Unified Communication and Collaboration of an SOA-based Converged Service Platform", HCII2011, CCIS 173, pp. 491-495, 2011

 ISBN: 978-1-61208-216-5