CLOUD COMPUTING 2016

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

ISBN: 978-1-61208-460-2

March 20 - 24, 2016

Rome, Italy

CLOUD COMPUTING 2016 Editors

Carlos Becker Westphall, Federal University of Santa Catarina, Brazil

Yong Woo Lee, University of Seoul, Korea

Stefan Rass, Universitaet Klagenfurt, Institute of Applied Informatics, Austria

CLOUD COMPUTING 2016

Forward

The Seventh International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2016), held between March 20-24, 2016 in Rome, Italy, continued a series of events targeted 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 fix them, especially on security, privacy, and inter- and intra-clouds protocols.

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 conference had the following tracks:

- Cloud computing
- Computing in virtualization-based environments
- Platforms, infrastructures and applications
- Challenging features

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the CLOUD COMPUTING 2016 technical program committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to CLOUD COMPUTING 2016. 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 also gratefully thank the members of the CLOUD COMPUTING 2016 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope that CLOUD COMPUTING 2016 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of cloud computing, GRIDs and virtualization. We also hope that Rome provided a pleasant

environment during the conference and everyone saved some time for exploring this beautiful city.

CLOUD COMPUTING 2016 Chairs

CLOUD COMPUTING 2016 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
Alain April, École de Technologie Supérieure - Montreal, Canada
Christoph Reich, Furtwangen University, Germany

CLOUD COMPUTING 2016 Industry/Research Chairs

Wolfgang Gentzsch, The UberCloud, Germany
Tony Shan, Keane Inc., USA
Anna Schwanengel, Siemens AG, Germany
Atsuji Sekiguchi, Fujitsu Laboratories Ltd., Japan

COULD COMPUTING 2016 Special Area Chairs

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

GRID
Jorge Ejarque, Barcelona Supercomputing Center, Spain
Javier Diaz-Montes, Rutgers University, USA
Nam Beng Tan, Nanyang Polytechnic, Singapore

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

Platforms
Arden Agopyan, ClouadArena, Turkey

CLOUD COMPUTING 2016

Committee

CLOUD COMPUTING Advisory Committee

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Wolf Zimmermann, Martin-Luther University Halle-Wittenberg, Germany
Yong Woo Lee, University of Seoul, Korea
Alain April, École de Technologie Supérieure - Montreal, Canada
Christoph Reich, Furtwangen University, Germany

CLOUD COMPUTING 2016 Industry/Research Chairs

Wolfgang Gentzsch, The UberCloud, Germany
Tony Shan, Keane Inc., USA
Anna Schwanengel, Siemens AG, Germany
Atsuji Sekiguchi, Fujitsu Laboratories Ltd., Japan

COULD COMPUTING 2016 Special Area Chairs

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

GRID
Jorge Ejarque, Barcelona Supercomputing Center, Spain
Javier Diaz-Montes, Rutgers University, USA
Nam Beng Tan, Nanyang Polytechnic, Singapore

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

Platforms
Arden Agopyan, ClouadArena, Turkey

CLOUD COMPUTING 2016 Technical Program Committee

Jemal Abawajy, Deakin University - Victoria, Australia
Imad Abbadi, University of Oxford, UK
Taher M. Ali, Gulf University for Science & Technology, Kuwait

Uwe Hohenstein, Siemens AG, Germany
Luigi Lo Iacono, Cologne University of Applied Sciences, Germany
Shadi Ibrahim, INRIA Rennes - Bretagne Atlantique Research Center, France
Yoshiro Imai, Kagawa University, Japan
Anca Daniela Ionita, University "Politehnica" of Bucharest, Romania
Xuxian Jiang, North Carolina State University, USA
Eugene John, The University of Texas at San Antonio, USA
Carlos Juiz, Universitat de les Illes Balears, Spain
Verena Kantere, University of Geneva, Switzerland
Bill Karakostas, VLTN gcv, Antwerp, Belgium
Sokratis K. Katsikas, University of Piraeus, Greece
Takis Katsoulakos, INLECOM Systems, UK
Zaheer Khan, University of the West of England, UK
Prashant Khanna, JK Lakshmipat University, Jaipur, India
Shinji Kikuchi, Fujitsu Laboratories Ltd., Japan
Peter Kilpatrick, Queen's University Belfast, UK
Tan Kok Kiong, National University of Singapore, Singapore
William Knottenbelt, Imperial College London - South Kensington Campus, UK
Sinan Kockara, University of Central Arkansas, USA
Joanna Kolodziej, University of Bielsko-Biala, Poland
Kenji Kono, Keio University, Japan
Dimitri Konstantas, University of Geneva, Switzerland
Arne Koschel, Hochschule Hannover, Germany
George Kousiouris, National Technical University of Athens, Greece
Sotiris Koussouris, National Technical University of Athens, Greece
Kenichi Kourai, Kyushu Institute of Technology, Japan
Nane Kratzke, Lübeck University of Applied Sciences, Germany
Heinz Kredel, Universität Mannheim, Germany
Hans Günther Kruse, Universität Mannheim, Germany
Yu Kuang, University of Nevada Las Vegas, USA
Alex Kuo, University of Victoria, Canada
Tobias Kurze, Karlsruher Institut für Technologie (KIT), Germany
Dharmender Singh Kushwaha, Motilal Nehru National Institute of Technology - Allahabad, India
Dimosthenis Kyriazis, University of Piraeus, Greece
Giuseppe La Torre, University of Catania, Italy
Romain Laborde, University Paul Sabatier, France
Petros Lampsas, Central Greece University of Applied Sciences, Greece
Erwin Laure, KTH, Sweden
Alexander Lazovik, University of Groningen, The Netherlands
Craig Lee, The Aerospace Corporation, USA
Yong Woo Lee, University of Seoul. Korea
Grace Lewis, CMU Software Engineering Institute - Pittsburgh, USA
Jianxin Li, Beihang University, China
Kuan-Ching Li, Providence University, Taiwan
Maozhen Li, Brunel University - Uxbridge, UK
Dan Lin, Missouri University of Science and Technology, USA
Wei-Ming Lin, University of Texas at San Antonio, USA
Panos Linos, Butler University, USA

Ruppa Thulasiram, University of Manitoba, Canada
Raul Valin, Swansea University, UK
Carlo Vallati, University of Pisa, Italy
Geoffroy R. Vallee, Oak Ridge National Laboratory, USA
Luis Miguel Vaquero-Gonzalez, Hewlett-Packard Labs Bristol, UK
Michael Vassilakopoulos, University of Thessaly, Greece
Jose Luis Vazquez-Poletti, Universidad Complutense de Madrid, Spain
Luís Veiga, Instituto Superior Técnico - ULisboa / INESC-ID Lisboa, Portugal
Salvatore Venticinque, Second University of Naples - Aversa, Italy
Mario Jose Villamizar Cano, Universidad de loa Andes - Bogotá, Colombia
Salvatore Vitabile, University of Palermo, Italy
Bruno Volckaert, Ghent University - iMinds, Belgiums
Lizhe Wang, Center for Earth Observation & Digital Earth - Chinese Academy of Sciences, China
Zhi Wang, Florida State University, USA
Mandy Weißbach, University of Halle, Germany
Philipp Wieder, Gesellschaft fuer wissenschaftliche Datenverarbeitung mbH - Goettingen (GWDG), Germany
John Williams, Massachusetts Institute of Technology, USA
Peter Wong, SDL Fredhopper, Netherlands
Christos Xenakis, University of Piraeus, Greece
Hiroshi Yamada, Tokyo University of Agriculture and Technology, 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
Ustun Yildiz, University of California, USA
Qi Yu, Rochester Institute of Technology, USA
Jong P. Yoon, Mercy College - Dobbs Ferry, USA
Jie Yu, National University of Defense Technology (NUDT), China
Ze Yu, University of Florida, USA
Massimo Villari, University of Messina, Italy
Vadim Zaliva, Tristero Consulting / Carnegie Mellon University (CMU), USA
José Luis Zechinelli Martini, Fundación Universidad de las Américas, Puebla (UDLAP), Mexico
Baokang Zhao, National University of Defence Technology, China
Xinghui Zhao, Washington State University Vancouver, Canada
Zibin Zheng, Sun Yat-sen University, China
Jingyu Zhou, Shanghai Jiao Tong University, China
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

Impact of Cloud Computing on Enhancing the Use of Renewable Energy

Kwa-Sur Tam
Department of Electrical & Computer Engineering
Virginia Tech
Blacksburg, Virginia, U.S.A.
Email: ktam@vt.edu

Abstract — **Renewable energy has been identified as one of the disruptive technologies that have the potential for massive impact on the society for the coming years. A major concern for using renewable energy such as solar photovoltaics and wind is its variable nature. The importance of this concern is increasing in recent years as the penetration levels of renewable energy have been increasing rapidly in the electric power grids worldwide. Forecasting is a major tool that can be used to address the variable nature of renewable energy. Cloud Computing provides the enabling technology to handle the complexity of renewable energy forecasting and can enhance more widespread and more effective utilization of renewable energy. This paper presents the new applications of using Cloud Computing to enhance the use of renewable energy through the achievement of two goals. The first goal is to present a Cloud Computing-enabled renewable energy forecasting system--the FaaS (Forecast-as-a-Service) framework--to demonstrate the technical feasibility. Based on the service-oriented architecture (SOA), the FaaS has been successful in generating user-specified solar or wind forecast on demand and at reasonable costs. Using the FaaS as the starting point, the second goal of this paper is to present from a broader perspective the potential impact of Cloud Computing on the use of renewable energy. Cloud Computing, coupled with other technology trends, supports the development of new applications and business models that could flourish the renewable energy industry.**

Keywords – Cloud Computing; services; service-oriented architecture; forecasting; renewable energy; cyberinfrastructure.

I. INTRODUCTION

Providing almost unlimited computing resources on a pay-per-use basis, Cloud Computing provides new options for data-intensive and computation-intensive applications. Cloud computing not only makes possible the completion of complex computational tasks within shorter time frames but also enables such capabilities to be available at affordable costs. This paper describes the impact of Cloud Computing on the use of renewable energy.

Renewable energy has been identified as one of the disruptive technologies that have the potential for massive impact on the society for the coming years [1]. A major concern for using renewable energy such as solar photovoltaics and wind is its variable nature. The importance of this concern is increasing in recent years as the penetration levels of renewable energy have been increasing rapidly in the electric power grids worldwide. Forecasting is a major tool that can be used to address the variable nature of renewable energy. Based on the forecast information, variability can be accommodated on the power supply side by implementing measures such as generation scheduling and storage backup, and on the power consumption side by implementing demand-side management and demand response programs. Accurate forecasting of renewable energy will provide important contribution to the realization of smart grid and enable more widespread and efficient utilization of renewable energy. Cloud Computing provides the enabling technology to handle the complexity of renewable energy forecasting.

This paper presents the new applications of using Cloud Computing to enhance the use of renewable energy through the achievement of two goals. The first goal is to present a Cloud Computing-enabled renewable energy forecasting system--the FaaS (Forecast-as-a-Service) framework--to demonstrate the technical feasibility. Based on the service-oriented architecture (SOA), the FaaS has been successful in generating user-specified solar or wind forecast on demand and at reasonable costs. The FaaS framework can be used as a software-as-a-service (SaaS) to provide prospecting or operational forecast for solar or wind power systems. It can also be used as a platform-as-a-service (PaaS) to develop more capabilities or more customized functionalities.

Using the FaaS as the starting point, the second goal of this paper is to present from a broader perspective the potential impact of Cloud Computing on the use of renewable energy. Cloud Computing, coupled with other technology trends, supports the development of new applications and business models that could flourish the renewable energy industry.

This paper offers contributions that are different from those reported in existing literature. Although both the FaaS and the CloudCast [2] are concerned with forecasting, FaaS is different from CloudCast not only in terms of the services provided but also in terms of the underlying design. Efforts to bring together service-oriented architecture and Cloud Computing have been reported in the literature [3][4]. FaaS is different from these efforts in that FaaS also addresses service pricing issues. There are patterns for object-oriented software design [5] and patterns for SOA service design [6]. The FaaS Framework may be viewed as the preliminary version of a Cloud Computing pattern for on-demand

 ISBN: 978-1-61208-460-2

quantitative forecasting processes. By improving the flexibility and economics of renewable energy forecasting services, FaaS also achieves the goal of Services Computing [7].

Section II presents the concepts and implementation of the FaaS framework and the forecasting results generated. Broader impact of Cloud Computing on the use of renewable energy is discussed in Section III. Conclusions are contained in Section IV.

II. CLOUD-ENABLED FORECASTING SYSTEM

As shown on the top of Figure 1, a quantitative forecasting process may be grouped into four major steps: problem definition; data collection; analysis and model formulation; and forecast generation. . Using the principles of service-oriented architecture (SOA), each of these major steps can be performed by a composite service. Figure 1 shows the layered organization of a SOA framework used in the FaaS framework for the forecasting of renewable energy. Services in the Service Layer consist of the fundamental and agnostic services that are not coupled to any specific application. They perform tasks such as data transfer over the Internet (Transfer Tools services), statistical analysis (Statistical Tools services), forecasting (Forecast Tools services), etc. Many of these basic services are used in both the wind and the solar power forecasting processes.

On top of the Service Layer are the Composite Service and Workflow Layer. The External Data Collection Framework (EDCF) is responsible for gathering relevant data from different sources over the Internet. These data are available from a variety of sources: federal agencies, national databases and archives, private organizations, universities, data vendors and equipment vendors. From these sources, different types of data are rendered in different formats: satellite images, sensor measurement data, computer model data, vendor product data, etc.

The Internal Data Retrieval Framework (IDRF) processes and analyses the externally collected data and stores the results as internal data for future uses. The Forecast Generation Framework (FGF) generates the pertinent forecast of either wind or solar power at the location specified by the user. The FaaS controller serves to organize the entire forecast process and orchestrates different services to implement the workflow.

Each of the major steps shown on top of Figure 1 is performed by a composite service: the EDCF for data collection; the IDRF for the support of analysis and model formulation; the FGF for forecasting and the FaaS controller for problem definition and overall coordination.

The EDCF, IDRF, FGF and the FaaS controller are all composite services designed by applying SOA principles [6][8]. They are implemented by using the Windows Communication Foundation (WCF) [9], Microsoft Azure and .NET technologies [10].

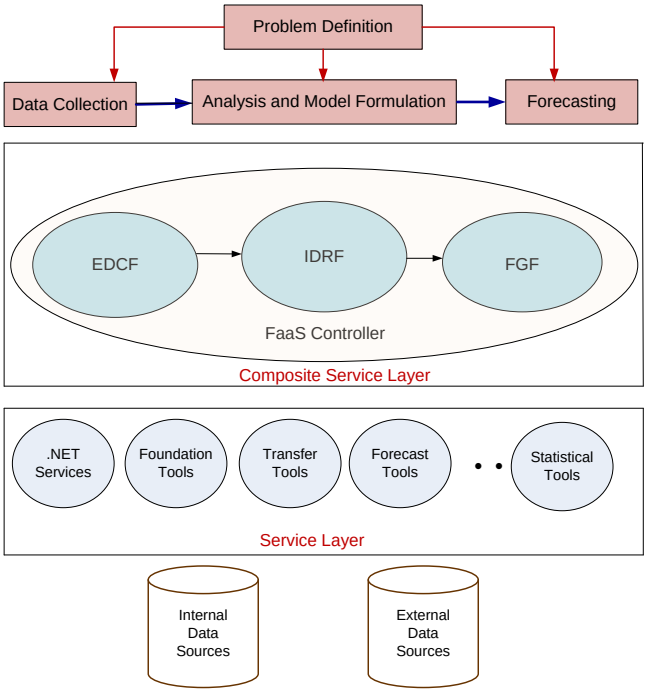


Figure 1. A SOA-based framework for the forecasting process

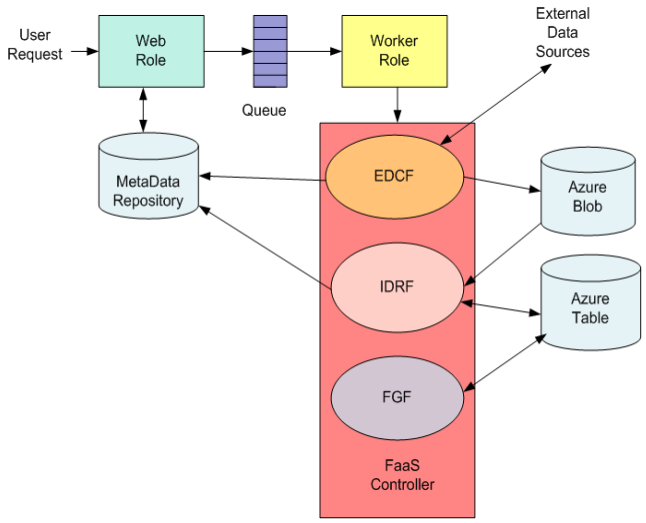


Figure 2. Implementation of the FaaS framework using Windows Azure

Figure 2 shows the implementation of the FaaS system using the Windows Azure Cloud Computing platform. To initiate a forecast process, a user through a web page specifies the renewable energy source (solar or wind), the location (latitude and longitude), the kind of forecasting

 ISBN: 978-1-61208-460-2

services (prospecting or operational), uncertainty quantification (with or without), and whether characteristics of energy conversion equipment (such as a particular model of a solar panel or a wind turbine from a list of manufacturers) should be included in the computation to make the forecast more realistic. Based on the customer-specified forecast request, the FaaS controller formulates a workflow consisting of all the tasks that need to be performed.

If relevant data are not available in the internal database (bottom layer in Figure 1), the FaaS controller informs the EDCF to obtain the needed data from external sources over the Internet and store the collected data in the Azure Blob storage. The FaaS system keeps a database of external data sources in a Meta Data Repository (MDR). MDR stores information about what kind of data a data source provides, the web address, the protocol for data access and the data format. Based on the metadata from the MDR, the EDCF initiates a data collection process to gather data for the location (or the nearest location) requested by the user from the pertinent data source. The EDCF utilizes the Transfer Tools services (shown in Figure 1) and other basic services to accomplish the external data collection process.

The IDRF takes the raw data from the Azure Blob storage, performs data analysis and stores the results in the Azure Table storage in the standardized formats for which the forecasting models are designed. As a composite service, the IDRF utilizes the appropriate services based on the metadata for the raw data. For example, the Statistical Tools service is used to generate statistics for long term historical data analysis. The Discrete Fourier Transform service is used to generate frequency domain components.

Using data prepared by the IDRF and stored in the Azure Table storage, the FGF generates the forecast based on the user's request. Different forecasting methods are implemented as services and used by the FGF. The forecast results are emailed to the user.

Because of the variety of information involved in the FaaS framework, it is important to have a well-organized Meta Data Repository (MDR) and an effective Meta Data Repository Management System (MDRMS) [11]. EDCF, IDRF, FGF and the FaaS controller all interact with the MDR through the MDRMS.

Figure 3 shows an example of the results provided by the FaaS system in response to a request for solar power forecast for operational planning purpose. The forecast is performed at the user-specified location in terms of latitude and longitude. If data is not available internally or externally for the requested location, data for the nearest location with data will be used to generate the forecast and the user will be notified of this distance. For example, Figure 3 shows that the distance between the requested and actual location for the forecast is 2.52 km. A number of basic services have been developed for tasks such as the computation of distances based on longitude and latitude coordinates, calculation of sky clearness index (K_D in Figure3), statistical analysis of data, etc. Figure 3 shows that the day ahead solar energy forecast is 6.036 kWh/m^2.

You requested operational forecast data.

Time of Report Delivery		3/24/2013 7:04:47 PM GMT
Time of Request		3/24/2013 7:04:44 PM GMT
Cost		16.68 dollars
Location: Name		Blacksburg, Virginia
Latitude 37.217		Latitude you entered 37.2
Longitude 80.417 W		Longitude you entered 80.4 W
State Code		VA
Source of Renewable Energy		Solar
Solar Panel:	Vendor	Sun Power
	Model	SPR-200-WHT-U
	Efficiency	16.08 %

Distance between requested and actual location: 2.52 km

Day-ahead Forecasted Energy Production: 6.036 kWh/m^2

K_D : 0.705

Figure 3. An example of forecast results in response to a request

Due to the complexity of SOA, the costs of SOA-based projects are difficult to estimate although there have been attempts to do so [12][13]. This project attempts to develop a method to price services so that a customer has the option to decide whether to proceed with the forecast request after viewing the estimated cost. The approach taken by the FaaS adopts the divide-and-conquer concept and product pricing concepts [14].

The price of each service is computed by using a 3-step process.

Step 1: Calculate the total cost by combining the cost of manpower for software development, the cost of resources utilized, etc., and imposing an overhead rate and indirect costs.

Step 2: Estimate the expected number of usage of this service over a time horizon before the next major update.

Step 3: Divide the total cost computed in step 1 by the expected number of usages in step 2 to obtain the service price per usage.

When services are combined to form composite services, the prices of constituent services are included in the cost of resources utilized. All the costs and prices are updated periodically after more usage information becomes available.

 ISBN: 978-1-61208-460-2

To implement this pricing method, each service is equipped with two endpoints – one endpoint is used for technical functionalities and the second endpoint is used for pricing purposes. When a service is consumed because of its technical functionalities, the pricing endpoint of the same service will also be incorporated into the overall pricing workflow. When a certain mission is accomplished by a sequence (or workflow) of services, not only the technical requirement is met but the associated price of accomplishing the mission is also calculated.

Using the economic endpoints of all services involved, the FaaS controller estimates the cost for the request and send the cost estimate to the requester. If the requester accepts the estimated cost, the requester will provide the email address to which the forecast results will be sent. The FaaS system will then perform the required tasks in the cloud and deliver the forecast results over the Internet after the work is complete.

Figure 3 shows that the estimated cost for that particular forecast request is US$ 16.68. Table I shows an example of the overall solar forecasting cost and the costs of the composite services for project prospecting and operational planning, respectively. The prices for prospecting forecasts are usually higher than those of the operational forecasts because they involve data over longer time horizons and utilize more computational resources. Results of this project indicate that the costs of prospecting forecasts are in the range of US$ 60-80 per request while the costs for operational forecasts are in the range of US$ 10-20 per request. Additional services, such as uncertainty quantification, can be requested for additional prices in the order of a few dollars. These costs are much lower than the monthly or annual subscription fees charged by current renewable energy forecast service vendors.

Figure 4 shows the application of the FaaS system to solar power forecasting. A similar diagram can be drawn to show the application of the FaaS system to wind power forecasting. As shown in the upper half of Figure 4, a large volume of data is needed for solar power forecasting. These data are available from a variety of sources that provide data of different types and in different formats. The EDCF is designed to obtain data of different types from various sources over the Internet and store them in the Azure Blob storage. The IDRF processes the raw data into a unified format useful for the different forecasting methods implemented as services. This two-step process is the approach adopted by the FaaS system to handle big data. FaaS demonstrates that automated collection and processing of large amount of data in various formats and from different sources is a unique capability provided by Cloud Computing.

As shown in the lower half of Figure 4, this framework delivers different types of forecasts to different types of users. There are potential users who want forecasts of future solar power production to support the making of investment decisions. Current users of solar power need to know the short-term forecasts to make arrangement for operation. Electric utilities need to know the solar power forecast with quantified uncertainty to properly plan for their operation. The FaaS system delivers a variety of user-specified forecast results over the Internet to meet forecast needs at reasonable costs. On-demand delivery of user-specified services at different levels of details for various kinds of applications is another unique capability provided by Cloud Computing.

TABLE I. AN EXAMPLE OF THE COSTS OF FORECASTING SERVICES

Forecast Type	Service	Cost (US $)	Overall Cost (US $)
Project Prospecting	EDCF	10.55	62.89
	IDRF	32.02	
	FGF	20.32	
	FGF (with UQ)	22.87	65.44
Operational Planning	EDCF	3.84	13.92
	IDRF	3.06	
	FGF	7.02	

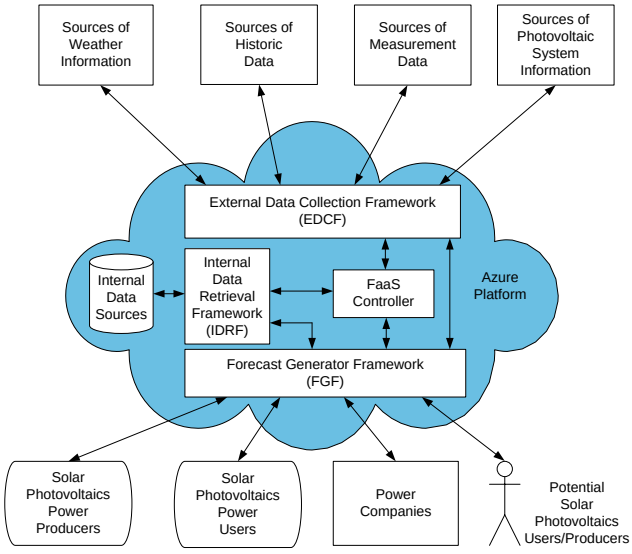


Figure 4. Application of FaaS to solar energy forecasting

III. IMPACT OF CLOUD COMPUTING

Renewable energy forecasting is data-intensive and computation-intensive. Providing almost unlimited computing resources on demand, Cloud Computing provides new options for the computation and delivery of different kinds of services, and opens up new opportunities for entrepreneurs to establish new business models. Creativity and innovativeness of entrepreneurship will add

new impetus to enable more widespread utilization of renewable energy and will hasten the fulfillment of its potential to meet the energy needs of human society.

Depending on what responsibilities are shifted to the cloud, current and potential users of renewable energy can choose a service model and a deployment model most appropriate for their respective situations. SaaS provides flexibility and cost effectiveness. An organization only needs to connect to the forecast application software through a Web browser and use it, without the hassle and expense of developing, implementing, or supporting it. FaaS is an example of such an application.

The FaaS framework enables different users to specify and pay for only the forecast services they need on demand. Cloud-based systems, such as the FaaS, are especially meaningful to individuals and small companies that would like to consider using renewable energy but lacks the resources to obtain forecast information that fits their particular needs. Cloud-based systems, such as the FaaS framework, can provide a broad impact by removing some barriers for more widespread use of renewable energy. Although some government agencies and large research labs that have substantial computing resources have the capability to perform similar computational tasks as those described in this paper, these organizations do not provide on-demand forecasting services at affordable prices to the general public, especially for customized services at customer-specified locations. A few commercial vendors provide forecasting services but they usually demand higher prices and long-term commitment. Since a Cloud-based system, such as the FaaS system, can provide customized forecast services on an on-demand pay-only-what-you-need basis, it plays an important role in making renewable energy forecasting widely accessible and affordable for current and potential renewable energy users.

The use of PaaS is appropriate when new capabilities need to be developed for a particular application. Generic services delivered with the PaaS can increase the speed of development and reduce costs. FaaS can function as a PaaS and provide these benefits by making its underlying services (developed as SOA services) available to developers to build new composite services and application-specific services as well as workflows. Users that want to use FaaS as PaaS have access to the service libraries to develop new capabilities by using/modifying/adding SOA services.

Forecasting renewable energy availability is even more important for isolated power systems because the forecasts enable the system operators to better prepare and manage the balance between the load demand and the power generation. Convenient and cost-effective access to accurate renewable energy forecasting can encourage the use of renewable energy especially in rural areas where it is expensive to build electric power transmission and distribution infrastructures. Efforts by the U.S. Department of Agriculture to develop wireless broadband access in small and medium-sized communities would enable the availability of Cloud-based forecast systems, such as the FaaS system, to many new renewable energy users.

Internet access is essential for the benefits of Cloud Computing to materialize. Mobile Internet technology, consisting largely of smartphones and tablets, has been undergoing fast growth in recent years not only in developed countries but even more remarkably in developing countries. Because of the prohibitive cost of building conventional wired infrastructure in developing countries, wireless Internet is expected to grow very rapidly in the coming years [15]. In parallel to this development, Cloud-based forecasting services delivered over mobile Internet are especially useful for the development of distributed renewable energy systems in developing countries where electric power grids have not been extensively developed, especially in rural areas.

Due to the advance in technologies such as miniature sensors and wireless networks, Internet of Things (IoT) will be widely adopted especially in health care, infrastructure and public-sector services in the coming years [16]. By using sensors to gather information which is then transmitted using wireless networks, IoT is bringing significant improvement to remote monitoring. With more information gathered by using IoT on a continual basis, the number of information sources shown in the upper portion of Figure 4 will increase significantly. Because of the amount of data generated, Cloud Computing technologies have been suggested to merge with IoT to form the Cloud of Things (CoT) [17]. CoT combines two of the twelve disruptive technologies that will transform life, business, and the global economy in the coming years [1]. Renewable energy forecasting and effective utilization of renewable energy can benefit greatly from the information collected by using IoT and processed/analyzed by using CoT. The FaaS system presented in this paper may be viewed as an early version of a more comprehensive CoT system along this trend.

Urbanization is an important factor to be included in the planning for sustainability. Currently more than half of the world population lives in the cities. Urban areas of the world are expected to absorb almost all the future population growth while at the same time drawing in some of the rural population. To handle the issues caused by growing urbanization, cities need to be transformed into "smart cities" that manage their resources (including renewable energy sources) efficiently. Internet of Things and Cloud Computing are enabling technologies that can increase the "smartness" by increasing the cities' awareness of its environment and situations. Along this direction, the ClouT project has been initiated as a collaborative project jointly funded by the 7th Framework Programme of the European Commission and by the National Institute of Information and Communications Technology of Japan. Cloud-based data collection and analytic systems, such as the FaaS system, will play an important role in smart cities in the future.

 ISBN: 978-1-61208-460-2

In the future, the FaaS framework may be expanded into an even more powerful Cyber-Infrastructure For Renewable Energy (CIFRE) as shown in Figure 5. CIFRE will serve as a focal point for obtaining and sharing data and information, upgrading models by using new and shared data, sharing different kinds of SOA services to build new applications, combining forecasts obtained from using different approaches and from different forecasters, collaboration and cooperation between different combinations of stakeholders, education and training, etc. Cloud Computing will be instrumental in the development and deployment of CIFRE.

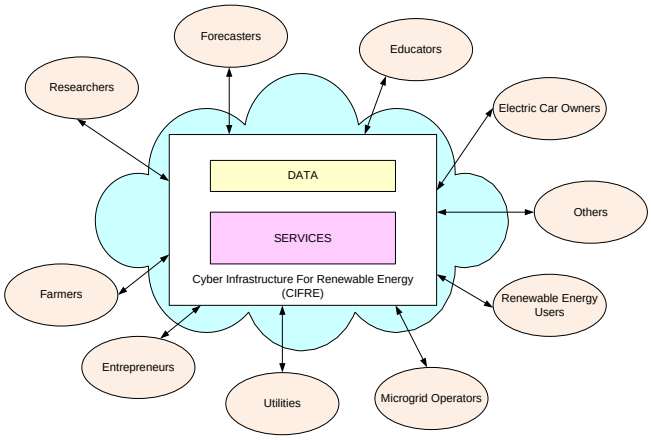


Figure 5. CIFRE (Cyber infrastructure for renewable energy) in the cloud

IV. CONCLUSIONS

This paper presents a Cloud Computing-enabled renewable energy forecasting system--the FaaS (Forecast-as-a-Service) framework. Based on the service-oriented architecture, the FaaS has been successful in generating user-defined solar or wind forecast on demand at reasonable costs. FaaS demonstrates that Cloud Computing offers two unique capabilities in the forecasting of renewable energy. The first one is automated collection and processing of large amount of data in various formats and from different sources. The second one is on-demand delivery of user-specified services at different levels of details for various kinds of applications.

A service pricing method that equips each service with a technical endpoint and an economic endpoint has been developed for the FaaS framework. When a mission is accomplished by a workflow, not only the technical requirement is met but the associated price is also calculated by using this method.

The broader impact of Cloud Computing on the use of renewable energy is presented. Coupled with mobile internet and Internet of things, Cloud Computing supports the development of new applications such as Cloud of things, smart cities, CIFRE and more widespread utilization of renewable energy in the rural areas.

ACKNOWLEDGMENT

This work is partially supported by the U.S. National Science Foundation under Grant 1048079. Contribution from Rakesh Sehgal is acknowledged.

REFERENCES

[1] J. Manyika, et al., Disruptive Technologies: Advances That Will Transform Life, Business, And The Global Economy, McKinsey Global Institute, 2013.

[2] D. Krishnappa, D. Irwin, E. Lyons and M. Zink, "CloudCast: Cloud Computing for short-term weather forecasts", Computing in Science & Engineering, pp. 30-37, 2013.

[3] Y. Wei, K. Sukumar, C. Vecchiola, D. Karunamoorthy and R. Buyya1, "Aneka cloud application platform and its integration with Windows Azure", Chapter 27, Cloud Computing: Methodology, Systems, and Applications, CRC Press, 2011.

[4] W. Tsai, X. Sun and J. Balasooriya, "Service-Oriented Cloud Computing Architecture", Seventh International Conference on Information Technology, pp. 684-689, 2010.

[5] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[6] T. Erl, SOA Design Patterns, Prentice Hall, 2009.

[7] J. Zhao, M. Tanniru, L. Zhang, "Services computing as the foundation of enterprise agility: Overview of recent advances and introduction to the special issue", Information System Front, Vol. 9, pp. 1–8, 2007.

[8] T. Erl, SOA Principles of Service Design, Prentice Hall, 2008.

[9] J. Lowy, Programming WCF Services, Third Edition, O'REILLY 2010.

[10] D. Chou, et al., SOA with .NET & Windows Azure, Prentice Hall, 2010.

[11] D. Marco and M. Jennings, Universal Meta Data Models, Wiley Publishing Inc., 2004.

[12] L. Yusuf, et al., "A Framework for Costing Service-Oriented Architecture (SOA) Projects Using Work Breakdown Structure (WBS) Approach", Global Journal of Computer Science and Technology, Vol. 11, Issue 15, pp. 35-47, 2011.

[13] Z. Li and J. Keung, "Software Cost Estimation Framework for Service-Oriented Architecture Systems using Divide-and-Conquer Approach", Proc. Fifth IEEE International Symposium on Service Oriented System Engineering, pp. 47-54, 2010.

[14] H. Snyder and E. Davenport, Costing and Pricing in the Digital Age, Library Association Publishing, 1997.

[15] Wireless Internet Institute, The Wireless Internet Opportunity for Devel oping Countries, World Times Inc, 2003.

[16] P. Parwekar, "From Internet of Things towards Cloud of Things", Proceedings, 2nd International Conference on Computer and Communication Technology (ICCCT-2011), pp. 329 – 333, 2011.

[17] M. Aazam, et al., "Cloud of Things: Integrating Internet of Things and Cloud Computing and the issues involved", Proceedings 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST), Islamabad, Pakistan, pp. 414-419, 2014.

 ISBN: 978-1-61208-460-2

Towards Using Homomorphic Encryption for Cryptographic Access Control in Outsourced Data Processing

Stefan Rass, Peter Schartner
Universität Klagenfurt, Department of Applied Informatics
email: {stefan.rass, peter.schartner}@aau.at

***Abstract*—We report on a computational model for data processing in privacy. As a core design goal here, we will focus on how the data owner can authorize another party to process data on his behalf. In that scenario, the algorithm or software for the processing can even be provided by a third party. The goal is here to protect the intellectual property rights of all three players (data owner, execution environment and software vendor), while retaining an efficient system that allows data processing in distrusted environments, such as clouds. We first sketch a simple method for private function evaluation. On this basis, we describe how code and data can be bound together, to implement an intrinsic access control, so that the user remains the exclusive owner of the data, and a software vendor can prevent any use of code unless it is licensed. Since there is no access control logic, we gain a particularly strong protection against code manipulations (such as "cracking" of software).**

Keywords*—*private function evaluation; cloud computing; licensing; security; cryptography.

I. INTRODUCTION

Cloud computing is an evolving technology, offering new services like external storage and scalable data processing power. Up to now, most cases of data processing, such as statistical computations on medical data, are subject to most stringent privacy requirements, making it impossible to have third parties process such person-related information.

A classical technique to prevent unauthorized parties from reading confidential information is by use of encryption. Unfortunately, this essentially also prevents any form of processing. This work concerns a generic extension [1] to standard ElGamal encryption, towards enabling permitted parties to process encrypted information without ever gaining access to the underlying data.

The core of this paper is a mechanism to endow the data and software owner with the capability of allowing or preventing designated parties from using either the data or the software for any data processing application. This is to let users retain full control over their data and software. The licensing scheme described herein is thus a method of providing or revoking the explicit consent to data processing in privacy. Moreover, unlike classical access control techniques, our scheme is cryptographic and as such cannot be circumvented nor deactivated by standard hacking techniques.

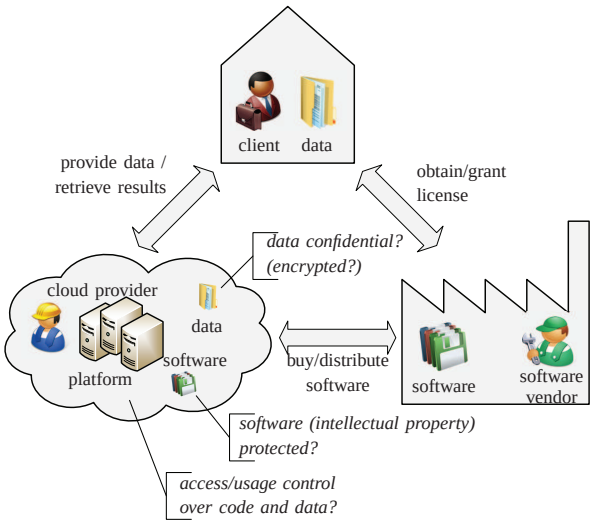


Figure 1. Example Scenario – Cloud Computing.

The most general scenario to which our licensing scheme (and computing model) applies involves three entities: first, there is the *client* (CL), who owns data that needs processing. The second player is the *software vendor* (SV), who owns the code for data processing. The third party is the *execution environment* (EE), which is the place where the actual data processing takes place (e.g., a cloud provider with sufficient hardware resources, or similar).

Figure 1 illustrates an example scenario, in which a client hands over its data to a cloud provider who runs third-party software for data processing services. Security issues are printed in italics.

Especially the client and software vendor have different interests, which may include (but are not limited to) the following:

- The client wants to keep its data confidential and wants to keep control over how and where it is processed
- The software vendor wants to prevent theft of its computer programs (software piracy), or other misuse of its software by unauthorized parties

The execution environment can be seen as the *attacker* in our setting: it is the only party that has access to both, the data

 ISBN: 978-1-61208-460-2

and the algorithms to process it. So, its main interest would be gaining access to the data, or run the program on data of its own supply. We emphasize that the described protection does not automatically extend to the algorithm itself. However, it is a simple yet unexplored possibility to apply code obfuscation in the computational model that we sketch in Section I-A.

Based on the above division, we can distinguish the following four scenarios:

1) All three entities separated: in this setting, the EE runs an externally provided software from the SV on data provided by the CL.
2) SV = EE: an example instantiation of this setting would be cloud SaaS, such as GoogleDocs. Here, the client obtains a licence to use a particular software, but seeks to protect his data from the eyes of the (cloud) provider.
3) CL = SV: here, the client is the one to provide the code for data analysis, yet seeks to outsource the (perhaps costly) computation to an external entity, e.g., a cloud provider.
4) CL = EE: the client obtains the software from the SV and runs the code on its own data within its own premises. Here, actually no particular licensing beyond standard measures is required (not even encrypted code execution), so we leave this scenario out of further investigations.

A. The Basic Idea – Outline of the Main Contributions

Briefly sketching what comes up, we will describe how algorithms can be executed on encrypted data, using a a *blind Turing machine* (BTM) [1]. Leaving the details of BTMs aside here (for space reasons), the central insight upon which this work is based is the fact that BTMs require a secret encoding of the data, which establishes compatibility between the data and the program that processes it. More specifically, BTMs, in the way used in this paper, allow the execution of arbitrary assembly instructions on encrypted data. Briefly (yet incompletely) summarizing the idea posed in [1], we encrypt a data item x into a pair $(E_{pk_1}(x), E_{pk_2}(g^x))$, where pk_1, pk_2 are two distinct public keys, g^x is a cryptographic commitment to x, and E is any public key encryption. The crux of this construction is the possibility of comparing two encrypted values $x_1 \stackrel{?}{=} x_2$, without revealing either value, based only on decryptions of the commitments $x_1 = x_2 \iff g^{x_1} = g^{x_2}$. Herein, neither commitment reveals x_1 or x_2, if computing discrete logarithm computations are intractable in the underlying group of E (the trick is similar yet with a different goal as for commitment consistent encryption; cf. [2]). Hereafter, we will use a subgroup of prime order q within the set $\mathbb{Z}_p$, when p is a large safe prime.

Executing arithmetic assembly instructions like `add A, B, C`, where $A \leftarrow B+C$ and B, C are encrypted values, computing the sum (or any other operation like multiplications, logical connectives, etc.) can be done by a humble table-lookup, based on the equality checking of encrypted inputs. Equally obvious is that the necessary lookup tables have to be small, i.e., we have only a small number of inputs $\{x_1, \ldots, x_n\}$. Practically, n is limited to small values of n, to keep the lookup tables (of size $O(n^2)$ feasibly small). Indeed, this is still an advantage of many fully homomorphic encryption schemes, which work on the bit-level (where we would have $n = 2$ for $x_1 = 0$ and $x_2 = 1$ in our setting).

The smallness of the plaintext space, together with the equality checking of the (so-modified) encryption scheme, also enables attacks by brute-force trial encryptions (of $x_1, \ldots, x_n$) and equality checks of the candidate plaintext to decipher any register content. Thwarting this attack is simple, if the encryption additionally uses a secret random representative a to encode the input before encrypting it (thus taking away the adversary's ability to brute-force try all possible plaintexts). That is, the encryption of x is actually one of $a \cdot x$. To ease notation in the following, we write $E_{pk}(g^{a \cdot x})$ as a shorthand of the secret message x being encoded with the random value a, where the encoding is

$$x \mapsto g^{ax}. \tag{1}$$

As a technical condition, we require $\gcd(a, p-1) = 1$.

Our description of the computational model is admittedly somewhat incomplete, as we do not discuss how memory access or control flow can be handled in the blind Turing machine model (when applied to assembly instruction executions). We leave this route for further exploration along follow up research, and confine ourselves to the observation that code (involving encrypted constants like offsets for memory access, etc.) and data can be made compatible or incompatible, based on whether the secret encoding used for the code (a value a) and the data (another value b) is equal or not.

The rest of the paper will be devoted to changing the secret value a – the *encoding* – or negotiating it between two or three parties (CL, SV, EE). The respective protocols form the announced *licensing* scheme, which are nothing else than the *authorization* to use the encryption's plaintext comparison facility. Practically, knowledge of a and the comparison keys (the secret decryption key sk_2 belonging to pk_2, to decrypt the commitments) enable (or in absence disable) the ability to run an arbitrary algorithm on encrypted data.

The authorization is thus bound to knowledge of an *evaluation key*, which is composed from the comparison token (secret key sk_2), plus the lookup tables (for all assembly instructions). The encoding a is *excluded* from the evaluation key, so that it can be given to the EE without enabling it to process data of its own interest.

Blind Turing machines provide a technical possibility to do the following upon a combination with the licensing scheme as described in Section III:

1) Encrypt a software in a way so that only licensed copies of it can be run on input data. This is *security for the software vendor*, in the sense of preventing software use without license, e.g., by the EE.
2) Encrypt data in a way to bind its use to a single licensed copy of a software (so that data processing by unauthorized parties is cryptographically prevented). This

is *security for the client*, in the sense of preventing misuse of either her/his software license or her/his data as given to the EE.

B. Example Applications

We briefly describe three possible applications, leaving more of this for extended versions of this work.

Cloud Services for Data Processing: consider an online service that offers data processing over a web-interface, using a software that runs remotely within the cloud. The client could safely input its data through the web-interface, knowing that the cloud is unable to execute the program (for which the client has obtained a license) on other data that what comes from the client. In addition, the client can be sure that the cloud provider does not learn any of the secret information that the customer submits for processing.

Such services are already existing, although not at the level of security that we propose here. One example is *Google Docs*.

En-route information processing: sensor networks and smart metering infrastructures use decentralized data processing facilities. In case of *smart metering*, data concentrators collect and preprocess data harvested from the subscribers, before sending properly compiled information to the head end for further processing (such as billing, etc.). Using the proposed licensing scheme, this processing could be done in entirely encrypted fashion, without "opening or breaking" the encrypted channel for the sake of intermediate processing.

A third example scenario is the **protection of intellectual property**, namely the firmware that runs inside a device. This example is expanded in full detail in Section V.

C. Organization of the Paper

Section II discusses related work. Section III is based on the model for private function evaluation as sketched in Section I-A, and describes the ideas underneath the main contribution as described in Section IV. That section also completes the description of how the authorization is implemented and granted. Security of our protocols is discussed in Section VI, and concluding remarks are made in Section VII.

II. Related Work – Computation in Privacy

Processing encrypted data is traditionally done using one of three approaches: homomorphic encryption, multiparty computation (MPC) and garbled circuits (GC). Picking up homomorphic encryption as the most recent achievement, many well-known encryption schemes are homomorphisms between the plain- and ciphertext spaces. Prominent examples are RSA and ElGamal encryption, which are both multiplicatively homomorphic. Likewise, Paillier encryption [3] enjoys an additive homomorphic property on $\mathbb{Z}_n$, where n is a composite integer (as for RSA), and the Goldwasser-Micali encryption [4], which is homomorphic w.r.t. the bitwise XOR-operation.

Surprisingly, until 2009 no encryption being homomorphic w.r.t. to more than one arithmetic operation was known. The work of Gentry [5] made a breakthrough by giving an encryption that is homomorphic for both, addition and multiplication. Ever since this first *fully homomorphic encryption* (FHE), many variations and improvements have appeared (e.g., [6]–[8] to name a few), among these being *somewhat homomorphic encryptions*, which permit several arithmetic operations, however, only a limited number of executions of each operation (e.g., arbitrarily many multiplications, but only a one addition over time).

Yao's concept of *garbled circuits* [9] provides a way to construct arithmetic circuits that hide their inner information flow by means of encryption. Interestingly, this works without exploiting any homomorphism, and is essentially doable with most standard off-the-shelf encryption primitives (cf. [10] and references therein).

Common to these two mainline approaches to the problem of data processing in confidentiality is the need to construct evaluation circuits (for both, fully homomorphic encryption and garbled circuits) that strongly depend on the data processing algorithm. In that sense, neither technique offers a fully automated mechanism to put an arbitrary algorithm to work on encrypted information, and compilers that take over this task are subject of intensive ongoing research [11]–[17]. Similar difficulties apply to multiparty computation approaches [18]–[20] or combinations of GC and MPC [10].

In the past, encryption circuits or interactive protocols have commonly been used as computational models, as opposed to Turing machines, which have only recently been considered as an execution vehicle [1], [21]. The latter of these references proposed the concept of a *blind Turing machine*, which is an entirely generic construction that uses standard ElGamal encryption (unlike [21], which works with attribute-based encryption). The idea relies on Turing machines as the most powerful known computational model (up to other models being equivalent to Turing machines), and the construction resembles the full functionality of a general Turing machine using encrypted content. This approach has briefly been outlined in Section I-A.

III. The Licensing Scheme

The main objective of the licensing protocols is to change the value a that encodes the secret data item $x \in \mathbb{Z}_p$ by (1). Hereafter, we let $\in_R$ denote a uniformly random draw from the given set. From the construction of blind Turing machines, i.e., the execution of instructions by table-lookups on the data being processed, it is evident that a program can only be executed if the code and data obey the same encoding (since the lookup table in the evaluation key must use the same encoding a as the data, for otherwise the lookup will fail). Establishing or changing the common encoding a is detailed in the next subsection.

A. Changing the Encoding

If a is known, then, it is easy to switch to another encoding based on b, via raising (1) to the power of $a^{-1}b$, where a^{-1} is

 ISBN: 978-1-61208-460-2

computed modulo $p-1$ (this inverse exists, as we assumed a relatively prime to $p-1$; see Section III). This gives

$$(g^{ax})^{a^{-1}b} \equiv g^{axa^{-1}b} \equiv g^{bx} \pmod{p}. \quad (2)$$

B. Negotiating an Encoding

If a given encoding of one entity (e.g., the SV) shall be changed to a chosen encoding of another entity (e.g., the CL), then the following interactive scheme can be used to switch from encoding a to encoding b, while revealing neither value to the other party. The protocol is as follows, where entity A secretly knows the encoding a, which shall be changed into the encoding b that entity B secretly chose. Common knowledge of both parties are all system parameters, in particular the generator g and prime p are known to both parties A and B.

1) $A \rightarrow B$: an encoded item g^{ax}.
2) $B \rightarrow A$: raise g^{ax} to b, and return the value $(g^{ax})^b \equiv g^{abx} \pmod{p}$.
3) A: strip a from the exponent via $(g^{axb})^{a^{-1}} \equiv g^{bx} \pmod{p}$. A can continue to work with the new encoding b, which is in turn unknown to A.

Notice that the knowledge of A is x, a, g^x, g^{ax} and g^{abx}, from which b cannot be extracted efficiently.

IV. Putting it to Work

With the encoding taking the form (1) and the encryption E being *multiplicatively homomorphic* (e.g., ElGamal), we can apply Diffie-Hellman like protocols to change the values in the exponent $g^{a \cdot x}$ even within an encryption. In the following, it is important to stress that any communication between the entities in the upcoming scenarios is encrypted, in order to prevent external eavesdroppers from trivial disclosure of secret information (an evident possibility in the protocols).

1) Licensing Scenario 1: Three Separated Parties: Suppose that a program P written by the SV resides within the EE under an encoding a (unknown to the EE). We assume that the program P is encrypted under the SV's public key pk_{SV} for reasons of intellectual property protection and to effectively prevent an execution without explicit permission by the SV.

To obtain a license (permission) and execute the program, the following steps are taken (Figure 2 illustrates the process in alignment to Figure 1).

1) The CL initiates the protocol by asking SV for a license.
2) The SV chooses a secret value $b \in \mathbb{Z}_p$ and sends the quantity $a^{-1}b \mod (p-1)$ to the EE, which it can use to "personalize" the program P by re-encoding it as

$$E_{pk_{SV}}(g^{ax})^{a^{-1}b} = E_{pk_{SV}}((g^{ax})^{a^{-1}b}) = E_{pk_{SV}}(g^{bx}), \quad (3)$$

   by virtue of the multiplicative homomorphy of $E_{pk_{SV}}$.
3) The CL prepares the evaluation key, i.e., the respective lookup-tables under *his own* public key pk_{CL}. This implies that all results obtained from the lookup table can only be decrypted by CL after the computation has finished. In particular, it assures that all internal intermediate results obtained over the execution of P will be encrypted under the CL's public key, so that they remain inaccessible for the EE or the SV.
4) Using b the CL can encode and submit its data to the EE for processing. The results are encrypted under pk_{CL} and hence only accessible to the CL afterwards.

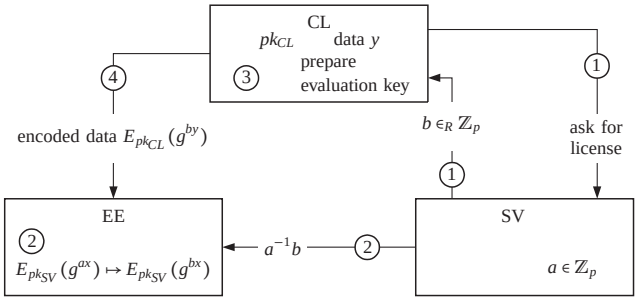


Figure 2. Licensing scenario involving three separated parties.

It is obvious that the scheme becomes insecure if two out of three of these entities collaborate in a hostile fashion. In either case, the secret encoding and also the secret data could be disclosed.

2) Licensing Scenario 2: SV = EE: Here, the EE/SV knows the encoding a but the client can interactively change it into his own chosen encoding b to obtain a license. Referring to Section III-B for the details, the remaining steps comprise the execution of the program P, which is then compatible with the secret encoding b under which the data has been prepared. For personalization, the SV/EE decrypts and submits all code items $E_{pk_{SV}}(g^{ax})$ to the client for re-encoding. Note that the CL *cannot* run the program, as it lacks the code itself (the CL gets only the constants found in the code). Figure 3 illustrates the details.

3) Licensing Scenario 3: CL = SV: This case is even more trivial, as the CL, being the SV at the same time, simply chooses the encoding a and submits its code and data to the EE for processing. No change or interactive negotiation of encoding is required here.

4) Involving a Different End-User: In some cases, the CL may be the source but not the final end-user of the data (e.g., in a smart meter network, where the CL is a user's smart meter, the EE is a data aggregator/data concentrator, and the end-user is the energy provider's head end system). In such cases, it is straightforward to prepare the evaluation key for a (fourth) party EU (*end user*). The change is simply by preparing the evaluation key under the EU's public key pk_{EU} instead of pk_{CL}. With this modification, all of the above scenarios work exactly as described.

V. Example Application Scenario

Another potentially very important application of our licensing scheme concerns the *protection of firmware (intellectual property)*. Suppose a manufacturer – here being the client CL – obtains the device's firmware from an external software vendor

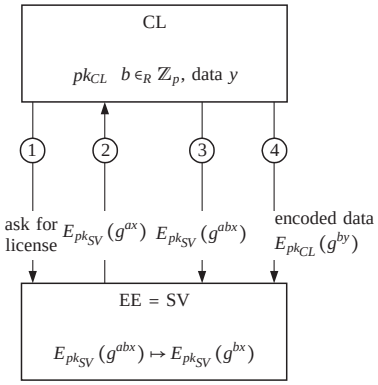


Figure 3. Licensing scenario when the data processing and software remains with the SV.

(SV). Furthermore, assume the device is equipped with an internal unique identity, such as a physically uncloneable function (PUF) or other hardwired unchangeable and uncloneable identity. We call this identity *ID*.

As before, let the firmware be code with encrypted fragments of the form $E_{pk_{SV}}(g^{ax})$, under a secret encoding $a \in \mathbb{Z}_p$ used by the SV, which is unknown to CL. After uploading the firmware, the device manufacturer obtains a license by

1) choosing a secret value β and submitting the blinded identity $\beta \cdot ID \bmod p$ to SV, and
2) retrieving the license $L' = (g^{b \cdot ID \cdot \beta}, a^{-1} \cdot b \cdot (ID^2) \cdot \beta^2 \bmod p)$ from the firmware manufacturer, where $b \in \mathbb{Z}_p$ is a secret random value chosen by the SV. The device manufacturer then strips the factors β, resp. β^2, from the contents of L' to obtain the final license $L = (\ell_1, \ell_2) = (g^{b \cdot ID}, a^{-1} \cdot b \cdot (ID^2) \bmod p)$.

We note that blinding β is only required to avoid attackers listening on the channel in an attempt to clone an identity and hence a device, runnable with the same license as for the honest manufacturer. Using β, the license L cannot be obtained from L' unless by the device manufacturer (or the device itself), knowing β.

Using the license L, the CL can personalize the code via

$$\left(E_{pk_{SV}}(g^{ax})\right)^{\ell_2} = \left(E_{pk_{SV}}(g^{ax})\right)^{a^{-1} b ID^2} = E_{pk_{SV}}(g^{b \cdot ID^2 \cdot x}), \quad (4)$$

and can encode input data y accordingly by computing

$$\ell_1^{y \cdot ID} \equiv (g^{b \cdot ID})^{y \cdot ID} \equiv g^{b \cdot ID^2 \cdot y} \pmod{p}. \quad (5)$$

We stress that an extracted firmware (e.g., software piracy) *will not* run on a structurally identical hardware, as the other device works with a different $ID' \neq ID$, even if the same license $L = (\ell_1, \ell_2)$ is brought into the device!

To see this, observe that the encoding is actually $b \cdot ID^2$, yet the encoding information for the data is only $g^{b \cdot ID}$, which enforces an exponentiation with the internally supplied identity (e.g., PUF-value) *ID*. Hence, the encoding by (5) will fail to reproduce ID^2 in the exponent, as *ID* cannot be replaced in the second device (as coming from a PUF for example). The exponent, in that case, will take the form $g^{b \cdot ID \cdot ID' \cdot y}$, which is incompatible with the program encoded via $g^{b \cdot ID^2}$.

VI. Security

By construction and the discussion in Section I-A, our scheme becomes insecure under any of the following two circumstances:

1) collaboration of at least two entities in any of the described licensing scenarios
2) a party succeeds in extracting the constant a that defines the secret encoding of symbols as defined in (1).

Obviously, we cannot mathematically rule out hostile cooperations among any of the entities in our context, but we can prove that the second of the above attack scenarios will fail under usual computational intractability hypotheses.

More concretely, we prove security of our licensing scheme by showing that the extraction of a license is at least as hard as computing discrete logarithms in the underlying group (see Definition VI.1). To this end, we distinguish different potential attackers and licensing scenarios according to our preceding discussion. Throughout this section, we assume passive adversaries and authenticated parties (thus, we do not discuss person-in-the-middle scenarios here).

Definition VI.1 (Discrete Logarithm Problem)**. Given:** *a prime p, a generator g of $\mathbb{Z}_p^*$ and a value $y \in \mathbb{Z}_p^*$.*
Sought: *an integer $x \in \mathbb{N}$, such that $y = g^x \bmod p$. We write $x = dlog_g(y)$ and call this the* base-g-logarithm of y.
We call this problem intractable, *if there is no efficient algorithm able to compute the base-g-logarithm of y.*

Under the intractability of discrete logarithms, security of our scheme is easy to prove in every scenario. Observe that the encryption wrapped around the values considered in the following can be neglected in cases where the attacker is an "insider", i.e., the client CL, the execution environment EE, or an external person-in-the-middle intruder.

A. Licensing Scenario 1: Three Separated Parties

Precluding collaborations between parties, the attacker can either be the client, the execution environment or an external eavesdropper. Consequently, we need to analyze security in each case separately.

This case is essentially trivial, as the client CL gets only his personal license b, but cannot access the encrypted quantity $a^{-1}b$ that is sent directly to the execution environment. As b is chosen stochastically independent of a, it does not provide any information about a.

Under a slight modification by sending g^b instead of b in this scenario, we can even allow an attacker to mount a person-in-the-middle attack, in the course of which he gets $a^{-1}b$ and g^b in plain text. The following result asserts security even under this modified stronger setting.

 ISBN: 978-1-61208-460-2

Proposition VI.2 (Security against external adversaries)**.** *Let $a \in \mathbb{Z}_p$ be the secret license used by the software vendor, and let $I = (a^{-1}b, g^b)$ be the attacker's information. Computing a from I is at least as hard as computing discrete logarithms.*

Proof. Let A be an algorithm that extracts a from $I = (a^{-1}b, g^b)$. We construct another algorithm A' that computes discrete logarithms and takes only negligibly more time for this than A needs. Given a value y, algorithm A' simply submits the pair (z, y) to A, where z is a uniformly random number. As b is uniquely determined by $y = g^x$, there is another unique number z' that satisfies $z = z' \cdot x$, where z' is stochastically independent of x. Hence, the pair (z, y) has the proper distribution to act as input to A, and A returns z' so that the sought discrete logarithm of y returned by A' is $x = z' \cdot z$. □

By symmetry, security against a malicious execution environment EE holds by the same line of arguments, and under both, the modified and original licensing scenario. The problem for a malicious EE is to compute the client's license b from its information $I = (a^{-1}b, g^{by})$, which is even harder as before, as there is another stochastically independent quantity y that blinds b in that case. We hence get the following result, whose proof is obvious from the preceding discussion:

Proposition VI.3 (Security against malicious EE)**.** *Let $b \in \mathbb{Z}_p$ be the secret license of the CL, and let $I = (a^{-1}b, g^{yb})$ be the attacker's information. Computing b from I is at least as hard as computing discrete logarithms.*

B. Licensing Scenario 2: SV = EE

Here, the problem is to extract b from $(g^{ax}, g^{abx}, g^{by})$. Given that the software vendor is identical to the execution environment, we can assume the attacker to know the values x and a, so that the actual problem is to compute b from $I = (g^b, g^{by})$.

Proposition VI.4 (Security in case of malicious SV=EE)**.** *Let p, q be primes so that $p = 2q + 1$ and let g generate a q-order subgroup of $\mathbb{Z}_p$. Computing the client's secret b from $I = (g^b, g^{by})$ is at least as hard as computing discrete logarithms in the subgroup $\langle g \rangle \subsetneq \mathbb{Z}_p$.*

Proof. The argument is again a reduction: let A be an algorithm that correctly returns b upon input $I = (g^b, g^{by})$. We construct an algorithm A' that computes discrete logarithms as follows: given a value y, we submit the input (y, z) to A, where z is a random value. As $y = g^x$ uniquely defines a value x, it also uniquely defines a value z' so that $z = g^{xz'}$. To see this, observe that the solvability of the equation $g^{by} = z$, by taking discrete logarithms on both sides, is equivalent to the solvability of congruence $by \equiv \mathrm{dlog}_g z \pmod q$, which is trivial. Hence, (y, z) has the proper input-distribution for A, which then correctly returns the discrete logarithm x of $y = g^x$. □

We stress that working in subgroups is not explicitly assumed in the previous security proofs, yet is a standard requirement in secure instantiations of ElGamal encryptions, such as over elliptic curves. Hence, the additional hypothesis of proposition VI.4 is mild and will always be satisfied in practical scenarios.

C. Licensing Scenario 3: CL = SV

Here, the problem is for the EE to extract information from the data submitted by the client. This is equivalent to either breaking the cipher or computing discrete logarithms, and hence covered by known security proofs concerning the underlying cryptographic concepts.

VII. Conclusion and Outlook

This work is compilation of concepts that enable secure and authorized processing of encrypted information. In essence, it is a deployment scheme for private function evaluation based on blind Turing machines, where the involved parties can secure their interests (prevention of software piracy and prevention of personal data misuse) by running interactive protocols.

Along experiments with implementations of the ideas sketched here, we identified various security issues and possible attacks, some of which were sketched in the previous sections. Future work is on implementing the described ideas and studies of security implications on a real prototype implementing a full computing platform. To this end, the concept of oblivious lookup tables [22] has been devised as a substitute that does lookups without comparing encrypted plaintexts.

Unfortunately, chosen instruction attacks are not entirely disabled in that case, since branching instructions and memory access can be turned into a plaintext comparison facility (e.g., by submitting conditional branches and observe the control flow, or by asking for two ciphertexts to address the same memory cell, using the fact that the physical addressing is most likely deterministic). A working prevention against such misuse calls for additional code obfuscation (particularly on the branch instructions) and secure memory access techniques (such as oblivious RAM or private information retrieval). Interestingly, this renders the proof of security of blind Turing machines against active adversaries (see [1]) practically void, as the assumptions of the proof are violated in side-channel scenarios.

As an overall conclusion, however, the following points can be made:

- Processing encrypted information *appears possible* using standard encryption, although this induces new vulnerabilities like side-channel information leakage. Whether ultimate security can be achieved in this generic construction (as incompletely sketched in Section I-A) is an interesting open issue; we hope that this article stipulates future research in this direction.
- Authorized processing of data can be achieved in various settings by agreeing on secret encodings and/or changing them interactively by exploiting the homomorphy of encryption, as described in Section III.

 ISBN: 978-1-61208-460-2

- The efficiency of our licensing scheme depends on how large the code is that we "personalize", since every instruction of a program and every data item has to be (re-)encoded. The main practical bottleneck will, however, be the underlying data processing system; in case of blind Turing machines, this amounts to roughly 1 multiplication and 1 exponentiation per instruction (processing up to 4 bits). See [23] for a detailed analysis and comparison to competing approaches.

It is important to note that any of the described schemes can be implemented in general groups; there is no need to strictly rely on modulo-arithmetic (this particular instantiation serves only illustrative purposes). Hence, for a practical implementation, we recommend elliptic curve groups (elliptic curve cryptography) or similar as a substitute for the structure $\mathbb{Z}_p$.

Most importantly, the scheme works mostly using off-the-shelf cryptographic primitives that have been known for decades, are well understood and enjoy good hardware support already.

REFERENCES

[1] S. Rass, "Blind turing-machines: Arbitrary private computations from group homomorphic encryption," International J. of Advanced Computer Science and Applications, vol. 4, no. 11, pp. 47–56, 2013.

[2] E. Cuvelier, O. Pereira, and T. Peters, "Election verifiability or ballot privacy: Do we need to choose?" Cryptology ePrint Archive, Report 2013/216, 2013, http://eprint.iacr.org/.

[3] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in Proc. of EUROCRYPT'99, ser. LNCS, vol. 1592. Springer, 1999, pp. 223–238.

[4] S. Goldwasser and S. Micali, "Probabilistic encryption," Special issue of J. of Computer and Systems Sciences, vol. 28, no. 2, pp. 270–299, April 1984.

[5] C. Gentry, "Fully homomorphic encryption using ideal lattices," in Proc. of STOC'09. New York, NY, USA: ACM, 2009, pp. 169–178. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536440

[6] C. A. Melchor, P. Gaborit, and J. Herranz, "Additively Homomorphic Encryption with *d*-Operand Multiplications," in CRYPTO, ser. LNCS, vol. 6223. Springer, 2010, pp. 138–154.

[7] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption," in Prof. of EUROCRYPT'10. Berlin, Heidelberg: Springer, 2010, pp. 62–91. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13190-5_4

[8] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in Proc. of TCC'05, LNCS 3378, 2005, pp. 325–341.

[9] A. C.-C. Yao, "How to Generate and Exchange Secrets (Extended Abstract)," in FOCS. IEEE, 1986, pp. 162–167.

[10] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in 20th USENIX Security Symp. USENIX Assoc., 2011.

[11] N. Tsoutsos and M. Maniatakos, "HEROIC: homomorphically encrypted one instruction computer," in Proc. of (IEEE DATE'14), March 2014, pp. 1–6.

[12] S. Carpov, P. Dubrulle, and R. Sirdey, "Armadillo: a compilation chain for privacy preserving applications," Cryptology ePrint Archive, Report 2014/988, 2014, http://eprint.iacr.org/.

[13] C. Aguilar-Melchor, S. Fau, C. Fontaine, G. Gogniat, and R. Sirdey, "Recent advances in homomorphic encryption: A possible future for signal processing in the encrypted domain," Signal Processing Magazine, IEEE, vol. 30, no. 2, pp. 108–117, March 2013.

[14] M. Brenner, J. Wiebelitz, G. von Voigt, and M. Smith, "Secret program execution in the cloud applying homomorphic encryption," in Proc. of IEEE DEST, May 2011, pp. 114–119.

[15] V. Kolesnikov and T. Schneider, "A practical universal circuit construction and secure evaluation of private functions," in Prof. of FC, G. Tsudik, Ed. Springer, 2008, pp. 83–97. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85230-8_7

[16] W. Melicher, S. Zahur, and D. Evans, "An intermediate language for garbled circuits," in Poster at IEEE Symp. on Security and Privacy, 2012.

[17] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "Reusable garbled circuits and succinct functional encryption," in Proc. of STOC'13. New York, NY, USA: ACM, 2013, pp. 555–564. [Online]. Available: http://doi.acm.org/10.1145/2488608.2488678

[18] Z. Beerliová-Trubíniová and M. Hirt, "Perfectly-secure MPC with linear communication complexity," in Proc. of TCC'08. Berlin, Heidelberg: Springer, 2008, pp. 213–230. [Online]. Available: http://dl.acm.org/citation.cfm?id=1802614.1802632

[19] W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "TASTY: Tool for Automating Secure Two-Party Computations," in CCS. ACM, 2010, pp. 451–462.

[20] Y. Lindell, B. Pinkas, and N. P. Smart, "Implementing two-party computation efficiently with security against malicious adversaries," in Proc. of SCN'08. Springer, 2008, pp. 2–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85855-3_2

[21] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "How to run turing machines on encrypted data," Cryptology ePrint Archive, Report 2013/229, 2013, http://eprint.iacr.org/.

[22] S. Rass, P. Schartner, and M. Wamser, "Oblivious lookup tables," 2015, accepted at the 15th Central European Conference on Cryptology (CECC), http://arxiv.org/abs/1505.00605.

[23] S. Rass, P. Schartner, and M. Brodbeck, "Private function evaluation by local two-party computation," EURASIP Journal on Information Security, vol. 2015, no. 1, 2015. [Online]. Available: http://dx.doi.org/10.1186/s13635-015-0025-9

 ISBN: 978-1-61208-460-2

Data Center Network Structure using Hybrid Optoelectronic Routers

Yuichi Ohsita, and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan
{y-ohsita, murata}@ist.osaka-u.ac.jp

***Abstract*—Large data centers hosting hundreds of thousands of servers have been built to handle huge amounts of data. In a large data center, servers cooperate with each other, and thus the data center network must accommodate a large amount of traffic between servers. However, the energy consumption of large data center networks is a major problem because several large-capacity switches or many conventional switches are required to handle the traffic. A low-energy hybrid optoelectronic router has been proposed to provide high bandwidth between data center servers. The hybrid optoelectronic router has an optical packet switching functionality and packet buffering in the electronic domain. The optical packet switching provides large bandwidth communication between the optical ports. In addition, the router can be directly connected to server racks by its electronic port. The packets from the server racks are stored in an electronic buffer in the router. Then, the packets are converted into optical packets and sent to another router. Finally, when the packets arrive at the router connected to the destination server rack, they are stored in the buffer, converted to electronic packets, and sent to the destination server rack. In this paper, we discuss a data center network structure that contains hybrid optoelectronic routers. We propose a method for constructing a data center network that uses hybrid optoelectronic routers efficiently. Furthermore, we discuss the effect of the network structure on the number of routers required to accommodate a large amount of traffic in a mega data center.**

Keywords*—*Data Center Network; Topology; Optical Packet Switch

I. INTRODUCTION

Large data centers with tens and even hundreds of thousands of servers have been built to handle the vast amount of data generated by various online applications. Data center servers communicate with each other to handle the data. A lack of bandwidth or large delay prevents communication between servers and increases the time taken to retrieve data. This degrades the performance of the data center.

The energy consumption of data centers, which increases with the data center size, is another major problem, and the energy consumption of the network is a substantial proportion of total energy usage [1]. Therefore, data center networks with high energy efficiency and high communication performance are required [2].

Optical networking is a promising approach to constructing networks with high energy efficiency [3]. Optical network devices provide low latency communication with low energy consumption because they relay optical signals without conversion to electrical signals. Optical networking also provides high bandwidth via technologies such as wavelength division multiplexing (WDM).

An optical packet switch architecture called the *hybrid optoelectronic router* was proposed for data centers by Ibrahim et al. [4]. The hybrid optoelectronic router has optical packet switching functionality and packet buffering in the electronic domain. The optical packet switching functionality provides large bandwidth communication between the optical ports by relaying the optical packets without conversion to electronic signals unless packet collision occurs. Even if a collision occurs, the router can retransfer the packets after storing them in the electronic buffer.

In addition, the hybrid optoelectronic router can be connected directly to server racks via its electronic port. The packets from the server racks are stored in an electronic buffer in the router, and then converted into optical packets and sent to another router. Finally, if the packets arrive at the router connected to the destination server rack, the packets are stored in the buffer, converted to electronic packets, and sent to the destination server rack.

The network structure is important for constructing a large data center that uses hybrid optoelectronic routers and should use the large bandwidth of the hybrid optoelectronic routers efficiently. However, if each server rack can be connected to multiple routers, the connection from the server racks may have a large effect on the network performance. We have proposed a network structure that uses optical packet switches and multiple connections from the server racks [5]. However, our previous work used multiple connections from the server racks only to provide connectivity when optical packet switches fail, and did not discuss the effect of using multiple connections from the server racks on the performance of the data center network.

In this paper, we propose a method to construct a data center network structure that accommodates a large amount of traffic by using the hybrid optoelectronic routers and multiple connections from the server racks efficiently. We evaluate our network structure and demonstrate that our method can accommodate more traffic than a torus network. In addition, we discuss the importance of using multiple links from the server racks, and show that multiple links are necessary to construct a large data center with sufficient bandwidth between server rack pairs.

The rest of this paper is organized as follows. Section II explains related work about data center networks using optical network technologies. Section III provides an overview of hybrid optoelectronic routers and the data center network that uses hybrid optoelectronic routers. Section IV proposes a method to construct data center network structures using

 ISBN: 978-1-61208-460-2

hybrid optoelectronic routers. Section V discusses building a data center network that can accommodate more servers without decreasing bandwidth based on our method of constructing data center network structures. Finally, we conclude this paper in Section VI.

II. Related work

Farrington et al. proposed a data center network architecture that uses optical path switches [6]. In this network, the optical path switches are placed at the core of the data center network and are configured to connect the server rack pairs that are generating a large amount of traffic. Similar architecture using optical circuit switches was proposed by Wang [7]. However, the configuration of the optical path switches takes time, and this architecture cannot handle frequent traffic changes that often occur in a data center [8].

Another approach is to use optical packet switches [4], [9]–[13]. Optical packet switches contain arrayed waveguide grating routers (AWGRs) and wavelength converters. Because the output port of the input signal depends on the wavelength of the input signal in AWGRs, the destination port is changed by changing the wavelength. Optical packet switches relay optical packets based on the optical labels attached to the packets. Because optical packet switches do not require the establishment of paths, a network constructed of optical packet switches can handle frequent changes in traffic.

Optical packet switches with a large number of ports have also been constructed by connecting optical packet switches with a small number of ports. Xi et al. constructed an optical packet switch with a large number of ports by connecting the optical switches in a Clos topology [12]. Liboiron-Ladouceur et al. built a large data center by connecting the optical switches in a Tree topology [14]. However, they considered only the connections between optical switches. In a data center, servers may have electronic ports instead of optical ports. Therefore, we need to consider the connection from the servers or server racks that have electronic ports.

An optical packet switch architecture proposed by Ibrahim et al. [4], called a hybrid optoelectronic router, has electronic ports that can be connected directly to the servers. By using the connections between optical switches, and connections between optical switches and server racks, the data center network can accommodate a large amount of traffic between a large number of servers. However, a data center network that efficiently uses the both of these connections has not been reported. Therefore, we discuss a data center network structure that accommodates a large amount of traffic by efficiently using hybrid optoelectronic routers.

III. Data center networks with hybrid optoelectronic routers

In this section, we introduce the hybrid optoelectronic router, and the data center network constructed of them.

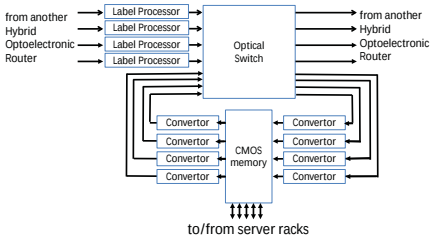


Figure 1. Hybrid optoelectronic router.

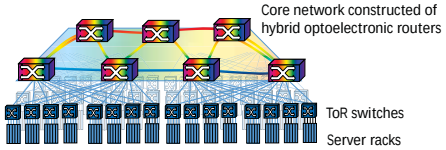


Figure 2. Network with hybrid optoelectronic routers in a data center.

A. Hybrid optoelectronic routers

Figure 1 shows a hybrid optoelectronic router. Each hybrid optoelectronic router has ports connected to other hybrid optoelectronic routers (hereafter called *optical ports*), and ports connected to the server rack (hereafter called *electronic ports*).

When an optical packet arrives through the optical port, the label processors identify its label and destination, and the controller controls the switching fabric to relay the packet to the destination. If the destination port is busy, the packet is stored in the electronic buffer after the optical packet is converted to an electronic packet. The packet is re-sent after it is converted into an optical packet.

Electronic packets from a server rack arrive through the electronic port. The electronic packet is buffered and converted into an optical packet. The converted packet is relayed in the same way as optical packets from the other optical packet switches. Optical packets sent to a server rack are also buffered and converted to electronic packets that are relayed to the server rack.

B. Data center network using hybrid optoelectronic routers

Optical networks containing hybrid optoelectronic routers provide large bandwidth communication. Therefore, we place an optical network containing hybrid optoelectronic routers at the core of the data center network. Each server rack has an ToR switch and is connected to the core network by connecting its electronic ToR switch to multiple hybrid optoelectronic routers. Figure 2 shows an example of a data center network containing hybrid optoelectronic routers.

In this network structures, the connections between the hybrid optoelectronic routers, and between the server racks and the hybrid optoelectronic routers must be set, which is discussed in Section IV.

 ISBN: 978-1-61208-460-2

C. Routing in the data center network by using hybrid optoelectronic routers

In the data center, the traffic changes frequently. If the routes are controlled by a central controller, the central controller has to collect the traffic information about the whole network frequently, which requires a large overhead.

Therefore, we use routing that balances the load among the shortest paths by using only the local information. Each hybrid optoelectronic router has a routing table, which includes multiple next hybrid optoelectronic routers on the shortest paths to each destination hybrid optoelectronic router. When a hybrid optoelectronic router receives a packet, the hybrid optoelectronic router selects the next router with the smallest load from the routing table.

When a ToR switch sends a packet to the core hybrid optoelectronic router network, the ToR switch encapsulates the packet by attaching the destination hybrid optoelectronic router. The packet is sent to one of the hybrid optoelectronic routers connected to the ToR switch. The first and destination router pair is selected from candidate pairs with the smallest number of hops for the first router connected to the source server rack and the destination hybrid router connected to the destination server. If there are multiple candidate pairs, the pair is selected randomly to balance the loads.

IV. Construction of the network structure

In this section, we propose a method for constructing the data center network structure by using hybrid optoelectronic routers. The server racks are grouped, and the racks in each group are connected to the same hybrid optoelectronic routers. Each server rack is connected to R hybrid optoelectronic routers, and each hybrid optoelectronic router is connected to S groups of server racks. We construct a network containing S^R groups of server racks and RS^{R-1} hybrid optoelectronic routers.

First, we connect the server racks to hybrid optoelectronic routers, and then set the connections between the routers.

A. Connection from server racks

We aim to maximize the amount of traffic that can be accommodated. A large number of hops between server racks decreases the amount of traffic that can be handled because hops consume bandwidths for a large number of links.

We connect server racks to hybrid optoelectronic routers to maximize the number of server racks that can communicate through only one hybrid optoelectronic router. We use a network architecture similar to the BCube [15].

We separate hybrid optoelectronic routers into R layers, and each server rack is connected to one hybrid optoelectronic router in each layer. To determine the connections between server racks and hybrid optoelectronic routers, we set the ID to the server rack group. Similarly, we set the ID to the hybrid optoelectronic routers in each layer.

We connect the ith hybrid optoelectronic router at the rth layer to the jth server rack when $\lfloor \frac{i}{S^{r-1}} \rfloor = \lfloor \frac{j}{S^r} \rfloor$ and $i \bmod S^{r-1} = j \bmod S^{r-1}$ are satisfied. By doing so, the server rack groups sharing hybrid optoelectronic routers are different for the different layers.

B. Connections between hybrid optoelectronic routers

After connecting the hybrid optoelectronic routers, we construct the connections between hybrid optoelectronic routers. To search for the best connections, we generate and select candidate connections.

1) Candidate connections: The number of available connections between hybrid optoelectronic routers is ${}_{(RS^{R-1})^2}C_{PRS^{R-1}}$, where P is the number of optical ports of each hybrid optoelectronic router. The number of available connections is too large, and it takes a long time to select the best one from all available connections. Therefore, we focus on candidates for which the number of hops between server racks is small, because networks where the number of hops is large cannot accommodate a large number of hops because the hops waste the bandwidth of many links. In addition, we focus on candidates where all hybrid optoelectronic routers play the same role, because if there are hybrid optoelectronic routers that play a special role, such as the root node of the tree topology, the loads on the special routers becomes large.

We construct the candidates through the following steps.

1) Set the network structure, where all server racks are connected to the hybrid optoelectronic routers but no links between hybrid optoelectronic router are constructed, as a candidate.
2) Add one link per hybrid optoelectronic router for each candidate if the candidate has an empty optical port. If no candidate has an empty optical port, end the process.
3) Select N candidates based on the number of hops between server racks.
4) Go to step 2.

In Step 2, we add links to each hybrid optoelectronic router so that all hybrid optoelectronic routers play the same role. After adding one link from the first hybrid optoelectronic router, we add links that have the same properties as the first link. In this paper, we regard the links satisfying the following constraints as links with the same properties.

- The source router for the link is included in the same layer.
- The destination router for the link is included in the same layer.
- The number of hops from the source router to the destination router on the network before adding the links is the same.

In Step 3, we select N candidates based on the number of hops between server racks. In this paper, we select the candidates with the largest number of hops between the server racks. If multiple candidates have the same largest number of hops. we compare the average number of hops between all server rack pairs.

Algorithm 1 shows the pseudo code for generating N candidates. In this pseudo code, R is the set of the hybrid optoelectronic routers, c^{init} is the initial candidate network

 ISBN: 978-1-61208-460-2

topology where all server racks are connected to the hybrid optoelectronic routers, but no links between hybrid optoelectronic routers are constructed. In each iteration from Line 3 to 35, we update the list of the candidates C by adding one optical link per hybrid optoelectronic router. To add one link per hybrid optoelectronic router, we first decide the router pair where the new link is added at Line 10. Then, for each router, we find target routers that have the same properties as the link added at Line 10, and add links between found route pairs from Line 12 to 23. At the end of each iteration, we save only N candidates from Line 33 to 34.

In these steps, we continue the iteration from Line 3 to 35 P times. In each iteration, we generate and evaluate at most $N(RS^{R-1})^2$ where RS^{R-1} is the number of hybrid optoelectronic routers. That is, to generate the candidates, we evaluate at most $PN(RS^{R-1})^2$ candidates. The number of generated and evaluated candidates becomes large as the number of hybrid optoelectronic routers becomes large. However, the number of hybrid optoelectronic routers is much smaller than the number of server racks. In addition, the candidates can be evaluated in parallel. Moreover, the candidate topologies are generated only once before constructing a data center. Therefore, we believe that the calculation time for generating the candidate topologies should not be a major problem.

2) Selection of the best candidate: Finally, we select the best candidate from the generated candidates. We simulate the routing in the data center when traffic is generated between all server rack pairs. Next, we select the candidate with the smallest link utilization to construct the network structure that can accommodate the largest amount of traffic. When constructing the data center network, the traffic within a data center is unknown. Thus. we set the traffic between all server rack pairs so that traffic between all server rack pairs equal.

C. Incremental construction

Constructing a data center network with a large number of server racks simultaneously is difficult. The data center should be incrementally constructed by adding server racks and routers. We propose a method to construct the data center network structure incrementally. We first calculate a suitable network structure including the maximum number of server racks and hybrid optoelectronic routers. Hereafter, we call this calculated network structure the largest network structure. Next, we construct the subset of the largest network structure that can connect the currently required number of server racks. When it is necessary to add more servers to the current network, we find the best network structure that includes the current routers and server racks and is a subset of the largest network structure. In the rest of this subsection, we explain the steps to finding the best network structure in detail.

1) Construction of the network structure when the number of the required server rack groups is given: We construct a network structure that can connect the number of the required server rack groups. The suitable network structure is calculated by generating candidates for the network structure that is a subset of the largest network structure, and selecting one of them. The candidates are generated by the following steps.

1) Construct the initial network. If there is a current working network, the current network is set as the initial network. Otherwise, select the server rack group with the smallest ID, and construct the initial network including the selected server rack group and all hybrid optoelectronic routers to which the groups are connected.
2) Add the constructed initial network to the *list of incom-*

Algorithm 1 Generation of N candidates for connections between hybrid optoelectronic routers.

```
1:  Clear the list of candidates, C.
2:  Add c^init to C.
3:  while True do
4:    Clear the list of newly generated candidates, C'
5:    for c ∈ C do
6:      Select the router, r_1 ∈ R, with the largest number of
        remaining ports.
7:      for r_2 ∈ R do
8:        if r_2 has remaining ports in c then
9:          Clear the list of temporal candidate C^tmp
10:         Construct a new candidate, c^new, by adding link
            between r_1 and r_2 to c.
11:         All c^new to C^tmp
12:         for r_3 ∈ R do
13:           Clear the list of temporal candidate C'^tmp
14:           for c^tmp ∈ C^tmp do
15:             for r_4 ∈ R do
16:               if r_4 has the remaining ports in the
                  network, c^tmp then
17:                 if link r_3, r_4 has the same property as
                    link r_1, r_2 then
18:                   Construct a new candidate, c'^new,
                      by adding a link between r_3 and r_4
                      to c^tmp.
19:                   Add c'^new to C'^tmp
20:                 end if
21:               end if
22:             end for
23:           end for
24:           C^tmp ← C'^tmp
25:         end for
26:         Add all candidates c^tmp ∈ C^tmp to C'
27:       end if
28:     end for
29:   end for
30:   if C' is empty then
31:     return C
32:   end if
33:   Constructing the list of candidates C'' by selecting N
      candidates from C'
34:   C ← C''
35: end while
```

plete candidates.

3) For each network in the list of incomplete candidates, generate the new networks by adding one hybrid optoelectronic router that is not included in the network, but has a link to one router included in the network in the largest network structure, and by adding links between hybrid optoelectronic routers included in the new network. Then, save the new network in the *list of newly constructed candidates*.
4) For each network in the list of newly constructed candidates, count the number of server rack groups connected to the hybrid optoelectronic router. If the number of server racks is more than the number of required server rack groups, add the network to the *list of candidates*.
5) If the list of candidates includes more than one candidate, end the process. Otherwise, replace the list of incomplete candidates with the list of newly constructed candidates, and go back to step 3.

In Step 1. we select the server rack group with the smallest ID, because all server rack groups play the same role in the largest network structure.

In these steps, we look for a network structure that can connect the required number of server rack groups by adding one hybrid optoelectronic router during each iteration. We end the process if at least one candidate is found. As a result, we can find candidates that can connect the required number of server racks with the smallest number of hybrid optoelectronic routers. The pseudo code for these steps is shown in Algorithm 2.

Next, we select the candidate that can accommodate the largest amount traffic through simulating the routing, similar to Section IV-B2.

Finally, we add optical links between hybrid optoelectronic routers or electronic links between server rack groups and hybrid optoelectronic routers if there are unused ports. By adding links, we can accommodate more traffic. In this paper, we generate all patterns of added links and select the best pattern to accommodate the largest amount of traffic through the routing simulation.

2) Construction of the network structure when the number of the required server racks is given: The network structure should be constructed to accommodate any possible traffic rate. To guarantee this, we use valiant load balancing (VLB) [16]. In VLB, we select the intermediate nodes randomly regardless of the destination to avoid concentrating traffic on certain links, even when traffic volume of certain node pairs is large. In the data center network, the intermediate node is selected from the server racks. The packet is encapsulated by attaching a header whose destination is the selected intermediate server rack. When the intermediate server rack receives the packet, it relays the packet to the final destination after removing the header.

By applying VLB, the traffic rate between server rack T satisfies the following inequality.

$$T \leq \frac{2B}{N}$$

Algorithm 2 Construction of candidate subnetworks of the largest network structure.

```
Clear C^incomplete.
Add the initial candidate, c^init, to C^incomplete.
while True do
  Clear C'^incomplete
  for c ∈ C^incomplete do
    for r ∈ R do
      if r is not included in c but has a link to a router
      included in c then
        Generate new candidate c^new by adding the
        router, r, and the server racks connected to r
        to c.
        Add c^new to C'^incomplete
      end if
    end for
  end for
  Clear C^complete
  for c ∈ C'^incomplete do
    if c include more than the required number of server
    rack groups then
      Add c to C^complete
    end if
  end for
  if C^complete is not empty then
    return C^complete
  end if
  C^incomplete ← C'^incomplete
end while
```

Here, B is the bandwidth from a server rack and N is the number of server racks. That is, we construct a network structure that can accommodate the flow of data with $\frac{2B}{S}$ between all server rack pairs.

Based on this, when the number of required server racks, N, is given, we construct the data center network that can accommodate N server racks by the following steps.

1) Set the number of server rack groups S to the number of the server rack groups included in the current network. If the data center network is newly constructed, set S to 1.
2) Set the number of server racks in each server rack group to $\frac{N}{S}$.
3) Construct a network that can accommodate S server rack groups by the steps in IV-C1.
4) Check whether the constructed network can accommodate the traffic $\frac{2B}{N}$ between all server rack pairs. If yes, designate the current network as a suitable network. Otherwise, go back to step 2 after incrementing S.

D. Properties of the constructed data center network

1) Constructed network structure: First, we show the network structure constructed by our method. Figure 3 shows the network constructed from 10 hybrid optoelectronic routers with four optical ports and 25 server rack groups that have two

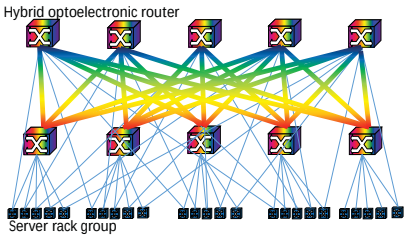


Figure 3. Example of the constructed network.

TABLE I
NUMBER OF HOPS IN THE CONSTRUCTED NETWORK STRUCTURE.

Number of hybrid optoelectronic routers on the path	Number of paths
1	192
2	288
3	96

electronic ports connected to hybrid optoelectronic routers. In this network structure, all hybrid optoelectronic router pairs can communicate without converting optical packets into electronic packets.

In this network structure, the number of hops between server racks is small (Table I). Most of the server rack pairs are connected to the same hybrid optoelectronic router or to hybrid optoelectronic routers that are directly connected to each other. Only 16 % of the server rack pairs require three hops to communicate with each other. In the network structure, the server rack pairs requiring three hops have three disjoint paths with the smallest number of hops. That is, this network structure can provide sufficient bandwidth by balancing the loads.

2) Delay: We investigate the delay between the server rack pairs through a simulation. We compare the delay in our network structure with that of the torus network constructed from the same number of hybrid optoelectronic routers and server racks [4]. We demonstrate the effectiveness of the data center network constructed considering the links from the server racks.

We model the delay for the hybrid optoelectronic router as follows. If packet collision does not occur, the hybrid optoelectronic router can relay the packet without buffering it. We assume that the delay in this case is 240 ns. If packets collide, the hybrid optoelectronic router converts the optical packet into an electronic packet, and stores it in the buffer. We assume that converting the optical packet into an electronic packet, and relaying it via the buffer takes 240 ns, similar to the case where the optical packets are relayed without collision. Storing the packet to the buffer and reading the packet from the buffer takes 180 ns. Converting the electronic packet into the optical packet and relaying it to the optical switch takes 240 ns. In addition, the packets stored in the buffer wait until the destination ports become available. We use the M/M/1 model to simulate the queuing delay in the buffer, where the average service time is 240 ns.

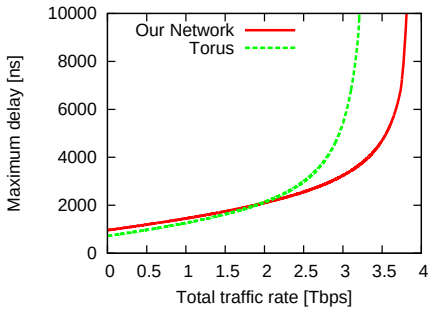


Figure 4. Comparison of the maximum delay among the server rack pairs.

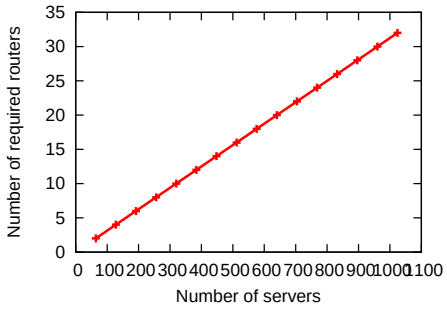


Figure 5. Number of required hybrid optoelectronic routers.

We construct both kinds of the network from 20 hybrid optoelectronic routers with four optical ports and 100 server racks with two electronic ports connected to hybrid optoelectronic routers. We generate the same traffic rate between all server rack group pairs.

Figure 4 shows the maximum delay among the server rack pairs. This figure indicates that our network structure can accommodate more traffic with a smaller delay than the torus network. This is because in our network structure, the connections between the server racks are decided considering the server rack groups connected to the hybrid optoelectronic routers.

3) Incremental construction: We construct the data center network incrementally. We set the total capacity of the electronic ports of the hybrid optoelectronic router to 32 Gbps, the bandwidth of the optical link to 100 Gbps, and the bandwidth of each server to 1 Gbps. That is, each hybrid optoelectronic router can connect 32 server racks. We set the largest network structure to the network structure constructed of 256 hybrid optoelectronic routers with four optical ports, and 32 server rack groups with two electronic ports. In the largest network structure, we connect 1024 servers without congestion.

Figure 5 shows that the number of required hybrid optoelectronic routers when we construct the network incrementally by adding 64 servers at each step. This figure indicates that we can always add 64 servers by adding only two hybrid optoelectronic routers.

 ISBN: 978-1-61208-460-2

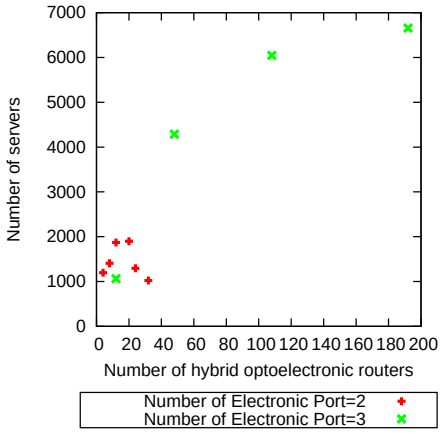


Figure 6. Number of hybrid optoelectronic routers vs number of accommodated servers (four optical ports).

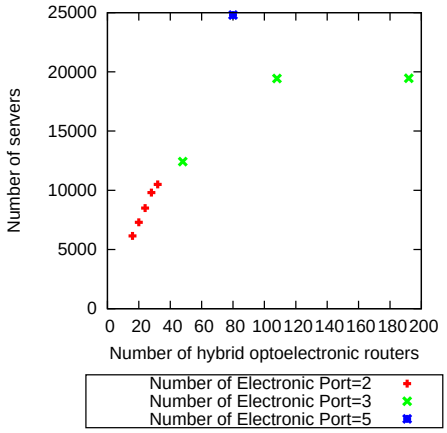


Figure 7. Number of hybrid optoelectronic routers vs number of accommodated servers (12 optical ports).

V. Discussion

In this section, we discuss the network structure that is suitable for a large data center based on our construction method. We investigate the relationship between the number of hybrid optoelectronic routers and the number of servers that can be accommodated without congestion for any traffic pattern by using VLB. In this investigation, we set the total capacity of the electronic ports of the hybrid optoelectronic router to a sufficiently large value, to focus on the bandwidth provided by the core network constructed of hybrid optoelectronic routers. We set the bandwidth of the optical link to 100 Gbps, and the bandwidth of the link from a server to 1 Gbps.

Figure 6 shows the relationship between the number of accommodated servers and the number of hybrid optoelectronic routers when each hybrid optoelectronic router has four optical ports. The figure shows network structures that include server racks with two or three electronic ports. As the number of hybrid optoelectronic routers increases, the number of optical links increases, which increases the capacity of the network and the number of accommodated servers. However, as the number of hybrid optoelectronic routers increases, the number of hops between hybrid optoelectronic routers increases. As a result, when the number of hybrid optoelectronic routers becomes sufficiently large, adding hybrid optoelectronic routers cannot greatly increase the number of servers that can be accommodated. In particular, when the number of electronic ports for each server rack is two, adding hybrid optoelectronic routers decreases the number of servers that can be accommodated. In contrast, the server racks with three electronic ports can accommodate more server racks. This is because the increase in the number of links from the server rack decreases the number of hops between server racks.

We also investigate the case where each hybrid optoelectronic router has 12 optical ports. Figure 7 shows the relationship between the number of accommodated servers and the number of hybrid optoelectronic routers when each hybrid optoelectronic router has 12 optical ports. Compared with Figure 6, the network constructed of hybrid optoelectronic routers with 12 optical ports can accommodate more servers because this network can provide more bandwidth owing to the larger number of optical links. In addition, the increase in the optical links decreases the number of hops between optical hybrid optoelectronic routers.

This figure also indicates that even if we use the hybrid optoelectronic router with 12 optical ports, we cannot accommodate more than 20,000 servers for server racks with two or three electronic ports. However, by connecting each server rack to four hybrid optoelectronic routers, we can connect 25,000 servers by using 80 hybrid optoelectronic routers. That is, to construct a large data center, multiple links from server racks are necessary.

VI. Conclusion

In this paper, we discussed a data center network structure using hybrid optoelectronic routers. We proposed a method to construct a data center network that uses hybrid optoelectronic routers efficiently. We investigated the effect of the network structure on the number of routers required to accommodate a large amount of traffic in a mega data center. The results indicate that multiple links from server racks are necessary to construct a large data center, even if hybrid optoelectronic routers are used.

Acknowledgments

This work was funded by the National Institute of Information and Communications Technology (NICT) R&D program, “Basic technologies for high-performance optoelectronic hybrid packet router”.

References

[1] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, “The cost of a cloud: research problems in data center networks,” ACM SIGCOMM Computer Communication Review, vol. 39, Jan. 2009, pp. 68–73.

[2] D. Abts, M. Marty, P. Wells, P. Klausler, and H. Liu, “Energy proportional datacenter networks,” in Proceedings of the 37th annual international symposium on computer architecture (ISCA 2010), June 2010, pp. 338–347.

 ISBN: 978-1-61208-460-2

[3] S. J. B. Yoo, “Optical packet and burst switching technologies for the future photonic internet,” Journal of Lightwave Technology, vol. 24, Dec. 2006, pp. 4468–4492.
[4] S. A. lbrahim et al., “100-Gb/s optical packet switching technologies for data center networks,” in Proceedings of Photonics in Switching, July 2014, pp. 1–2.
[5] Y. Ohsita and M. Murata, “Data center network topologies using optical packet switches,” in Proceedings of DCPerf, June 2012, pp. 57–64.
[6] N. Farrington et al., “Helios: a hybrid electrical/optical switch architecture for modular data centers,” ACM SIGCOMM Computer Communication Review, vol. 40, Oct. 2010, pp. 339–350.
[7] G. Wang et al., “c-through: Part-time optics in data centers,” in Proceedings of ACM SIGCOMM, Oct. 2010, pp. 327–338.
[8] T. Benson, A. Anand, A. Akella, and M. Zhang, “MicroTE: Fine Grained Traffic Engineering for Data Centers,” in Proceedings of ACM CoNEXT, Dec. 2011, pp. 1–12.
[9] C. Guillemot et al., “Transparent optical packet switching: The european ACTS KEOPS project approach,” Journal of Lightwave Technology, vol. 16, Dec. 1998, pp. 2117–2134.
[10] Z. Zhu et al., “Rf photonics signal processing in subcarrier multiplexed optical-label switching communication systems,” Journal of Lightwave Technology, vol. 21, Dec. 2003, pp. 3155–3166.
[11] Z. Pan, H. Yang, Z. Zhu, and S. J. B. Yoo, “Demonstration of an optical-label switching router with multicast and contention resolution at mixed data rates,” IEEE Photonics Technology Letters, vol. 18, Jan. 2006, pp. 307–309.
[12] K. Xi, Y. H. Kao, M. Yang, and H. J. Chao, “Petabit optical switch for data center networks.” Technical Report, Polytechnic Institute of New York University, http://eeweb.poly.edu/~chao/publications/petasw.pdf.
[13] X. Ye et al., “DOS: a scalable optical switch for datacenters,” in Proceedings of ANCS, Oct. 2010, pp. 1–12.
[14] O. Liboiron-Ladouceur, I. Cerutti, P. G. Raponi, N. Andriolli, and P. Castoldi, “Energy-efficient design of a scalable optical multiplane interconnection architecture,” IEEE Journal of Selected Topics in Quantum Electronics, vol. 17, Mar. 2011, pp. 377–383.
[15] C. Guo et al., “BCube: A high performance, server-centric network architecture for modular data centers,” ACM SIGCOMM Computer Communication Review, vol. 39, Aug. 2009, pp. 63–74.
[16] M. Kodialam, T. V. Lakshman, and S. Sengupta, “Efficient and robust routing of highly variable traffic,” in Proceedings of HotNets, Nov. 2004, pp. 1–6.

 ISBN: 978-1-61208-460-2

The Impact of Public Cloud Price Schemes on Multi-Tenancy

Uwe Hohenstein, Stefan Appel
Corporate Technology
Siemens AG, Corporate Technology, Otto-Hahn-Ring 6
D-81730 Muenchen, Germany
Email: {Uwe.Hohenstein,Stefan.Appel}@siemens.com

***Abstract*—Multi-tenancy is one key element to make Software-as-a-Service profitable. Multi-tenancy refers to an architecture model where one software instance serves a set of multiple clients of different organizations, i.e., tenants. This reduces the number of application instances and consequently saves operational costs. This paper focuses on using relational databases in multi-tenant architectures, thereby stressing the cost aspect in public cloud environments. Investigating the various price schemes of cloud providers, it illustrates the difficulties to achieve cost-efficient multi-tenancy. As a result, the broad variety of price factors and schemes lead to certain, very different strategies and require adapting multi-tenant architectures to fit the respective cloud providers' specifics.**

Keywords-multi-tenancy; databases; cost; SaaS.

I. INTRODUCTION

Software is more and more becoming an on-demand service drawn from the Internet, known as Software-as-a-Service (SaaS). SaaS is a delivery model that enables customers, the so-called *tenants*, to lease services without local installations and license costs. Tenants benefit from a "happy-go-lucky package": The SaaS vendor takes care of hardware and software installation, administration, and maintenance. Moreover, a tenant can use a service immediately due to a fast and automated provisioning [1].

Multi-tenancy is a software architecture principle allowing SaaS to make full use of the economy of scale: A shared infrastructure for several tenants saves operational cost due to an increased utilization of hardware resources and improved ease of maintenance [4]. Multi-tenancy is often considered as the key to SaaS.

Several authors, e.g., [23], discuss architectures according to what is shared by the tenants: the topmost web frontend, middle tier application server, and underlying database. Concerning the database, [5] describes a number of patterns, which support the implementation of multi-tenancy. We here distinguish between a *1-DB-per-tenant* and a *1-global-DB* strategy. The first one provides a database (DB) of its own for each tenant, thus achieving high data isolation, while several tenants share a common database without physical data isolation in the second variant. Further variants as discussed by [5] are irrelevant in this work.

In this paper, we report on industrial experiences when deploying SaaS in public clouds. Particularly, we focus on cost aspects of multi-tenancy for SaaS using a database because we feel economical aspects not appropriately tackled so far in research. Indeed, economic concerns are important as SaaS providers need to operate with high profit to remain competitive. We here elaborate on the huge differences of price schemes for relational database systems of public cloud providers and the impact on multi-tenancy. Even if various software engineering techniques propose NoSQL databases, relational systems are still often used in industrial applications, especially if being migrated to the Cloud.

Section II presents some related work and motivates why further investigations about cost aspects are necessary. We investigate the price models of various well-known public cloud providers in Section III: Amazon Web Services (AWS), HP Cloud, Microsoft Azure, and Oracle. The price information can be found at their homepages. We discuss in detail the impact of the price models on multi-tenancy strategies and the difficulties to optimize costs. In particular, we quantify the respective costs for implementing multi-tenancy by comparing a 1-DB-per-tenant strategy with a 1-global-DB. Finally, conclusions are drawn in Section IV.

II. RELATED WORK

The work in [4] considers performance isolation of tenants, scalability issues for tenants from different continents, security and data protection, configurability, and data isolation as the main challenges of multi-tenancy. These topics are well investigated. For instance, [17] investigates configurability of multi-tenant applications in case studies.

The possible variants of multi-tenancy have been described, among others, by [5]. Based on the number of tenants, the number of users per tenant, and the amount of data per tenant, [25] makes recommendations on the best multi-tenant variant to use.

Armbrust et al. [1] identify short-term billing as one of the novel features of cloud computing and [8] consider cost as one important research challenge for cloud computing. However, most works on economic issues around cloud computing focus on cost comparisons between cloud and on-premises and lease-or-buy decisions [22]. For example, [9] provides a framework that can be used to compare the costs of using a cloud with an in-house IT infrastructure, and [15] presents a formal mathematical model for the total cost of ownership (TCO) identifying various cost factors. Other authors such as [2][10], focus on deploying scientific applications on Amazon, thereby pointing at major cost drivers. [11] performs the TPC-W benchmark for a Web application with a backend database and compares the costs for operating the web application on several major cloud providers. A comparison of various equivalent architectural

 ISBN: 978-1-61208-460-2

solutions, however, using different components, such as queue and table storage, has been performed by [7]. The results show that the type of architecture can dramatically affect the operational cost.

Cost aspects in the context of multi-tenancy are tackled by [18][19]. They consider approaches to reduce resource consumption as a general cost driver, looking at the infrastructure, middleware and application tier, and what can be shared among tenants.

Another approach, discussed by [24], reduces costs by putting values of utilization and performance models in genetic algorithms.

The authors of [13] develop a method for selecting the best database in which a new tenant should be placed, while keeping the remaining database space as flexible as possible for placing further tenants. Their method reduces overall resource consumptions in multi-tenant environments. Cost factors taken into account are related to on-premises installations: hardware, power, lighting, air conditioning, etc.

Based on existing single-tenant applications, [3] stresses on another cost aspect for multi-tenant applications: maintenance efforts. The recurrence of maintenance tasks (e.g., patches or updates) raises operating cost.

The work in [6] recognizes a viable charging model being crucial for the profitability and sustainability for SaaS providers. Moreover, the costs for redesigning or developing software must not be ignored in SaaS pricing. Accordingly, [18] discusses a cost model for reengineering measures.

The challenges of calculating the costs each tenant generates for a SaaS application in a public cloud are discussed in [21]. This is indispensable to establish a profitable billing model for a SaaS application. The paper shows that only rudimentary support is available by cloud providers.

To sum up, the profitable aspects of multi-tenancy for SaaS providers are researched insufficiently. All the mentioned work is quite general and does mostly not take into account common public cloud platforms and their price schemes. Even best practices of cloud providers, for instance [16] and [19], do not support SaaS providers to reduce cost. As the next section illustrates, there is a strong need to investigate cost aspects for those platforms.

III. COST CONSIDERATIONS

Deploying multi-tenant applications in a public cloud causes expenses for the consumed resources, i.e., the pricing scheme of cloud providers comes into play. Unfortunately, the price schemes for cloud providers differ a lot and are based upon different factors such as the data volume, data transfer, etc. That is why we investigate the price schemes for databases of some major public cloud providers. The goal is to discuss variances in price schemes and how these affect multi-tenancy strategies for SaaS applications. We assume that each tenant demands a certain amount of database storage. We then compare storage that is provided using a dedicated database per tenant with a global database for all tenants to guide a decision.

Please note it is *not* our intention to compare different cloud providers with regard to costs or features. That is the reason why we keep the providers anonymous. There is also no common tool offered by all providers. Furthermore, the price schemes of cloud providers are quite diverging and incorporate different factors. We rather illustrate the variety of price schemes and service offerings leading to different architectures. This also means that the discussion of each offering has a different structure. Moreover, the prices are changing frequently, while the scheme usually remains stable. We here refer to the state as of September 2015.

We only consider resources that are available on-demand to fully benefit from the cloud. This excludes, e.g., reserved instances since those require long-term binding and thus impose a financial risk.

TABLE I. PRICE SCHEME FOR OFFERING 1.

Consumption	*Price*	*Additional GB*
0 to 100 MB	$4.995 (fix price)	
100 MB to 1 GB	$9.99 (fix price)	
1 to 10 GB	$9.99 for 1st 1 GB	$3.996
10 to 50 GB	$45.96 for 1st 10 GB	$1.996
50 to 150 GB	$125.88 for 1st 50 GB	$0.999

A. Offering 1

Offering 1 is a database server available as Platform-as-a-Service (PaaS) in a public cloud. PaaS include licenses and seems to be reasonable for multi-tenancy. Without PaaS, there is no elasticity since licenses must be ordered in time.

Table I presents the recent prices for a Microsoft SQL Server in the US East region. In addition, outgoing data is charged for each database individually with a decreasing rate. The first 5 GB are for free, each additional GB is charged with 8.7ct/GB and 8.3ct/GB above 10 TB, decreasing to 5ct/GB for more than 350TB. However, the cost reduction is insignificant unless there is extremely high outgoing transfer. The storage consumption is the main cost driver. Each database is paid for the amount of stored data. There is no cost difference between using one or several database servers for hosting the databases due to virtualization. We even could not detect any performance difference between placing databases on one virtual server or several ones. One has to pay for the consumed storage in every database – the number of databases and servers is irrelevant.
At a first glance, the price scheme suggests the same costs for 1-DB-per-tenant and 1-global-DB (keeping all tenants). There seems to be no cost benefit for sharing one database between several tenants, since SaaS providers are charged for the total amount of used storage. However, there are indeed higher costs for individual tenant databases since

- sizes larger than 1 GB are rounded up to full GBs;
- smaller databases are more expensive per GB than larger ones due to a progressive reduction.

Since pricing occurs in increments of 1 GB, hundred tenants with each a 1.1 GB database are charged with 100*2 GB, i.e., 100 * $13.986 = $1398.60 a month. In contrast, one database with 100 * 1.1GB = 110 GB is charged with $185.82, i.e., a total difference of $1212.78 or a difference of $12.13 for each tenant (per-tenant difference).

Figure 1 compares the costs of both strategies for various numbers of tenants (10,25,...,400). The x-axis represents the database size, the y-axis the per-tenant difference in US$,

 ISBN: 978-1-61208-460-2

i.e., the additional amount of money a SaaS provider has to pay for *each* tenant compared to a 1-global-DB strategy (note that the prices in Figure 1 must be multiplied by the number of tenants for total costs). The difference stays below $10 for tenant sizes up to 3 GB. The number of tenants is mostly irrelevant. This is why the lines are superposing; only the "10 tenants" line is noticeable. In the worst case, we have to pay $50 more for each tenant with a 1-DB-per-tenant strategy. A linear price drop occurs after 50 GB because even 1-DB-per-tenant uses larger and cheaper databases. Anyway, a 1-DB-per-tenant strategy can become quite expensive compared to a 1-global-DB strategy.

Please note the amount of *used* storage is charged. That is, an empty database is theoretically for free. However, even an empty database stores some administrative data so that the costs are effectively $4.995 per month (for < 100MB). Anyway, these are small starting costs for both a 1-DB-per-tenant and a 1-global-DB strategy.

There is no difference between provisioning a 10 GB and a 150 GB database from a cost point of view as the stored data counts. A 1-global-DB strategy, having the problem not to know how many tenants to serve, can start with 150 GB, thus avoiding the problem of later upgrading databases and possibly risking a downtime while having low upfront cost. Even for a 1-DB- per-tenant strategy, larger databases can be provisioned in order to be able to handle larger tenants without risk.

However, there is a limitation of 150 GB per database, which hinders putting a high amount of tenants with larger storage consumption in a single database. Reaching the limit requires splitting the database into two.

Along with this comes the challenge to determine a cost-efficient placement strategy. Assume an existing 90 GB database and that we need 40 and 30 GB more space for two further tenants: Putting 60 GB into the existing 90 GB database and 10 GB into a new one is the cheapest option with $225.75 + $45.96 = $271.71, more than $70 cheaper than using a new 40 GB and a new 30 GB database: $165.84 + $105.84 + $85.88 = $357.56. Even using a new 70 GB is more expensive with $311.70. An appropriate tenant placement strategy is to fill databases up to the 150 GB threshold, maybe minus some possible space for tenants' expansions, e.g., $286.58 = $205.80 (90+40 GB) + $85.88 (30 GB).

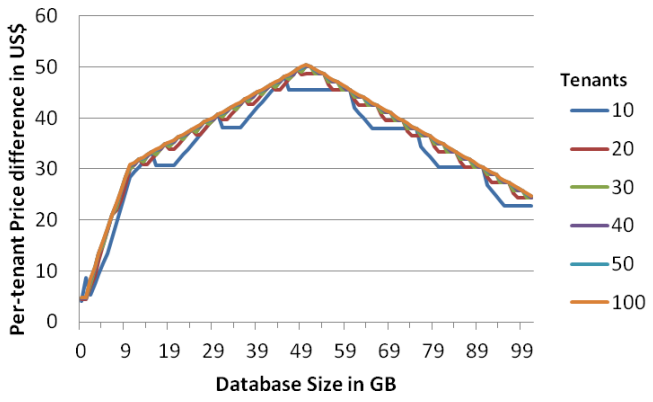


Figure 1. Price difference per tenant for Offering 1.

TABLE II. PRICE SCHEME FOR OFFERING 2.

Level	*Price/month*	*DB size*	*Session limit*	*Transaction rate / hour*
B	~$5	2	300	16,600
S0	~$15	250	600	31,260
S1	~$30	250	900	56,040
S2	~$75	250	1,200	154,200
S3	$150	250		306,000
P1	~$465	500	2,400	378.000
P2	~$930	500	4,800	820,800
P3	~$3,720	500	19,200	2,646,000

B. *Offering 2*

This candidate offers three tiers (**B**asic, **S**tandard, **P**remium). Table II shows the Microsoft SQL Server prices in the US East region for various performance levels inside.

Again, each individual database is paid according to the price scheme. But in contrast to Offering 1, the *provisioning* of the tier is relevant, not the effective storage consumption.

Figure 2 compares per-tenant costs for the 1-DB-per-tenant and 1-global-DB strategies in the same way as in Figure 1. 1-global-DB uses S0 databases, while 1-DB-per-tenant uses B (<= 2 GB) and S0 (> 2 GB) depending on the required size.

One of the worst cases that could happen for 1-DB-per-tenant is to have 100 tenants with 2.2 GB (S0) each, resulting in $1500 per month since each tenant cannot be satisfied with the B tier. In contrast, 1 * 220 GB (S0) for 1-global-DB costs $15. That is a per tenant difference of $14.85. However, it is unclear here whether an S0 level is sufficient for handling 100 tenants from a performance point of view.

A 1-DB-per-tenant strategy is about $5 more expensive if the size is lower than 2GB, and about $15 otherwise. The difference is never higher than $14.77, and drops to $12 for 50 GB and to $9 for 100 GB.

For each database, we have to pay at least $5 a month for at most 2 GB and $15 for up to 250 GB. The costs occur even for an empty database. These baseline costs have to be paid for a 1-gobal-DB, too, starting with the first tenant.

Especially for a 1-global-DB approach, a new challenge arises: Each service level determines not only an upper limit for the database size but also for the number of allowed parallel sessions and the number of (internal) worker threads. Furthermore, there is an impact on the transaction rate (cf. Table II). We have to stay below these limits. Upgrading the category in case of reaching the limit happens online, i.e., without any downtime – in theory: if the database size limit is reached, no further insertions are possible until the upgrade has finished. According to the documentation, such a migration can take several minutes up to hours depending on the database size. If the allowed number of sessions is reached, no further clients can connect unless sessions are released by other users. And if the transaction rate is insufficient, the performance will degrade. Hence, a prediction of tenants' data and usage behavior is required. The number of sessions might become the restrictive factor for a 1-global-DB strategy. In the following, we discuss the impact of the number of users and required sessions on costs by means of sample calculations.

 ISBN: 978-1-61208-460-2

TABLE III. COMPARISON OF CONFIGURATIONS.

	Configuration 1 vs. 2		*# sessions for 1 vs. 2*		*Transaction rate 1000/h (1 vs. 2)*	
a	5*S0	1*S2	3000	1200	156	154
b	2*S0	1*S1	1200	900	62	56
c	2*P1	1*P2	4800	4800	756	820
d	4*P2	1*P3	19200	19200	3283	2646
e	31*S0	1*P1	18600	2400	969	378

Keeping 100 tenants in 1*S0 offers 600 sessions, i.e., 6 sessions per tenant (which might be too small); the monthly costs are $15. We can scale-up to 1*P3 with 19,200 sessions, i.e., 192 per tenant, for a high price of $3720. To achieve the same number of sessions, we can also scale-out to 32*S0 for $480 or use 64*B for $360 if each database is smaller than 2 GB. In contrast, a pure 1-DB-per-tenant strategy for 100 tenants costs $500 for B: This seems to be affordable, especially because of 30,000 sessions. For the price of one P3, we also get 248*S0 databases with 148,000 sessions (6 times more than 1*P3) and a 3 times higher transaction rate of 7,752,480.

For serving 100 tenants with 20 parallel users each, we need 2000 sessions in total. We can achieve this by either 7*B (for $35), 4*S0 ($60), 3*S1 ($90), 1*P1 ($465), or 2*S2 ($500) with very different prices. A pure 1-DB-per-tenant for B is with $500 in the price area of the last two options, but supporting 300 sessions per tenant instead of 20.

Figure 3 illustrates the costs in US$ to achieve x sessions for 100 tenants. B1 represents a pure 1-DB-per-tenant strategy using B-level instances. The P levels are most expensive, even S2 is quite expensive. An obvious question is what the benefit of higher levels in the context of multi-tenancy is. Table III compares several configurations with same prices. There is no consistent behavior. However, several smaller machines seem to be superior to same priced larger ones with a few exceptions. One exception is row (c) where 1*P2 is a little better than 2*P1. More sessions can usually be achieved if n smaller tiers are used instead of one larger one for the same price.

Considering Table II again, we also notice that the session and transaction rates increase from tier to tier less proportional than the prices. Exceptions for transaction rates are S1->S2 and P1->P2. It seems to be reasonable to scale-out instead of scaling-up to obtain more sessions and transactions.

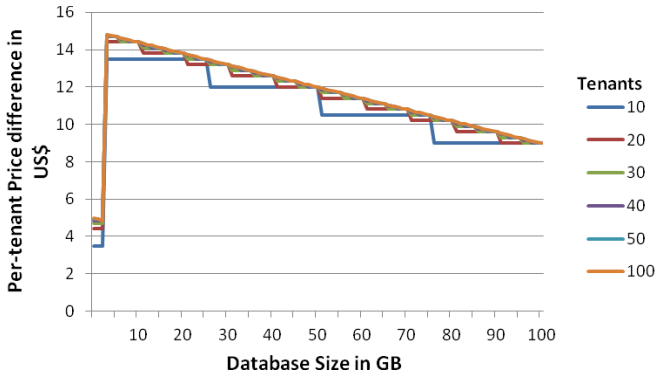


Figure 2. 1-DB-per-tenant vs. 1-global-DB for Offering 2.

TABLE IV. PRICE SCHEME FOR OFFERING 3.

Instance Type	*RAM*	*Storage*	*Price/month*
XS	1 GB	15 GB	$73
S	2 GB	30 GB	$146
M	4 GB	60 GB	$292
L	8 GB	120 GB	$584
XL	16 GB	240 GB	$1,168
XXL	32 GB	480 GB	$2,336

Another advantage is that baseline costs can be saved. A 1-global-DB strategy requires a high-level database with a high price already for the first tenant independent of the number of eventually stored tenants.

Indeed, it is difficult to derive a strategy for identifying a suitable configuration. Important questions arise:

- Is an upgrade possible in short time, without outage? This would allow for 1-global-DB to start small for few tenants and upgrade if performance suffers or the number of sessions increases. For 1-DB-per-tenant, we could start with B and upgrade to S0.
- Are only the session and transaction rates of a level relevant, or are there any other (invisible) performance metrics to consider? The documentation mentions only that the predictability, i.e., the consistency of response times, is increasing from B to Px, however being the same within a tier.

C. Offering 3

Offering 3 provides a MySQL database as PaaS. The regular prices are for virtualized databases on an hourly basis. The payment is based upon the following factors:

- The instance type, which limits the maximal database size and determines the RAM (cf. Table IV).
- The outgoing data transfer: the first GB is for free, we then pay 12ct/GB up to 10 TB, further GBs for 9ct up to 40 TB, etc.

In contrast to Offering 1, the provisioned storage is paid. The prices and the features increase with the instance type linearly, i.e., each next higher instance type doubles the RAM and maximal database size for a doubled price.

Comparing the strategies, we notice that 5 tenant databases à 15 GB (XS) are charged with $365. One global database à 75 GB is more expensive (!) with $584 since we are forced to provision a 120GB (L) database. The difference per tenant is $43.80. However, using 15GB (XS) increments for 1-global-DB, we can achieve the cheaper price.

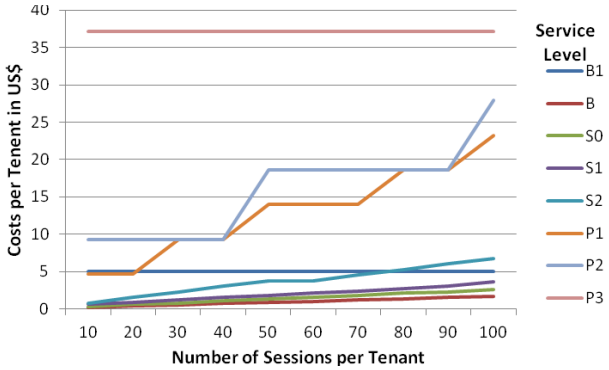


Figure 3. Costs to achieve x sessions per tenant for Offering 2.

TABLE V. PRICE SCHEME FOR OFFERING 4.

Type	*Max database size*	*Data Transfer*	*Price/month*
5GB	5 GB	30 GB	$175
20GB	20 GB	120 GB	$900
50GB	50 GB	300 GB	$2,000

Hence, we should use XS partitions in order not to pay for unused storage. Thus, an appropriate cost strategy for 1-global-DB is to fill XS databases one by one with tenants. However, this has architectural implications in order to connect each tenant to the right database instances. A 1-DB-per-tenant approach could also benefit that way. There is no need to use larger instances unless we do not want to spread tenant data across databases due to implementation effort.

A worst case scenario is storing 15 tenants with 100MB each. 1-DB-per-tenant is charged with $1095 = (15*XS), while one global XS database costs $73 for 1.5 GB. That is a difference per tenant of $68.13.

Figure 4 illustrates that 1-DB-per-tenant, compared to 1-global-DB based upon XS databases, is more expensive for sizes much smaller than the storage threshold. Reaching the threshold, the difference diminishes. Hence, it is reasonable to use one database for each tenant if the storage size is near a threshold. In summary, we observe larger per-tenant differences depending on database sizes. The range where the difference stays below $20 is very small. Moreover, the variances for different numbers of tenants are small.

An incremental acquisition of XS databases even saves baseline costs. However, it is an open issue to be investigated whether larger instances provide a better performance. The time for upgrading from one instance type to another is not important here.

D. Offering 4

Three database types are available, each limiting the maximal amount of storage. Table V shows that each type also limits the allowed data transfer.

A comparison of the types gives some first insights: 20GB is 5 times more expensive than 5GB, but offers only 4 times more data transfer and storage. 50GB is 2.2 times more expensive than 20GB, but offers 2.5 times more data transfer and storage. And 50GB is 11 times more expensive than 5GB, but offers only 10 times more resources. Hence, 20GB has the worst price ratio, 5GB the best one. Obviously, using 5GB databases seems to be reasonable for either strategy unless we do not want to spread tenant data across databases.

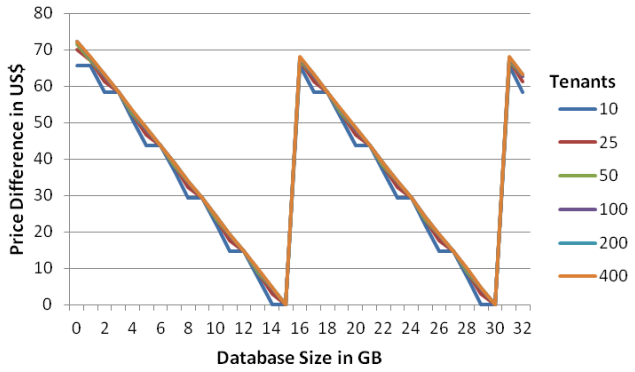


Figure 4. Price diff. of 1-DB-per-tenant vs. 1-global-DB for Offering 3.

TABLE VI. COMPARISON OF SAMPLE CONFIGURATIONS

Config	*#tenants*	*database size*	*Costs*	*Data transfer*	*Per-tenant costs*
100*5GB	100	1 GB	17,500	3000	175
2*50GB			4,000	600	40
5*20GB			4,500	600	45
20*5GB			3,500	600	35
200*5GB	200	4 GB	35,000	6000	175
16*50GB			32,000	4800	160
40*20GB			36,000	4800	180
100*5GB	100	5 GB	17,500	3000	175
10*50GB			20,000	3000	200
25*20GB			22,500	3000	225

Table VI compares a 1-DB-per-tenant configuration (the first lines) with others. For 200 tenants à 4GB, using 20GB databases is more expensive than 1-DB-per-tenant; the same holds for 100 tenants à 5GB.

Figure 5 summarizes the price-per-tenant differences if 5GB increments are used. A 1-DB-per-tenant strategy is only reasonable if the database size is near a multiple of 5 GB, or if the required data transfer is high. The larger the distance is, the higher will be the per-tenant costs compared to a 1-global-DB. This saw tooth behavior is repeating. The number of tenants has again no impact.

Since the data transfer is limited by the instance type, a challenge arises for the 1-global-DB strategy: this can stop several or all tenants from accessing the database. Additional data transfer cannot be acquired even for extra charges.

A possible strategy for 1-global-DB is to start with 5GB and to add further ones later; this means less upfront costs. Moreover, 5GB is the cheapest category wrt. gains. Please note that downsizing is not possible. This causes further costs in case a tenant stops using the SaaS service.

E. Offering 5

Offering 5 provides a virtual machine (VM) with a Microsoft SQL Server for various operating systems. The price model is quite complex covering several factors.

At first, a VM has to be chosen for hosting the database server. Table VII summarizes the prices for a Windows OS in the East US region. Each tier has a different number of virtual cores (vCores), RAM, and temporary disk space. A0-A7 covers the standard tier; A0-A4 are also available in a basic tier with little lower prices ($13-$440) than the standard tier.

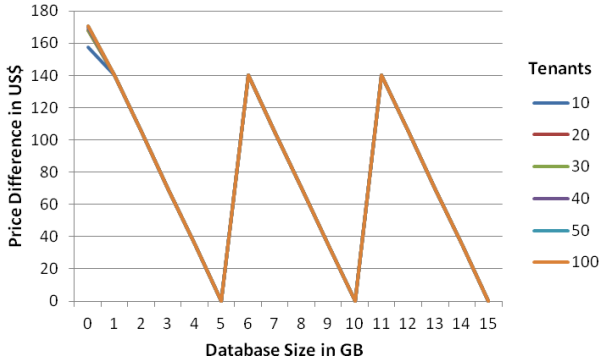


Figure 5. Price diff. of 1-DB-per-tenant vs. 1-global-DB for Offering 5.

TABLE VII. PRICE SCHEME FOR OFFERING 5

Tier	*vCores*	*RAM*	*TempDisk*	*price/month*	*#disks*
A0	1	768 MB	20 GB	$15	1
A1	1	1.75 GB	70 GB	$67	2
A2	2	3.5 GB	135 GB	$134	4
A3	4	7 GB	285 GB	$268	8
A4	8	14 GB	605 GB	$536	16
A5	2	14 GB	135 GB	$246	4
A6	4	28 GB	285 GB	$492	8
A7	8	56 GB	605 GB	$983	16
A8	8	56 GB	382 GB	$1,823	16
A9	16	112 GB	382 GB	$3,646	16
D1	1	3.5 GB	50 GB	$127	1
D2	2	7 GB	100 GB	$254	2
D3	4	14 GB	200 GB	$509	4
D4	8	28 GB	400 GB	$1,018	8
D11	2	14 GB	100 GB	$600	2
D12	4	28 GB	200 GB	$1,080	4
D13	8	56 GB	400 GB	$1,943	8
D14	16	112 GB	800 GB	$2,611	15

A8 and A9 are network-optimized instances adding an InfiniBand network with remote direct memory access (RDMA) technology. The D-tier is compute-optimized with 60% faster CPUs, more memory, and a local SSD. An OS disk space of 127 GB is available and must be paid with ignorable 2.4ct per GB/month.

Furthermore, the server is charged per minute. The prices depend on the number of cores of the used VM: $298 for 1-4 core machines, $595 for 8 cores, and $1190 for 16 cores for a SQL Server Standard Edition in a month. The Enterprise Edition is more expensive, e.g., $4464 for a 16-core VM.

Additional costs occur for attached storage. There is a maximum of number of 1TB data disks (#disks in Table VII). The costs can be neglected with 5ct/GB-month for the first 1000 TB of consumed storage in a page blob. The costs are thus dominated by other factors than disk space.

As a major difference to previous price schemes, a database *server* is provisioned and paid instead of a single database. The database server offers full control like operated on-premises. Several databases can be managed in that server. This directly implies that a 1-DB-per-tenant strategy is feasible, i.e., each tenant can obtain a database of its own with individual credentials. A strong isolation is thus given without any extra charge. As a consequence, a strategy could be to use one database server and set up one database for each tenant until performance decreases.

Instead of comparing 1-DB-per-tenant and 1-global-DB, we have to consider how many database servers (hosting several databases) of what tier we have to apply for the expected number of tenants and users. One strategy could be to start with a small VM and increase the instance type with the number of tenants. This implies that such an upgrade is possible within short time.

If an upgrade could cause a downtime, we have to decide whether to use several small VMs or few larger ones from a cost perspective. There are high minimal costs of at least $311 per month for each database server ($13 for the smallest Windows VM A0 Basic plus the database server). A high number of tenants/databases will obviously require larger VMs, leading to higher baseline costs.

TABLE VIII. CONFIGURATIONS TO ACHIEVE 112 GB RAM.

configuration	*# vCores*	*Price (decreasing)*
8*D11	16	$4,800
8*A4	64	$4,288
8*D3	32	$4,072
2*D13	16	$3,886
A9	16	$3,646
2*A8	16	$3,646
D14	16	$2,611
8*A5	16	$1,968
2*A7	16	$1,966

The question is what configuration is sufficient for a given number of tenants and amounts of data. Unfortunately, no performance hints are provided to ease the decision. It might be better to provision a larger VM since it can be used for other purposes as well if being idle. A brief evaluation of an SQL Server Standard Edition shows the following:

a) 1*A9 costs $4836 ($3646 for the VM, $1190 for the SQL Server), offering 16 vCores and 112 GB RAM.

b) For the same price we get 8.5*A3 with ~34 vCores and ~60 GB RAM in total.

c) Alternatively, we can also purchase 3*A7 with 24 vCores and 168 GB RAM.

d) 13*A1 comes for $4745 with ~23 GB RAM and 13 vCores.

(d) offers the least equipment for the price because of the high number of database servers, each for $298. Options (b) and (c) favor either vCores or RAM. In general it looks reasonable to avoid high-class VMs and to use several middle-class VMs.

An incremental provisioning can also help to reduce upfront baseline costs, which occur already with the first tenant. However, a deeper empirical performance evaluation is necessary due to further open questions:

- What VM should be chosen? Table VIII shows several options to achieve 112 GB RAM with very different prices and numbers of vCores.
- Similarly, there are many variants for 14 or 28 GB RAM, each yielding different vCores with prices ranging from $246 to $600 for 14GB RAM, and from $492 to $1089 for 28 GB RAM.
- Finally, when pursuing the approach with one or few small servers, it is indispensable to know how long it last to upgrade the category of a VM.

F. Offering 6

Similar to Offering 5, this option again offers a VM running a database server, however, with several differences regarding pricing. Several database systems such as MySQL, Oracle, PostgreSQL, and SQL Server are supported in various database instance classes of three categories: Micro, Standard, and Memory-optimized.

The prices depend on the chosen instance type, the type of database server, and the region. The prices for a MySQL database in the US East region are presented in Table IX. The underlying VM and the MySQL license are already included in the price. The instance class determines the number of virtual CPUs (vCPU) and the main memory.

 ISBN: 978-1-61208-460-2

TABLE IX. PRICE SCHEME FOR OFFERING 6.

Category	*Instance type*	*Price / month*	*vCPU*	*RAM*
Micro	XS	$12.41	1	1
	S	$24.82	1	2
	M	$49.64	2	4
Standard	M	$65.70	1	3.75
	L	$135.05	2	7.5
	XL	$260.10	4	15
	XXL	$520.20	8	30
Memory-optimized (MemOpt)	L	$175.20	2	15
	XL	$346.75	4	30.5
	XXL	$689.85	8	61
	4XL	$1379.70	16	122
	8XL	$2759.40	32	244

An additional cost factor is the outgoing data transfer to the Internet of 9ct/GB: prices decrease with the amount, 1 GB-month is for free. The price decrease for larger volumes is insignificant.

Furthermore, the amount of data is charged according to two alternative classes of database storage:

a) General purpose for 11.5ct per GB-month with a range from 5 GB to 3 TB; 3 IOPS per GB are included.

b) Provisioned IOPS for 12.5ct per GB-month and additional $0.10 per requested IOPS/month, with a range from 100 GB to 3 TB and 1,000 IOPS to 30,000 IOPS.

IOPS (IO per second) determines an upper limit for IO. IO itself is not charged.

Again, we have to decide how many servers are reasonable. The baseline costs for each database server are determined by the minimal settings: The smallest installation in terms of cost for MySQL is Micro XS with $12.41/month. Provisioned IOPS storage is available at a minimum of 100 GB and 1000 IOPS, i.e., $12.50 (100*12.5ct) plus $100 (1000 IOPS à 10ct) ending up with costs of at least $112.50 per server. Using alternate general purpose storage, we have to provision at least 5 GB for 57.5ct (5*11.5ct); but then only 15 IOPS are available (see (a) above). Hence, setting up a minimal MySQL server, e.g., for each tenant, comes with at least $13 using general purpose storage, while provisioned storage is much more expensive with $125.

A calculation and comparison of using one or several database servers is difficult since several factors are unclear. A high-end server might be more appropriate since the high provisioning cost for storage occur only once. According to the documentation, the network performance also increases with a higher instance class. However, many smaller servers avoid higher, tenant-independent upfront investments for a larger instance and required IOPS.

Table X shows some configurations with similar monthly costs. The provided equipment differs a lot. Obviously, (a) is better equipped than (b). But each of (a) and (d) has an advantage for vCPUs or RAM, respectively.

It is important to note that the *provisioned* numbers are relevant, not the effective usage. This means, the required storage for each tenant has to be estimated in order not to overpay for unused resources. The same holds for the IOPS rate. These costs occur already for the first tenant independently of consumed resources.

TABLE X. COMPARISON OF CONFIGURATIONS.

	Configuration	*vCPUs*	*RAM*	*Costs*
a	1* 8XL (mem-opt)	32	244	$2759
b	2 * 4XL (standard)	16	244	$2759
c	20 * L (standard)	40	150	$2704
d	42 * M (standard)	42	157	$2759

One strategy could be to start with small servers and increase the instance type and configuration if necessary. This implies that such an upgrade is possible within short time and without downtime in the meantime. Otherwise, a larger machine with larger storage and IOPS can be provisioned from the beginning, however, causing high starting costs already for some few tenants.

Another strategy is to use one smaller server for each tenant, e.g., for $13. Then, the expenses increase tenant by tenant. This also gives more flexibility for provisioning IOPS according to tenants' requirements.

An important question is what IOPS rate is sufficient since the IOPS rate is a limiting factor: Throttling of users can occur if the limit is reached. Obviously, keeping several tenants in one server requires higher IOPS rates. It is unclear what the advantage of provisioned storage is compared to general purpose storage. From a pure cost perspective, 6000 IOPS are charged with $600 for provisioned storage. To achieve 6000 IOPS with general purpose storage, we have to use 2TB (remind the factor in (a)), i.e., being much cheaper with $230 and already including storage. Since there is an upper database limit of 3 TB in any case, general purpose IOPS ends with 9,000 IOPS; provisioned storage can handle up to 30,000 IOPS.

IV. CONCLUSIONS

This paper took a deeper look into the price schemes of popular cloud database providers and investigated their cost impact on multi-tenancy. We thereby focused on storing tenants' data in relational databases. We showed that a cost-efficient database deployment for multi-tenancy heavily depends on providers due to very different price schemes. Several differences become apparent.

- Offering 2-6 charge for provisioned storage, i.e., upfront costs occur even if small data is stored. In contrast, Offering 1 charge for storage consumption which avoids starting costs instead.
- Sometimes, databases are paid (cf. Offerings 1-4); sometimes whole DB servers are provisioned (cf. Offering 5 and 6) so that several isolated databases can be managed with specific credentials.
- Offerings 2 and 4 define certain limits on transaction rates, data transfer, or number of sessions. Reaching such a limit could stop a SaaS application for serving tenants.
- For Offerings 3, 5, and 6, equipment such as RAM increases with each level, while this is not controllable and visible in Offerings 1, 2, and 4.

There are direct cost factors such as storage, IOPS, sessions, cores, or data transfer, i.e., they are directly part of the price scheme. We detected indirect cost factors, too. For example, it might be necessary to use and pay a larger virtual

machine (VM) in order to achieve a certain transaction rate, e.g., Offering 2.

The broad spectrum of price schemes makes it difficult to find an appropriate provider-independent cost-optimized configuration for multi-tenant applications. However, we could present some analyses comparing the cost of a 1-DB-per-tenant and a 1-global-DB strategy and displaying the characteristics for different tenant sizes. The results also have a strong impact on the cloud provider selection. For example, if a strong isolation is requested, a provider with too high prices for a 1 DB-per-tenant strategy might not be qualified for a selection.

As a consequence, it is difficult to select *the* best provider from the cost perspective. But we think that our analysis helps architects of multi-tenant software to decide upon a cloud offering for the anticipated requirements. Besides architects, cloud providers can benefit from our analysis when it comes to adjust their service offerings.

This all affects portability of SaaS applications, too. It is not easy to define an economic provider-independent strategy for multi-tenancy. Furthermore, architectures must take into account several aspects. For example, monitoring consumption becomes necessary because of thresholds such as a database upper limit of parallel sessions, IO limits, or any other type of throttling. This is indispensable to react in time if a threshold is reached because a service is in danger of being stopped.

Future work will tackle open questions, including practical investigations. One important question is about the provisioning time. This point is relevant in any strategy since additional databases have to be acquired. Similarly, upgrading a database level is important for saving upfront costs.

Finally, we intend to collect further challenges from an industrial perspective.

REFERENCES

[1] M. Armbrust et al., "A View of Cloud Computing," Communications of the ACM, 53(4), April 2010, pp. 50-58.

[2] B. Berriman, G. Juve, E. Deelman, M. Regelson, and P. Plavchan, "The Application of Cloud Computing to Astronomy: A Study of Cost and Performance," Proc. of 6th IEEE Int. Conf. on e-Science, 2010, pp. 1-7.

[3] C. Bezemer, A. Zaidman, B. Platzbeecke, T. Hurkmans, and A. Hart, "Enabling Multitenancy: An Industrial Experience Report," in: Technical Report of Delft Uni. of Technology, TUD-SERG-2010-030, 2010.

[4] C. Bezemer and A. Zaidman, "Challenges of Reengineering into Multitenant SaaS Applications," in: Technical Report of Delft Uni. of Technology, TUD-SERG-2010-012, 2010.

[5] F. Chong, G. Carraro, and R, Wolter, "Multi-Tenant Data Architecture," June 2006, http://msdn.microsoft.com/en-us /library/aa479086.aspx [retrieved: February 2016]

[6] T. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and Challenges," in Proc. 24th Int. Conf. on Advanced Information Networking and Applications, 2010, pp. 27-33.

[7] U. Hohenstein, R. Krummenacher, L. Mittermeier, and S. Dippl, "Choosing the Right Cloud Architecture - A Cost Perspective," in Proc. on Cloud Computing and Services Science (CLOSER), 2012, pp. 334-344.

[8] A. Khajeh-Hosseini, I. Sommerville, and I. Sriram, "Research Challenges for Enterprise Cloud Computing," in Proc. 1st ACM Symposium on Cloud Computing, SOCC 2010, Indianapolis, pp. 450-457.

[9] M. Klems, J. Nimis, and S. Tai, "Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing," in Designing E-Business Systems. Markets, Services, and Networks, Lecture Notes in Business Information Processing, Vol. 22, 2008, pp.110-123.

[10] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. Anderson, "Cost-Benefit Analysis of Cloud Computing versus Desktop Grids," in Proc. of the 2009 IEEE Int. Symp. on Parallel&Distributed Processing, May 2009, pp 1-12.

[11] D. Kossmann, T. Kraska, and S. Loesing, "An Evaluation of Alternative Architectures for Transaction in Processing in the Cloud," ACM SIGMOD 2010, pp. 579-590.

[12] R. Krebs, C. Momm, and S. Kounev, "Architectural Concerns in Multi-Tenant SaaS Applications," in CLOSER 2012, pp. 426-431.

[13] T. Kwok and A. Mohindra, "Resource Calculations With Constraints, and Placement of Tenants and Instances for Multi-Tenant SaaS Application," in Proc. Int. Conf. on Service-Oriented Computing, (ICSOC) 2008. LNCS, vol. 5364, pp. 633-648.

[14] M. Lindner, F. Galán, and C. Chapman, "The Cloud Supply Chain: A Framework for Information, Monitoring, Accounting and Billing," in Proc. on ICST Cloud Computing, 2010, pp. 1-22.

[15] B. Martens, M., Walterbusch, and F. Teuteberg, "Evaluating Cloud Computing Services from a Total Cost of Ownership Perspective," 45th Hawaii International Conference on System Sciences (HICSS-45), 2012, pp. 1564-1572.

[16] Microsoft, "Developing Multitenant Applications on Windows Azure". http://msdn.microsoft.com/en-us/library /ff966499.aspx [retrieved: January 2016]

[17] R. Mietzner, F. Leymann, and M. Papazoglou, "Defining Composite Configurable SaaS Application Packages Using SCA, Variability Descriptors and Multi-Tenancy Patterns," in 3rd Int. Conf. on Internet and Web Applications and Services (ICIW), 2008, pp. 156-161

[18] C. Momm and R. Krebs, "A Qualitative Discussion of Different Approaches for Implementing Multi-Tenant SaaS Offerings," in Proc. Software Engineering 2011, pp. 139-150.

[19] C. Osipov, G. Goldszmidt, M. Taylor, and I. Poddar, "Develop and Deploy Multi-Tenant Web-Delivered Solutions Using IBM Middleware: Part 2: Approaches for Enabling Multi-Tenancy," in: IBM's technical library, 2009.

[20] A. Schwanengel, U. Hohenstein, and M. Jaeger, "Automated Load Adaptation for Cloud Environments in Regard of Cost Models," in Proc. on CLOSER, 2012, pp. 562-567.

[21] A. Schwanengel and U. Hohenstein, "Challenges with Tenant-Specific Cost Determination in Multi-Tenant Applications," in 4th Int. Conf. on Cloud Computing, Grids and Virtualization, Valencia (2013), pp. 36-42.

[22] E. Walker, "The Real Cost of a CPU Hour," Computer 2009, Vol. 42(4), pp. 35-41.

[23] S. Walraven, E. Truyen, and W. Joosen, "A Middleware Layer for Flexible and Cost-Efficient Multi-Tenant Applications," in Proc. on Middleware, 2011 (LNCS 7049), pp. 370-389.

[24] D. Westermann and C. Momm, "Using Software Performance Curves for Dependable and Cost-Efficient Service Hosting," in Proc. on Quality of Service-Oriented Software Systems (QUASOSS), 2010, pp. 1-6.

[25] Z. Wang et al, "A Study and Performance Evaluation of the Multi-Tenant Data Tier Design Pattern for Service Oriented Computing," in IEEE Int. Conf. On eBusiness Engineering, (ICEBE) 2008, 94-101

 ISBN: 978-1-61208-460-2

A Novel Framework for Simulating Computing Infrastructure and Network Data Flows Targeted on Cloud Computing

Peter Krauß
Karlsruhe Institute of Technology
Steinbuch Centre for Computing
76128 Karlsruhe
Email: peter.krauss@kit.edu

Tobias Kurze
Karlsruhe Institute of Technology
Library
76131 Karlsruhe
Email: kurze@kit.edu

Achim Streit
Karlsruhe Institute of Technology
Steinbuch Centre for Computing
76128 Karlsruhe
Email: achim.streit@kit.edu

Bernhard Neumair
Karlsruhe Institute of Technology
Steinbuch Centre for Computing
76128 Karlsruhe
Email: bernhard.neumair@kit.edu

***Abstract*— Understanding how computing infrastructure decisions influence overall performance and cost can be very difficult. Simulation techniques are an important tool to help analyse and evaluate different infrastructure configurations and deployment scenarios. As diversity of cloud computing resources is growing, the space of possible infrastructure configurations becomes larger and finding the best solution becomes harder. In our previous paper, we presented a framework to benchmark cloud resources to obtain objective and comparable results. Based on those results, our simulation framework allows to model applications and to estimate their performance. Compared to other infrastructure or cloud simulation tools our framework excels when it comes to the simulation of network data flows.**

Keywords–Broad band networks, quality of service, infrastructure, cloud, simulation, framework

I. INTRODUCTION

Due to the rise of cloud computing during the last few years - for a widely accepted definition of the term see [1][2] - and its further growing adoption in commercial, industrial and research endeavors [3][4], there are more and more different infrastructure offerings and possible applications. Different types of applications have different requirements on the underlying (virtual) hardware and are structured differently. Finding deployment policies that deliver good performance and are cost-efficient under varying conditions is challenging because of multiple reasons.

First, the number of possible deployments for a given application respectively for workload executing systems, grows rapidly with the number of components of an application as well as with the number of available resources to choose from. Secondly, clouds do not always deliver the same performance and are subject to varying demand. Further, as cloud resources are delivered via public broadband networks, it is hard or even impossible to reproduce identical test conditions. As it is neither practical to try out each and every possible deployment nor feasible to reproduce an identical test environment, simulating the respective scenarios is the better approach. Therefore, we developed an infrastructure simulation framework that focuses on cloud environments, but in principle it can be used for any kind of infrastructure. The framework allows the simulation of large-scale applications that may be deployed across different cloud providers.

Our framework provides informations about the complete behavior of a configured workload running on a user defined infrastructure. During the simulation, a user can access all the status information he might need, this includes load on disk, CPU, memory as well as network interfaces. This allows a user to attach e.g., cost models or simulate bad or malfunctioning hardware. Our approach is unique in the way it models networks and is very lightweight compared to other tools.

The paper is structured as follows:
A short overview of cloud respectively infrastructure simulation tools is given in Section II. Next, we present our infrastructure model and introduce resource providers and our network simulation model. Then, we give a description of how workloads are specified, followed by the section dedicated to the implementation of the simulator. Next, in addition to the theoretical model, we describe how calibration of the simulator is done and show the validity of the framework. Lastly a short summary including a discussion of the validation's results is given.

II. RELATED WORK

In the field of cloud simulation, a lot of research with varying focus has already been done. Lim et al. developed MDCSim [5], a simulation platform for multi-tier data centers that allows performance and power consumption analysis. The simulator is event based and designed as a three-layer architecture – user layer, kernel layer, communication layer. It is able to capture details such as kernel level scheduling

 ISBN: 978-1-61208-460-2

strategies. For communication, the simulator supports IBA and Ethernet over TCP/IP. By changing the timing parameters other protocols can be incorporated. For kernel requests, the simulator considers multiple CPUs as a single resource and is modeled according to the Linux scheduler 2.6.9.

Another well known toolkit for modeling and simulating cloud environments is CloudSim [6][7]. It is event-based and allows to model data centers, virtual machines and resource provisioning policies. Furthermore it is possible to model federation of clouds resp. inter-networked clouds. Allocation of virtual machines to hosts is done based on a First-Come-First-Serve basis. CloudSim implements time-shared as well as space-shared policies at host and at VM level to assign resources. Besides aforementioned aspects, CloudSim models the cloud market based on a multi-layered design, whereby the first layer represents costs per unit related to the IaaS model and the second layer comprises costs related to the SaaS model. Network behavior is modeled using a latency matrix and a message will be forwarded by the event management engine after a delay specified by the according entry in the latency matrix.

NetworkCloudSim [8] has been developed by Garg and Buyya and is an extension for CloudSim providing a network and generalized application model, allowing a more precise evaluation of scheduling and provisioning policies. NetworkCloudSim adds three main entities: Switch, NetworkDatacenter and NetworkDatacenterBroker. Also new classes to model networks have been added to the original classes of CloudSim. This allows a better modelling of applications that rely on specific means of communication such as MPI for example.

DartCSim+ [9] is another enhancement of CloudSim. The authors claim to have overcome limitations of CloudSim in regard to three aspects: – simultaneous support of power and network model is not possible – simulation of network components is not power-aware – migration does not take into account network overheads To overcame aforementioned limitations, the authors developed a range of entities as extension for CloudSim.

Wickremasinghe et al. propose another CloudSim related tool named CloudAnalyst [10], that helps studying the behaviour of large-scale cloud applications. CloudAnalyst takes into account the geographic dìstribution of an application and of its users. CloudAnalyst provides a graphical user interface that allows users to model their experimental setups.

Nunez et al. developed another cloud simulator called iCanCloud [11][12] with focus on lage experiment setups. iCanCloud features a global hypervisor that can integrate any brokering policy and is configurable through a graphical user interface. It also provides a POSIX-based API and supports configurations for different storage systems such as remote file systems like NFS or parallel files systems and RAID configurations.

In our own previous work [13] we created a performance measuring tool. The publication contains a description of the used architecture to monitor a large amount of cloud resources over time. We were able to show that the perfromance of virtualized infrastructure resources does vary over time, even though it should be constant, as well as the performance of network interconnects between virtual machines. This aspect is not covered by common simulation frameworks. The framework presented in this paper is capable of taking the factors shown in [13] into account. We use the measured performance values from [13] to calibrate our simulation framework presented in this publication.

III. Infrastructure Model

Our model describes an infrastructure cloud environment as a hierarchy of entities. The entities on the lowest level provide resources, which are in our model *calculation power* in units called *cpu cycles*, *ephemeral memory* in units called *mbytes*, *persistent storage* space in multiples of mbytes and finally *network traffic* which is described by transfers between network enabling entities characterized by a bandwidth in units of *mbytes per time step*. The entities that provide those resources are called *vCPU*, *vMemory*, *vStorage* and *vNet* which creates instances of *vTransfer* objects.

On the next level, the model introduces another entity called *vMachine*. This entity can be associated with any combination and any amount of entities of the level below and is a loose equvivalent to a virtual machine in a real cloud. Further a *vMachine* can be associated with so called *vWorkload* objects. Those objects contain a description of a workload and are described in the course of this paper (see Section IV).

The next layer contains *vDatacenter* objects that in turn contain *vMachine* objects. A *vDatacenter* in addition is associated with two coordinates that express a location in a 2D-space and can be used to position different datacenters relative to each other. The distance between datacenters is factored in when data transfers happen, since previous works showed that this can affect transfers [13].

Beside those layers, the model is based on a global environment object containing all entities related to the simulated infrastructure. This environment provides a global, steadily increasing counter to represent the simulation time in logical seconds. With each step the timer advances by one and the simulation progresses. The environment makes sure that all objects associated with it are notified at least once per time step about the new simulation time so they will update and possibly advance their state.

In the following subsections, major parts of this model will be explained in deeper detail.

A. Resource Providers

As mentioned before the model is based on four resources that are provided by four resource providers. These entities are always associated with a maximum capacity and a provisioning speed. The first describes how many units of a certain resource are provided whereas the latter expresses at what speed this resource can be delivered. In addition, each resource can either be ephemeral or static. Ephemeral resources reset their capacity at the beginning of each new time step while static resources do not. In our model the following four resource providers are defined:

vCPU As mentioned, this object provides *calculation power* in units of *cpu cycles*. The resource is ephemeral and resets to its predefined maximum capacity with every

simulation step and the resource can be consumed at instant speed.

vMemory To represent a machine's internal memory, we introduced a *vMemory* resource provider. In contrast to physical RAM, this resource does reset its capacity with every time step due to how workloads are designed in our model: A *vWorkload* does not allocate and free memory but it provides an absolute value representing how much memory is required at a certain point in time. All operations on a *vMemory* object are performed instantly.

vStorage A *vStorage* object represents a comparably slow and limited storage device for example a hard disk or solid state drive. The provider's operational speed is capped at a predefined rate (throughput) which in addition is shared among all entities currently using the resource.

vNet This object represents a network device which can transfer a given amount of data per time step. The object's bandwidth is shared among all currently active transfers and each transfer's speed further depends on the receiver's bandwidth. Since proper network simulation is a major part of our framework this will be discussed in detail in a following section (see Section III-B). We assume a physical network card to operate in duplex mode, sending and receiving speeds are not correlated in any way and are thus modeled as two independent vNet instances that may be configured with different bandwidths.

vTransfer vTransfer objects contain all information that describe the state of ongoing data transfer, such as current transmission speed and associated vNet instances (origin/destination). In contrast to the previously presented objects, vCPU, vMemory, vStorage and vNet, a vTransfer does not have a representation in real hardware and is a completly virtual construct.

Since access to the 'fast' resources *calculation power* and *ephemeral memory* is considered to be performed instantly, no sophisticated scheduling is needed: In case those resources are requested by one or more workloads, the resource provider simply evenly divides the available resources among the requesters. In our current implementation, the scheduler follows a round-robin manner to divide the resources. In case a request cannot be fulfilled, an appropriate notification is sent to the *vWorkload* for it to react according to its configuration. For example, a common workload will request as many CPU cycles as possible and a certain amount of memory. As long as the first request does not lead to a critical error, a typical workload can still execute even if there's only a few cycles assigned to it. On the other hand, a workload will most likely crash if a memory requirement cannot be met.

In contrast, access to the 'slow' resources, storage and network, has to be scheduled. We model two operations on storage devices: `write` and `read`. Both operations are performed equally but in the first case, the capacity of the resource provider is decreased by the amount of transferred data whereas in the latter case, the capacity remains constant. When an operation on such a storage resource is initiated, the resource provider will register the operation and evenly divide the available bandwidth among the contenders and process them in parallel. An operation is considered as failed, if the requested amount of data cannot be written or read.

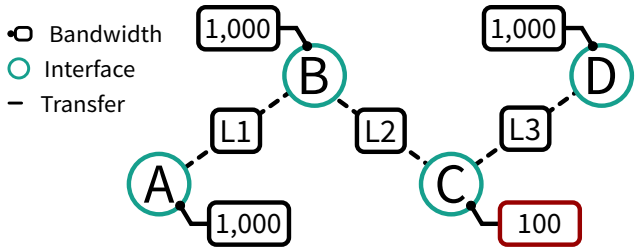


Figure 1. A simple example of a network graph.

B. Network Simulation

One of the major concerns of the simulator described in this paper is the network simulation model. As indicated above a proper calculation of per-transfer speeds can be difficult due to the dependency on the sender's as well as the receiver's bandwidth. The complexity increases with the number of transfers, which, in addition, might or might not share the same destination. A simple example is depicted in Figure 1. The example shows four vNet objects named A, B, C and D associated with the bandwidths $bw_A = 1{,}000$, $bw_B = 1{,}000$, $bw_C = 100$ and $bw_D = 1{,}000$ and three transfers between those ($L_1 = B \rightarrow A$, $L_2 = B \rightarrow C$ and $L_3 = C \rightarrow D$). The data transfers are represented as instances of *vTransfer*. It is obvious that the bandwidth of B has to be shared among two transfers L_1 and L_2 while being capped by the available bandwidth of C after taking L_3 into account. The arising challenge is to balance the transfer speeds in a realistic manner in real-time while considering that the network configuration might significantly change between time steps due to added or removed nodes or transfers. Due to those constraints known algorithms like those based on the *max-flow/min-cut*-theorem [14] are not suitable for our environment. Instead we developed an agent-based algorithm that determines the available bandwidth per active transfer at the current simulation time solely based on the information provided by the receivers of the transfered data.

The algorithm we propose uses an iterative approach that converges to a solution where an optimal data flow between all vNet objects is achieved. This algorithm is further based on a graph whose vertexes $M = \{M_1, \ldots, M_n\}$ are representing data sending or data receiving network interfaces (vNet objects) and whose set of edges $L = \{L_1, \ldots, L_n\}$ are representing network transfers (vTransfer objects). Each vertex M_i is associated with a limited transfer rate $bw(M_i) = bw_i > 0$ expressing the bandwidth of the vNet. The set of vTransfers N_i contains all vTransfers originating from M_i and is a subset of L.

To calculate the current transfer speed s_t at time t of a vTransfer object $L_k = L_{i \rightarrow j} = L_{M_i \rightarrow M_j}$ with $L_{i \rightarrow j} \in N_i$ originating from $o(L_k) = M_i$ and targeted to $d(L_k) = M_j$, the algorithm works in an iterative manner realistically saturating the interfaces.

In a first step ($t = 0$), for each vNet instance all available bandwidth is divided proportionally among the active vTransfers based on the theoretically possible transfer speeds to the corresponding remote partners which is assumed to be defined by the speed of the slowest involved vNet. This can

TABLE I. A SAMPLE CALCULATION FOR THE NETWORK GRAPH SHOWN IN FIGURE 1.

		L_1	L_2	L_3
1st Iteration	A	1000	–	–
	B	909	91	–
	C	–	50	50
	D	–	–	1000
	min()	909	50	50
2nd Iteration	A	1000	–	–
	B	948	52	–
	C	–	50	50
	D	–	–	1000
	min()	948	50	50
3rd Iteration	A	1000	–	–
	B	950	50	–
	C	–	50	50
	D	–	–	1000
	min()	950	50	50

be expressed in the following equation:

$$s'_0(L_{i\to j}) = s'_{t=0}(L_{i\to j}) = \frac{bw_j}{\sum_{n\in N_i} bw(d(n))} \times bw_i \quad (1)$$

In most cases, this assumption is just a rough approximation and not exact, but it is a starting point for the algorithm to work with. With further iterations the value will be gradually corrected.

The result of calculation (1) represents the bandwidth a sending vNet entity M_i could provide for a transfer $L_{i\to j}$ to a target M_j at maximum at time $t = 0$. The same value then has to be calculated for the receiving vNet object. The speed used for the transfer is then determined by the smaller of both values:

$$s_0(L_{i\to j}) = \min(s'_0(L_{i\to j}), s'_0(L_{j\to i})) \quad (2)$$

For the next steps ($t > 0$) the speed of a vTransfer instance will no longer be based on the predefined bandwidth of the involved vNets but on the previously calculated transfer speeds. The equations to determine the speed s of a transfer $L_{i\to j}$ at any time $t > 0$ are adjusted to:

$$s'_t(L_{i\to j}) = \underbrace{\frac{s_{t-1}(L_{i\to j})}{\sum_{n\in N_i} s_{t-1}(n))}}_{A} \times bw_i \quad (3)$$

$$s_t(L_{i\to j}) = \min(s'_t(L_{i\to j}), s'_t(L_{j\to i})) \quad (4)$$

With the factor A always being less or equal than 1.0 the maximum bandwidth of source network interface is never exceeded. Further A can only be equal to 1.0 in the case of a single active transfer, which results in that transfer using the maximum available capacity.

Using this algorithm we are able to determine the speed of a transfer at any time simply by factoring in either previously calculated transfer speeds in case of $t > 0$ or the neighboring machines theoretically possible transfer speeds in case of $t = 0$. A sample calculation for the example discussed above is shown in Table I. The first fours rows of the tables contain the values for s'_t for the transfer corresponding transfer expressed by the columns. The last row of each table contains the value of s_t. One can see that over time, the speeds of the transfers converge to a final value that overall saturates the vNets A to D.

TABLE II. PROPERTIES OF A WORKLOAD OBJECT.

Resource type	data type	description
vCPU	integer	Value expressing the total amount of calculation power the workload will use expressed in *cpu cycles*.
vMemory	f(cpu cycles)	Function returning the amount of used vMemory when a given amount of cpu cycles has be processed or is remaining.
vStorage, vNetwork	list	List of triplets expressing the amount of *cpu cycles* to be remaining or processed for a defined operation to be executed and a flag allowing parallel execution.

Finally, when using the model above, the calculated transmission speeds will always be based upon ideal network connections transmitted over ideal network cables without any kind of noise. Thus, we want to point out, that in our implementation we add in a factor called 'scatter' to express some randomness in latency and bandwidth and represent a combination of all kind of complex signal fluctuations. Our test show, that a variation of a few percent in transfer speeds is realistic due to physical connections not performing perfectly at all times.

IV. WORKLOAD DESCRIPTION

In our model, each workload is associated with a certain amount of load related to the resources described in Subsection III-A. We do not consider workloads not associated with any resource consumption, so the minimal defined workload will be a workload only consuming a single cycle of calculation power. This enables the use of *calculation power* as a reference for other resource usages: Instead of modeling usage of vMemory based on the time passed since the launch of the workload we model memory conumption as a function of consumed *calculation power*. In general, the definition of memory as a function over cpu cycles instead of processing time is a more realistic approach: a started but not advancing (i.e., due to missing resources) workload will not change its state and thus not change its consumption of memory.

Since the usage of storage and network are time-consuming processes due to the non-instant character of the underlying resource providers we cannot transfer this mechanism par for par. As a solution we designed storage and network usage as triggers likewise based on the amount of cpu cycles. Those triggers are simple integers and are executed as soon as the number of consumed or remaining cpu cycles has reached the configured value. Upon execution the trigger can then instantiate new network transfers or storage operation which are then processed. In addition the user can specify if those operations should be processed in parallel or if the workload should pause meanwhile.

Beside the described resource-related properties which are summarized in Table II each workload may execute user-defined subroutines upon reaching certain states to control the flow of workloads. Predefined states are *init* which gets executed when the workload is launched on a vMachine instance, *finish* which is associated with the workload terminating and

 ISBN: 978-1-61208-460-2

error, the state that a workload changes to upon failing. In addition a user can add other trigger and subroutines.

To start the simulation of a workload on a given infrastructure, the workload object has to be registered at a vMachine instance. The vMachine then enables access to its resources by reading the workload definition at the current simulation time and passing appropriate requests to the resource providers which then schedule the incoming requests.

V. IMPLEMENTATION

We realized the described model using Python 2.7 based on the concepts of the object-oriented programming paradigm. The most important components to mention are the different resource providers vNet, vMemory, vStorage and vNetwork along with vTransfer, the higher level entities vMachine and vDatacentre and finally the global environment object called *env*. Whereas most of the implementation can be considered straight forward, we want to highlight some aspects we consider implementation specific.

First, we implemented the resource provider's scheduling in case of more than one workload accessing the resource at a time is done in a round-robin manner. Hereby, the resource provider allocates an equal amount of the available resource to each consumer. In the future, we plan to implement other scheduling mechanisms.

Python natively does not offer an event handling concept, so we based our simulator on generators. A generator function allows to define a function that behaves like a iterator. The details of this concept are described in [15] (PEP0289 and PEP0342). This enables the implementation of time-dependent actions, like network transfers and storage operations as loops. An action requiring more than one time step to be performed is expressed as a set of smaller parts that are one time step long and yield at the end of their execution. We believe that using generators leads to less error-prone code due to the simplicity and the linear, synchronous layout.

Finally, due to the simulator not running in parallel but calling subtasks subsequently one at a time, one has to make sure that one task doesn't change the simulation state prematurely. We solved this by deep-copying the simulator's state before the execution of the subtask begins. Then, all subtasks refer to the copy when reading values while making the changes to the original state.

VI. CALIBRATION AND VALIDATION

To use the simulation framework, users must specify the available infrastructure by defining resource providers as explained in Section III-A. The model can be calibrated based on either of two correlated degrees of freedom: either the resource provider's performance or the workload's resource usage can be tuned. As soon as one dimension is set, the other has to be picked accordingly to get consistent results.

With the model being a simplification of reality it is up to the user to configure the infrastructure in a suitable way for one's use-case. Any calibration settings are workload specific, since for example a workload might or might not benefit from certain CPU features (i.e., Streaming SIMD Extensions (SSE) [16]) or network related characteristics (e.g., delays due to encryption settings).

TABLE III. AVERAGE PERFORMANCE OF COMMON CPUS REGARDING PSEUDO-RANDOM NUMBER GENERATION.

CPU name	clock frequency	Performance [MB/s]
Intel i3-2100	3.1 GHz	$16.3 \pm 0.9\%$
Intel i5-4288u	2.6 GHz	$13.5 \pm 1.4\%$
Intel Xeon E5520	2.27 GHz	$7.2 \pm 1.8\%$
Intel Xeon E5-2650	2.0 GHz	$3.6 \pm 8.5\%$

In addition to this systematic calibration, performance variations on all sorts of resources might occur due to non-perfect behavior of hardware. Those variations are comparably small and are covered by the aforementioned scatter factors we added to the model in our implementation. Those aspects are usually only known to system or software engineers or can be obtained by benchmarking.

We provide consistent calibration values based on the following assumptions for an Amazon EC2 environment:

- A workload that is modeled for a resource provider that supports a certain CPU feature may need less CPU cycles and is modeled accordingly.
- The same application that is modeled for a resource provider that does not support a certain CPU feature may need more CPU cycles and is modeled accordingly.
- The simulated execution of the same application using resource providers that are n-times more powerful than the original resource providers for which the application had been tuned in the beginning should take in the order of a n-th of the time when compared to the initial simulation.
- The simulated execution of an application should yield approximately the same results when compared to real run-time, respectively should be faster or slower in the same order of magnitude relative to the application the model's parameters were derived from.

To derive calibration values, we ran a high number (approx. 560,000) of tests on different machine configurations using a distributed benchmark suite. A detailed description of the tests can be found in [13]. To obtain the calibration values, we correlated the resources of the simulation environment with the performance of the corresponding real hardware.

A. Calibration

In a first step, we calibrated calculation power based on data of a synthetic CPU benchmark. In our case this is the single-threaded calculation of pseudo-random numbers for checksumming purposes on virtual machines of the EC2 instance type "t2.small" in the region "us-east-1". For our calibration, we set $2{,}000\,\mathrm{cpu\ cycles} \mathrel{\hat{=}} 2{,}000\,\mathrm{MHz}$, which is the average clock frequency a virtual CPU on the used EC2 instances provides as stated by Amazon. The benchmarks showed an average random number generation rate of $3.6 \pm 8.5\%$ megabytes per second per core using an Intel Xeon E5-2650 running at 2 GHz. Table III shows the results of the same benchmark on other CPUs for reference. Using this data, we can state that a real-world workload using the same amount of computing power as required for the generation of $3{,}600{,}000$ pseudo-random numbers would run one second. As mentioned above, this value might seem to be abstract compared to e.g.,

TABLE IV. AVERAGE PERFORMANCE OF COMMON CPUS REGARDING PSEUDO-RANDOM NUMBER GENERATION.

Evaluated transfer	distance	std. deviation
us-east-1 ↔ us-west-1	3,600 km	13.4%
us-west-1 ↔ eu-west-1	8,000 km	14.5%
ap-southeast-1 ↔ eu-west-1	11,300 km	17.3%
ap-southeast-1 ↔ us-west-1	13,900 km	19.7%

GFLOPS, but since a user might not necessarily know the amount of FLOPS his specific workload conducts as well as he might not know what CPU features are beneficial to his program, the description based on the benchmark results of his actual workload is more usable.

In contrast to the calibration of calculation power, storage and memory are less complex. The only free parameter associated with a vStorage instance is the bandwidth per time step. Nearly 50,000 performance tests on the aforementioned Amazon EC2 instances resulted in an average of $40 \pm 2.5\%$ MB/s. For vMemory, the only free parameter is the capacity, which is expressed as an integer. Other parameters, like e.g., access latency are not covered by the model.

The last parameters to be calibrated are those related to the network simulation. The bandwidth of a network device is primarily defined by its specifications and can usually be set straight forward. However, due to our calibration environment being virtualized, we had to measure the actual network capabilities which resulted in a bandwidth of 350 Mbit/sec±18%. Secondarily, the model considers the distance between sender and receiver as an influencing factor. A network transfer will be slower proportionally to the distance between two communicating machines. This factor can be significant and results in variations of transfer speeds of up to 13% (for transfers between us-east-1 situated in North Virginia, USA and us-west-1 in California, USA) to 20% (for transfers between us-west-1 and ap-southeast-1 in Singapore). A more detailed listing of the influence of distance between transfer endpoints on the fluctuation range of transfer speeds can be found in Table IV. We assume a fluctuation of 2% per kilometer on the first 4,000 km and 1% per kilometer above. This is in good accordance with the measured results and, in rough approximation, expresses that connections on the first 4,000 km are most likely connections on mainland with delays caused by e.g., routing devices. Longer distances can be assumed to be oversea connections that are mostly free of interference. The final parameter the model includes factors in the quality of the involved hardware components. Network transfers between the same machines on the same cable still fluctuate in a non-negligible, statistical manner following a gaussian curve. In our data we measured a standard deviation of 5%.

With the values deduced from our benchmarks and presented in this section, we want to validate the simulation environment's functionality by predicting the behavior of a real-world scenario using the simulator and compare the results.

B. Validation

The validation of the system's behavior regarding pure CPU load was done by simulating the calculation of 1,000 MB of pseudo-random numbers as explained in the calibration section. Based on the observations above, the expected runtime per job is 288 s. This expectation is covered by the simulation however, the observed values on real machines showed strong fluctuations of up to 31% and a standard deviation of 7.9 %. A detailed description can be found in [13].

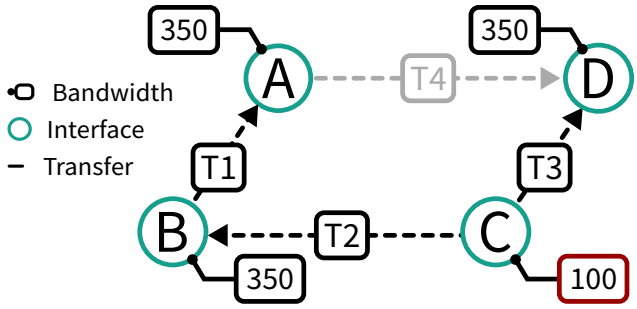


Figure 2. The graph representing the network validation environment.

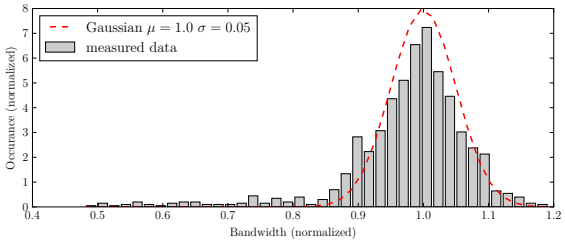


Figure 3. Normalized histogram of the measured bandwidths.

Next, we validated the behavior of the vStorage implementation. Since the system is based on a slow resource provider equal to the one implemented to represent network devices the validation of vStorage is shortened. The simulated workload is a single threaded application writing 10,000 MB to disk. According to the calibration, this process should take 250 s. We were able to reproduce this value in our simulation environment with a average deviation equal to the calibrated 2.5 %. This result was then compared to the performance values measured by `bonnie++`. We were able to show good accordance between the test cases although the aforementioned effect of varying performance biased the results. In numbers, the deviation between simulated and observed results for the duration of the benchmark was averaging at 5.3 % with peaks of 13.1 %.

Exemplary we choose a network setup similar to the one described in Section III-B and depicted in Figure 1 for validation of the network model. However, we slightly modified the scenario to match the environment we used for calibration (Amazon EC2, region *us-east-1*, "t2.small" instances). To be more conclusive, in addition to a static network layout, we assume another transfer between machine A and D to appear after five seconds. The new scenario is depicted in Figure 2 and reads as follows:

> At time $t = 0$, machine A initiates the transfer of 500 MB of data to machine B ($T1$). Meanwhile machine C sends 500 MB of data to B ($T2$) and 500 MB of data to D ($T3$). A, B, D hold 350 Mbit/s network devices, C is associated with a 100 Mbit/s device. At $t = 5$, A initiates an additional transfer of 500 MB of data to D ($T4$).

We executed the scenario 100 times in reality and using

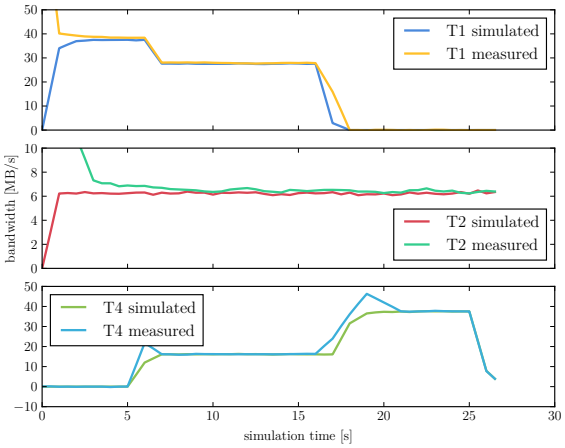


Figure 4. Comparison of measured and simulated bandwidths over time.

the simulator. In our first configuration, all machines are situated within the same data center and thus the geographical correction factor is zero and we only expect fluctuations of five percent due to our calibration. After dropping obvious outliers, the results confirm the assumption of a gaussian distributed micro-scattering as visualized in Figure 3.

The plot depicted in Figure 4 shows the transfer speeds for each transfer at a given time averaged over all performed benchmarks. Altogether we observed accurate accordance between simulation and reality as shown by the convergence of the transfer speeds to the expected values. However, in the first seconds, a large discrepancy can be seen. Since the values are far above the specifications of the network device, we assume the filling of caches or buffers being the cause. The transfers $T2$ and $T3$ evidently converge to $6.25\,\mathrm{MB/s}$, the theoretical maximum of the sending network card. $T1$ stays slightly below the theoretical maximum sending speed of A. This is expected since the overall bandwidth has to be shared with $T4$ as soon as $t \geq 5$.

In a second series of tests, we moved machine C to eu-west-1, $3{,}600\,\mathrm{km}$ away from the other machines. This leads to the geographical correction factor being relevant and thus to be factored in during calculation. Again, the whole scenario was done 100 times. This time we observed a bigger discrepancy and larger fluctuations. As mentioned in [13], factors like time of day respectively day of week do affect the performance of the virtual environments and their attached network devices. Since the deviation of the simulated results from those measured in reality is not larger than $7\,\%$ at any time $t_{\text{transfer}} > 3s$ during our tests, we consider the simplification as acceptable.

VII. Discussion, Conclusion and Outlook

As the tests show, our simulation framework provides good results after calibration has been done. The avarage deviation of the simulation results compared to real-world findings is lower than 7.9 percent for CPU simulation, lower than 5.3 percent for storage simulation and lower than 7 percent for network simulation.

The network simulation converges almost to the same mean transfer speed, only during the first few seconds the simulation underestimates the transfer speed that can be achived in the real-world. Our simulation never yields results for transfer speeds that are above the actual link speed - in contrast to the real world due to cache effects in the first few seconds. After around three seconds the simulated transfer rate is close to the real-world transfer rate. As the focus of our simulation framework is not to deliver a precise estimation of transfer speeds at all times, but to provide good overall results, the mentioned deviations are negligible in our case.

Our CPU simulations yielded very similar average results when compared to real-word tests. However, there are some deviations that can be explained by varying performance of the cloud infrastructure. As we found in [13], the performance of cloud infrastructure is not constant over time. Neglecting those variations, our results were quite good.

Our simulation framework allows to model infrastructure and workloads in a flexible way. There are two degres of freedom that allow to set one of them conviniently while choosing the other accordingly. The simulation framework yields consistent results and converges quickly. As it is written in Python, it is very lightweight and extensible with little effort. However, during testing we noted that some facts that are particularly important when simulating cloud infrastructure cannot yet be captured by our framework. Most importantly: cloud resources do not always yield the same performance. Without knowing the exact reasons, we strongly suspect that varying load of the providers' infrastructure as well as varying load of the inter-connecting networks.

Therefore, the next iteration of our simulation framework must support resource providers that yield a time-dependent amount of resources instead of a fixed number of, e.g., CPU cycles. Also, multi core support as well as the ability to capture CPU features are points to adress in future releases. Further we plan to expand the framework in the future to support a user when developing higher level functionalities such as scheduling mechanisms for distributed systems or simulations of platform services. Another aspect that is not yet covered by our framework and that should be tackled in future releases is a price model for infrastructure resources - something that is of particular interest in cloud computing.

References

[1] P. Mell and T. Grance, "The nist definition of cloud computing," National Institute of Standards and Technology, vol. 53, no. 6, 2009, p. 50.

[2] S. NIST, "800-145,"," A NIST definition of cloud computing",[online] http://csrc. nist. gov/publications/nistpubs/800-145/SP800-145. pdf, 2011, [accessed 1-February-2016].

[3] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V., "Cloud Computing – Ein ganzheitlicher Blick über die Technik hinaus," 2010.

[4] BITKOM Research, "Cloud monitor 2014," 2014.

[5] S. hwan Lim, B. Sharma, G. Nam, E. K. Kim, and C. R. Das, "Mdcsim: A multi-tier data center simulation, platform," in in Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on, 2009.

[6] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41, no. 1, 2011, pp. 23–50.

 ISBN: 978-1-61208-460-2

[7] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities," Jul. 2009, [accessed 1-February-2016].

[8] S. Garg and R. Buyya, "Networkcloudsim: Modelling parallel applications in cloud simulations," in Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on, Dec 2011, pp. 105–113.

[9] X. Li, X. Jiang, K. Ye, and P. Huang, "Dartcsim+: Enhanced cloudsim with the power and network models integrated," in Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on, June 2013, pp. 644–651.

[10] B. Wickremasinghe, R. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, April 2010, pp. 446–452.

[11] A. Núñez and et al., "icancloud: A flexible and scalable cloud infrastructure simulator," J. Grid Comput., vol. 10, no. 1, Mar. 2012, pp. 185–209.

[12] G. Castane, A. Nunez, and J. Carretero, "icancloud: A brief architecture overview," in Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on, July 2012, pp. 853–854.

[13] P. Krauss, T. Kurze, and A. Streit, "Cloudbench – A Framework for Distributed, Self-organizing, Continuous and Topology-aware IaaS Cloud Benchmarking with Super-peer Networks," in e-Science (e-Science), 2015 IEEE 11th International Conference on, Aug 2015, pp. 273–278.

[14] Wikipedia, "Max-Flow-Min-Cut-Theorem — Wikipedia, the free encyclopedia," 2015, [accessed 1-February-2016]. [Online]. Available: https://de.wikipedia.org/wiki/Max-Flow-Min-Cut-Theorem

[15] D. Goodger and B. Warsaw, "Index of Python Enhancement Proposals," 2016, [accessed 1-February-2016]. [Online]. Available: https://www.python.org/dev/peps/

[16] R. Ramanathan, R. Curry, S. Chennupaty, R. L. Cross, S. Kuo, and M. J. Buxton, "Extending the World's Most Popular Processor Architecture," Intel Corporation, White Paper, 2006.

 ISBN: 978-1-61208-460-2

Overcome Vendor Lock-In by Integrating Already Available Container Technologies Towards Transferability in Cloud Computing for SMEs

Peter-Christian Quint, Nane Kratzke
Lübeck University of Applied Sciences, Center of Excellence CoSA
Lübeck, Germany
email: {peter-christian.quint, nane.kratzke}@fh-luebeck.de

***Abstract*—Container clusters have an inherent complexity. A distributed container application in the cloud can be complex at planning, installation and configuration, maintenance and search for failures. Small and medium enterprises (SMEs) are mostly limited by their personnel and financial restrictions. Using advanced cloud technologies like a container cluster often requires high personnel expenses or the use of an external system builder. In addition to economical, security- and governance issues there is also the concern of technical vendor lock-ins. This paper introduces C^4S, an open source system for SMEs to deploy and operate their container application with features like elasticity, auto-scaling and load balancing. The system also supports transferability features for migrating container between different Infrastructure as a Service (IaaS) platforms. This paper presents a solution for SMEs to use the benefits of cloud computing without the disadvantages of vendor lock-in.**

Keywords–Microservice; Container; Docker; Container Cluster; Software Defined Network; Cloud Computing; SME

I. INTRODUCTION

Infrastructure as a service (IaaS) enables companies to get resources like computational power, storage and network connectivity on demand. IaaS can be obtained on public or private clouds. Public clouds are provided by third parties for general public use. Type representatives are Amazon's Elastic Compute Cloud (EC2) and Google Compute Engine (GCE). Private Cloud are intended for the exclusive use by a single organization [1]. They are mostly installed on the respective companies own infrastructure. OpenStack is a cloud platform for providing (not exclusively) private clouds. One big benefit using cloud computing is the elastic scaling. Elasticity means the possibility to match available resources with the current demands as closely as possible [2]. Scalability is the ability of the system to accommodate larger loads by adding resources or accommodate weakening loads by removing resources [3]. With autoscaling, resources can be added automatically when they are needed and removed when they are not in use [4]. The resources are allocated on demand and the customer only has to pay for resources he really used. The system described in this paper will support several, public and private, cloud environments. Features like elastic scaling and transferability will also be available. The authors define transferability as the possibility to migrate some or all containers between different cloud platforms. This is needed to avoid vendor lock-in by the cloud providers, which is a major obstacle for small and medium enterprises (SMEs) in cloud computing [5]. Only a few research projects deal with the specific needs of SMEs in cloud computing [6].

In the last few years, container technologies like Docker got more and more common. Docker is an open source and lightweight virtualization solution to provide an application deployment without having the overhead of virtual machines [7]. With Docker, applications can be easily deployed on several machine types. This makes it possible to launch containers from the same application (image), e.g., on a personal computer or a datacenter server.

Container clusters like Kubernetes (arose from Google Borg [8]) and Mesos [9] can deploy a huge number of containers on private and public clouds. A big benefit of cluster technologies is the horizontal scalability of the containers, the fast development and the contained software defined network, which is often necessary for distributed container applications. Container and container cluster software are mostly open source and free to use.

SMEs are mostly financially and personnel-wise restricted (see the European definition of SME [10]). Since the management of container cluster applications with features like transferability and elasticity is complex, it can be very difficult to achieve for a small (maybe only one person size) IT department. Getting started using services like Infrastructure as a Service (IaaS) might be very simple. But the use of advanced cloud technologies like clusters, containers and cloud benefits like auto-scaling and load balancing can quickly grow into complex technical solutions. The cloud provider supplied services (i.e., auto-scaling) might pose another issue due to often having non-standardized service APIs. This is often resulting in inherent vendor lock-in [11]. However, there are products and services to manage these technologies like the T-Systems Cloud Broker and Amazon EC2 Container Service (ECS). These management solutions also have disadvantages. For example, the Cloud Broker is a commercial product, which is inherently designed for very big companies. This kind of cloud broker services move the dependencies from the cloud provider to the system/service provider like T-Systems. ECS works only with Amazon EC2-instances, which means there is still a vendor lock-in. Both solutions just shift vendor lock-in to another company. Creating an open source system for easy deployment and managing of cloud applications in a container cluster would support SMEs using these technologies without worries about vendor lock-in.

The software presented in this paper is called C^4S. The name is an acronym for Container Cluster in Cloud Computing System. C^4S is designed to (automatically) deploy an operating container cluster application without vendor lock-in. Moreover, the system will be able to monitor the cloud platform, the container cluster and the containers themselves. Beside bare reporting, the system will offer methods to keep the application running in most failure states. Altogether, the C^4S can make container cluster cloud computing technologies usable for SMEs without large and highly specialized IT departments.

 ISBN: 978-1-61208-460-2

C*⁴*S is not exclusively designed for SMEs. The user group of C4S is not limited to specific company types, so (e.g., big international) companies with small and specific in-house-departments can use it, too.

Section II describes the features and the requirements of the ***C*⁴*S*** system. An overview about the ***C*⁴*S*** architecture is given in Section III. The intended validation of concept, which is performed at the final phase, is presented in Section IV. Related work, several business services and products are described in Section V. Finally, the conclusion is presented in SectionVI.

II. GENERAL REQUIREMENTS OF *C*⁴*S*

C*⁴*S is designed to handle the high complexity of a container cluster with benefits like elasticity, auto-scaling and transferability. Feature requirements and the technical specifications are explained below. By designing and developing a generic cloud service description language, a solution to define secure, transferable and elastic services of typical complexity will be provided. Thus, they are deployable to any IaaS cloud infrastructure. This work promotes the implementation of easy-to-handle, elastic and transferable cloud applications.

A. Deploying and Controlling Applications

The basic feature of ***C*⁴*S*** is to deploy a distributed container application on cloud environments. Therefore, the user can easily configure the needed containers, the interfaces and the cloud environments. According to the user set configuration, the system will automatically deploy the application on the container cluster and the cloud platforms. Overall, there are three controlling levels the management solution ***C*⁴*S*** has to support:

The **Container Application** can be configured, launched, controlled, changed and stopped. Application parts (e.g., container types) must be replaceable (e.g., changing of container images to keep the application up to date). Details like the status and the configuration of every single container and an overview of all running containers should also be visible for the customers.

The **Container Cluster** is used for automatic deployment of the container on the available virtual machines (running on IaaS platforms). Therefore, the cluster solution can create, terminate and transfer containers. The user is able to set values like the deployment limitations and restrictions in the container host selection.

Virtual Machines on IaaS platforms form the host system of the container cluster with the containers. The system should support a management solution for monitoring the status as well as creating and terminating virtual machines on several cloud platforms. The user is able to set values like the virtual machine limitations and the favored platform for each container type.

A view with all used machines, their type, running time, and other data is also necessary to observe, e.g., economic information like actual costs. The system has to be able to monitor on all controlling levels. In case of failure, the monitoring system should trigger reports and automatic actions to keep the application running.

B. Usage of Cloud Features

As described in Section I, cloud computing enables features like **elasticity**, **scalability** and **load balancing**. ***C*⁴*S*** enables the user to handle the inherent complexity of these features in an easy way. **Auto-Scaling** containers on the cluster and virtual machines on IaaS platforms is also featured.

C. Prevent Dependencies

To avoid **vendor lock-in by the cloud provider**, the system can install a multi-cloud container cluster with transferability features. On demand, some or all containers can migrate from one cloud provider to another. Accordingly, the user has full control over where the containers are running.

To **prevent dependencies by used software and services**, the ***C*⁴*S*** will be published under MIT license. It is recommended that all third-party parts like the cluster software are also open source. Thus, the consuming companies avoid dependencies to the C⁴S system and adapt the source code to their special needs. The system has to be designed generic for several cloud platforms. A modular architecture enables extensions for other platforms. Beside the cloud platforms, the users should not be limited by the choice of the container cluster. The modular architecture enables later extensions for missing cluster connectivity.

III. *C*⁴*S* ARCHITECTURE

The architecture is divided into four layers. The core of ***C*⁴*S*** is the deployment and the monitoring engine. The user can manage the deployment and get the monitoring events over two interfaces. The other two parts are the container cluster and the IaaS environments.

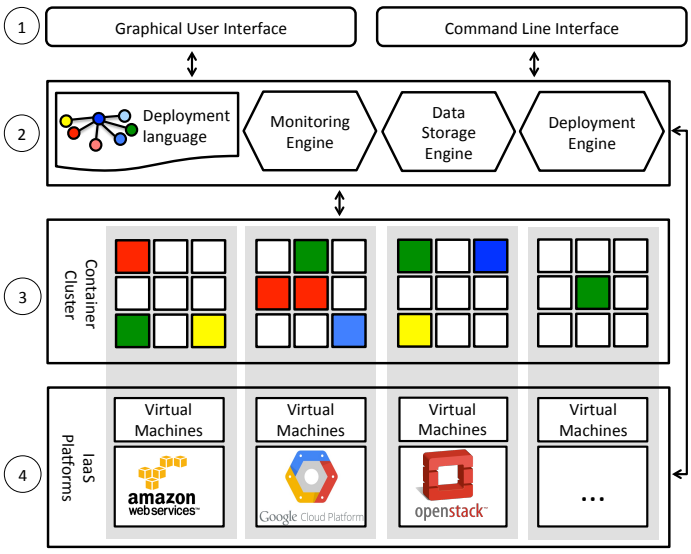


Figure 1. ***C*⁴*S*** architecture overview

A. Interfaces ①

The management system will provide a web-based graphical user interface (GUI) and a command line interface (CLI), see Figure 1, ①. Here the user can set the the account data and limits of the IaaS platforms, the configuration for scaling and also set transfer orders (e.g., moving containers to another

cloud platform). It is also possible to set rules for creating, terminating and moving containers on a special event. By setting these rules, the system can deal with failures like a cloud platform interrupt. Other features like the automatic container cluster software installation can be started using the interfaces. Status information about the containers, the cluster and the cloud platform are visible, monitoring events and the triggered actions are also reported.

B. Deployment and Monitoring Engines ②

This subsection describes the engines shown at Figure 1, ②. The deployment engine is responsible for creating and deleting instances on the cloud platforms, managing the container cluster and deployment of the containers on it, making it the main core of $\boldsymbol{C}^4\boldsymbol{S}$. Features like load balancing and container transfer are also controlled by this engine. The deployment language is designed for stating the constraints of the configuration. All needed informations about the container application, the cluster and the IaaS platforms can be described by the deployment language. The monitoring engine observes the containers on the cluster and virtual machines on the IaaS platform. Additional to reporting states and failures, actions can be triggered by events. For example, if the engine registers an exceptionally high workload of a container, it reports to the deployment engine to scale out containers, and vice versa. The engine can deploy the application on different cloud platforms and is able to transfer containers between them. The data storage engine is compatible with several block and object storage systems to avoid vendor lock-in. The engine also enables scalability and security features for data storage.

C. Container Cluster ③

The container cluster deploys the containers on the virtual cloud machines (see Figure 1, ③). The management system can terminate and create the containers in the cluster network. The system can also transfer a container from one virtual machine to another, which is not even necessarily running on the same cloud platform. These actions are controlled by the deployment engine and supervised by the monitoring engine. In combination with the container cluster, the engines make it possible to migrate services from one private or public cloud infrastructure to another (not necessarily compatible) cloud infrastructure.

D. IaaS Platforms ④

The $\boldsymbol{C}^4\boldsymbol{S}$ can manage several cloud platforms (see Figure 1, ④). It is possible to create and terminate IaaS instances accordingly on demand. Hence, it has to communicate with the cloud platforms. Because of missing standardization [12] for every provider API a special driver has to be designed. These require a modular and easy to adapt software architecture. The deployment language is designed for informations like access data, limits and all other settings which can be set over the described interfaces.

IV. INTENDED VALIDATION OF CONCEPT

It will be shown that SMEs can manage a container cluster over (multi) cloud platforms. At first it will be demonstrated that building a system, which fits all the required features, is possible. Therefore, a working, open source $\boldsymbol{C}^4\boldsymbol{S}$ prototype, which conforms the requirements set in Section II, will be developed. The system has to be implemented in a modular and extendable way. As cluster platform, $\boldsymbol{C}^4\boldsymbol{S}$ will support Kubernetes first, other cluster environments will follow. Presenting interchangeability and the open source type of $\boldsymbol{C}^4\boldsymbol{S}$ will show that dependencies by the used software can be prevented. To avoid vendor lock-in by the cloud provider, the prototype must be able to install a multi-cloud container cluster. First, the system will be compatible with the IaaS cloud platform type representatives Amazon EC2, Google GCE and OpenStack. To support other platforms, appropriate drivers can be implemented. Transferability features like moving all containers from one cloud platform to another will be implemented. Terminating all containers and virtual machines on one provider and creating them on another at the same time, without changes in features like elasticity and auto-scaling, will proof that $\boldsymbol{C}^4\boldsymbol{S}$ is preventing vendor lock-in. The software will also manage container application deployment. It will deploy a container cluster, create and terminate containers and is usable for deploying applications. Also, workloads will be created to test the autoscaling features. With enforced failure states, the robustness of the system will be demonstrated. It will be shown that the system is able to keep the applications running even when containers and virtual machines get disconnected. In the second part of the proof of concept, a company will employ the software. Thus, the expense for a small business using the container cluster manager will be evaluated. Finally, a proof of concept will be realized by several business companies. These companies will use the $\boldsymbol{C}^4\boldsymbol{S}$ system on their own for testing a productive application deployment with real workload. Load balancing, elasticity, auto-scaling and transferability features will be applied in production. This way it will be shown that SMEs can handle the complexity of a container cluster application running on multiple cloud platforms without vendor lock-in or dispensing with features like auto-scaling.

V. RELATED WORK

There are several solutions with overlapping features and/or usage scenarios available. However, a system which fits all requirements and features set in Section II for the $\boldsymbol{C}^4\boldsymbol{S}$ deployment manager does not exist.

A. Container Cluster, Load Balancing and Scaling

A Container Cluster should run on homogeneous machine types to provide fine-grained resource allocation capabilities [9]. In previous work, the similarity of different cloud provider instance types was analyzed. It was concluded that only a few instance pairs are really similar. There are just a few virtual machine types, which should be used when running an application on a multi cloud platform environment [13]. Another issue to consider is the network performance impacts using technologies like container and software defined networks (SDN). Previous investigations found that performance impacts depends i.e., on the used machine types and the message sizes [14]. Using a (encrypted) cross-provider SDN also causes performance impacts, especially when using low-core machines [15].

B. IaaS Management and Transferability

Container migration from one cloud provider to another is an important feature of $\boldsymbol{C}^4\boldsymbol{S}$. Vendor lock-in is caused, i.a., by a lack of standards [12]. Currently the proprietary EC2 is the de-facto standard specification for managing cloud infrastructure.

 ISBN: 978-1-61208-460-2

However, open standards like OCCI and CIMI are important to reduce vendor lock-in situations [16]. ***C***^{4}***S*** includes a special IaaS driver for each supported cloud provider. Other research approaches in cloud migration can be reviewed under [17]. There are several solutions like Apache Libcloud, KOALA [18], Scalr, Apache jclouds and deltacloud and T-Systems Cloud Broker for managing and deploying virtual machines on IaaS platforms. Except for the T-Systems Cloud Broker, the solutions are open source but have mostly payable services, reduced functionality or limited virtual machine quantities. These systems support features like creating, stopping and scaling virtual machines on IaaS cloud platforms. Some of them like the T-Systems Cloud Broker, Scalr and Apache jclouds are designed for cross-platform IaaS deployment. In contrast to the ***C***^{4}***S*** requirements, the presented cloud managers are limited to IaaS managing and do not offer container deploying services. Some of them do not prevent vendor lock-in by cloud providers or create new dependencies by itself (e.g., T-System Cloud Broker, KOALA is limited to Amazon AWS API compatible services).

C. Application Deployment

Peinl et al. [19] has defined requirements for a container application deploying system. These coincide strongly with the requirements for the ***C***^{4}***S*** system, which have been discussed in Section II-A. He also gives an overview about container cluster managing. For easy deployment a container application with monitoring, scaling and controlling benefits, there exist several commercial solutions like the Amazon EC2 Container Service (ECS), Microsoft Azure Container Service and Giant Swarm. Limited to the providers own IaaS infrastructure, these solutions are not designed for multi-cloud platform usages, especially between public clouds (a requirement of ***C***^{4}***S***). Open source cluster managers are Apache Mesos and Kubernetes. These systems are designed to run workloads across tens of thousands of machines. The benefits and issues using cluster technologies are very high reliability, availability and scalability [9] [8]. However, they are not designed to create and terminate virtual machines (like AWS instances), but to deploy applications on given resources. So, they cannot prevent cloud provider dependencies on their own, but provide essential ingredients to do so. Another cluster management tool for increasing the efficiency of datacenter servers is called Quasar, which is developed by the Stanford University and designed for maximizing resource utilization. The system performs coordinated resource allocation. Several techniques analyze performance interferences, scaling (up and out) and resource heterogeneity [20].

VI. CONCLUSION

The ***C***^{4}***S*** is in the planning state, although some parts are already implemented, like the cloud platform driver for fast deploying IaaS instances. The next step is the creation of a deployment language for dedicated containers to run on a Kubernetes container cluster, finding solutions for container cluster scaling problems and handling stateful tasks like file storage. The system will be implemented in a modular and generic way to allow an easy adaptation for different cloud platforms and container cluster software. With ***C***^{4}***S*** SMEs will be able to deploy and operate their container applications on an elastic, auto-scaling and load balancing multi-cloud cluster with transferability features to prevent vendor lock-in.

ACKNOWLEDGMENT

This research is funded by German Federal Ministry of Education and Research (Project Cloud TRANSIT, 03FH021PX4). The author thank the University of Lübeck (Institute of Telematics) and fat IT solution GmbH (Kiel) for their support of Cloud TRANSIT.

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing," 2011.

[2] M. Armbrust et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, 2010, pp. 50–58.

[3] L. M. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," ACM SIGCOMM Computer Communication Review, vol. 41, no. 1, 2011, pp. 45–52.

[4] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2011, p. 49.

[5] N. Kratzke, "Lightweight virtualization cluster - howto overcome cloud vendor lock-in," Journal of Computer and Communication (JCC), vol. 2, no. 12, oct 2014.

[6] R. Sahandi, A. Alkhalil, and J. Opara-Martins, "Cloud computing from smes perspective: A survey-based investigation," Journal of Information Technology Management, vol. 24, no. 1, 2013, pp. 1–12.

[7] J. Turnbull, The Docker Book: Containerization is the new virtualization. James Turnbull, 2014.

[8] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in Proceedings of the Tenth European Conference on Computer Systems. ACM, 2015, p. 18.

[9] B. Hindman et al., "Mesos: A platform for fine-grained resource sharing in the data center." in NSDI, vol. 11, 2011, pp. 22–22.

[10] Definition recommendation of micro, small and medium-sized enterprises by the european communities. Last access 12th Nov. 2015. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2003.124.01.0036.01.ENG

[11] N. Kratzke, "A lightweight virtualization cluster reference architecture derived from open source paas platforms," Open J. Mob. Comput. Cloud Comput, vol. 1, 2014, pp. 17–30.

[12] J. Opara-Martins, R. Sahandi, and F. Tian, "Critical review of vendor lock-in and its impact on adoption of cloud computing," in Information Society (i-Society), 2014 International Conference on. IEEE, 2014, pp. 92–97.

[13] N. Kratzke and P.-C. Quint, "How to operate container clusters more efficiently? some insights concerning containers, software-defined-networks, and their sometimes counterintuitive impact on network performance," International Journal on Advances in Networks and Services, vol. 8, no. 3&4, 2015, (in press).

[14] N. Kratzke and P.-C. Quint, "About automatic benchmarking of iaas cloud service providers for a world of container clusters," Journal of Cloud Computing Research, vol. 1, no. 1, 2015, pp. 16–34.

[15] N. Kratzke, "About microservices, containers and their underestimated impact on network performance," CLOUD COMPUTING 2015, 2015, p. 180.

[16] C. Pahl, L. Zhang, and F. Fowley, "Interoperability standards for cloud architecture," in 3rd International Conference on Cloud Computing and Services Science, CLOSER. Dublin City University, 2013, pp. 8–10.

[17] P. Jamshidi, A. Ahmad, and C. Pahl, "Cloud migration research: a systematic review," Cloud Computing, IEEE Transactions on, vol. 1, no. 2, 2013, pp. 142–157.

[18] C. Baun, M. Kunze, and V. Mauch, "The koala cloud manager: Cloud service management the easy way," in Cloud Computing (CLOUD), 2011 IEEE International Conference on. IEEE, 2011, pp. 744–745.

[19] R. Peinl and F. Holzschuher, "The docker ecosystem needs consolidation." in CLOSER, 2015, pp. 535–542.

[20] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and qos-aware cluster management," ACM SIGPLAN Notices, vol. 49, no. 4, 2014, pp. 127–144.

 ISBN: 978-1-61208-460-2

Cloud Data Denormalization of Anonymous Transactions

Aspen Olmsted, Gayathri Santhanakrishnan
Department of Computer Science
College of Charleston, Charleston, SC 29401
e-mail: olmsteda@cofc.edu, santhanakrishnang@g.cofc.edu

***Abstract*— In this paper, we investigate the problem of representing transaction data in PAAS cloud-based systems. We compare traditional database normalization techniques with our denormalized approach. In this research, we specifically focus on transactional data that is not attached to a specific customer. The absence of the relationship in the customer object allows for the vertical merging of tuples. to be stored in aggregate form. The journaling features of the data store, allow full audits of transactions while not requiring anonymous data to be materialized in fine-grained levels. The horizontal merging of objects is also deployed to remove detail lookup data instance records and replace them with business rule objects.**

Keywords-web services; distributed database; modeling

I. INTRODUCTION

In this work, we investigate the problem of representing transactional data in a platform as a service (PAAS) cloud-based system. In traditional client-server architectures, database normalization is used to ensure that redundant data is not stored in the system. Redundant data can lead to update anomalies if the developer is not careful to update every instance of a fact when modifying data. Normalization is also performed to ensure unrelated facts are not stored in the same tuples resulting in deleting anomalies.

Data representation in the cloud has many of the same challenges as data representation in client/server architectures. One challenge data representation in the cloud has that is not shared with client/server is the minimization of data because costs of cloud data storage are significantly higher than local storage. We are thinking of simple costs for disk storage, not true costs. Organizations budget only the costs of disk drives for local storage which are in the tens of dollars per gigabyte but cloud storage can be in the hundreds of dollars per gigabyte per month [1]. Often this storage is expressed as the number of tuples in the data store instead of the number of bytes on the disk drive holding the data. For example, force.com [2] charges for blocks of data measured in megabytes but they calculate usage as a flat 2KB per tuple. Zoho Corporation also tracks data storage by the tuple for serval of their cloud products including Creator [3] and CRM [4]. The tuple count method is used as it is easier to calculate in a multi-tenant system where the physical disk drives are shared by many clients.

In this paper, we present an algorithm that will minimize the number of tuples used to store the facts for a software system. We use a motivating example from a cloud software system developed by students in our lab. The algorithm performs three main operations:

- The horizontal merging of objects – several distinct relations are combined into one.
- The vertical merging of objects – several distinct instances of the same type of facts is combined into one.
- Business rule adoption – instead of storing tuples to represent availability of lookup data, we replace the tables with pattern based business rules

We apply our algorithm to a system in the humanities application domain and show an approximately 500% reduction in tuple storage.

Date [5] invests a good deal of his text on the definition of denormalization. He argues that denormalization is when the number of relational variables is reduced, and functional dependencies are introduced where the left hand of the functional dependency no longer is a super key. In our work, we perform many optimizations. When we horizontally merge relations, then we are performing a true denormalization. Other optimizations such as vertical merging do not fit Date's definition of denormalization. We choose to stay with the term denormalization algorithm as it is a set of steps taken after the normalization process.

The organization of the paper is as follows. Section 2 describes the related work and the limitations of current methods. In Section 3 we give a motivating example where our algorithm is useful and describe our denormalization algorithm. Section 4 describes additional enhancements through the design of business rule objects. Section 5 explores reporting from the denormalized objects utilizing the object version history stored in the journal. Section 6 contains our comparison of the proposed method and the traditional database normalization method. We conclude and discuss future work in Section 7.

.

II. RELATED WORK

Sanders and Shin [6] investigate the process to be followed when denormalization is done on relational data management systems (RDBMS) to gain better query performance. Their work was done before the cloud database offerings became prevalent. In the cloud, database performance is less of an issue to storage requirements because the systems are designed to distribute queries across many systems.

Conley and Whitehurst [7] patented the idea of denormalizing databases for performance but hiding the denormalization for the end user. Their work focuses on merging two relations into one relationship to eliminate the processing required to join the records back together. Their work uses horizontal denormalization to gain performance.

 ISBN: 978-1-61208-460-2

Our work uses both horizontal and vertical denormalization to minimize storage space and increase usability.

Most denormalization research work was done in the late 1990s and was focused on improvement in query performance over the correctness and usability of the data. Recently, folks like Andrey Balmin have looked at denormalization as a technique to improve the performance of querying XML data. Like the previously mentioned research, this work differs from our work in the desired end goal of minimized data storage and end user usage.

In Bisdas' [8] blog, he demonstrates ways that end users can improve data visualization by vertically merging hierarchical data in the Salesforce, data model. He takes advantage of the trigger architecture to create redundant data in the hierarchy. Taber [9] also recommends denormalization to improve data visualization. The problem with both solutions is that data storage requirements are increased while correctness is jeopardized by the redundant data.

In our previous work [10], we study UML models from the perspective of integrating heterogeneous software systems. In the work, we create an algorithm to sort cyclical UML class data diagrams to enable transaction reformation in the integration. In the process, discoveries were made on the freshness of data at different layers in the UML graph. The knowledge is useful in this study when considering anomalies that can happen in response to data updates.

Additional semantics for models can be added by the integration of the matching UML Activity and Class diagrams. UML provides an extensibility mechanism that allows a designer to add new semantics to a model. A stereotype is one of three types of extensibility mechanisms in the UML that allows a designer to extend the vocabulary of UML to represent new model elements [11]. Traditionally these semantics were consumed by the software developer manually and translated into the program code in a hard coded fashion.

Developers have implemented business rules in software systems since the first software package was developed. Most research has been around developing expert systems to process large business rule trees efficiently. Charles Forgy [12] developed the Rete Algorithm, which has become the standard algorithm used in business rule engines. Forgy has published several variations on the Rete Algorithm over the past few decades.

III. DENORMALIZATION

We demonstrate our work using a Tour Reservation System (TRS). TRS uses web services to provide a variety of functionalities to the patrons who are visiting a museum or historical organization. We use the UML specification to represent the meta-data. Figure 1 shows a UML class diagram for an implementation of this functionality. The Unified Modeling Language includes a set of graphic notation techniques to create visual models of object-oriented software systems [13]. In this study, we use data collected by the Gettysburg Foundation on visitors to their national battlefield. The system is modeled and implemented on the force.com [2] cloud platform.

Figure 1 shows a normalized UML class model of reservation transactions of visitors to the Gettysburg National Battlefield. In the model the central object ticket represents a pass for an entry that is valid for a specific date and time and a specific activity. Activities are itinerary items the visitor can be involved in while visiting the battlefield. In the normalized model, each ticket is linked to a specific activity schedule entry that will designate the date and time the pass is valid for entry. Each activity schedule is linked to an activity object that designates the name and location of the activity.

Each ticket is linked to a user in the Gettysburg organization who was responsible for the transaction. Each ticket can be linked to a patron object. In the case of advanced reservations, there will be a valid patron object linked to the ticket. Advanced reservations are transactions that take place through the organization's website or over the phone to a reservation agent. In the case of walk-up transactions, there will not be a linked patron. A walk-up transaction is a transaction that takes place when a visitor arrives on the site without a prior reservation and pays for the ticket at the front desk.

In Figure 1, the multiplicity of the association between the patron and the ticket is a zero or one to many. A multiplicity that can be zero represents anonymous data. Anonymous data is data that does not need to be specified in order for the transaction to be valid. In the example transaction, the patron can remain anonymous but still visit the battlefield and partake in the activities. In the case of the sample Gettysburg data, 60 percent of ticketing transactions were anonymous.

In the case of the force.com [2] PAAS, data storage is charged by the tuple. With an enterprise license to the platform, the organization is granted access to one gigabyte of data storage. The storage is measured by treating every tuple as two kilobytes. This form of measurement allows

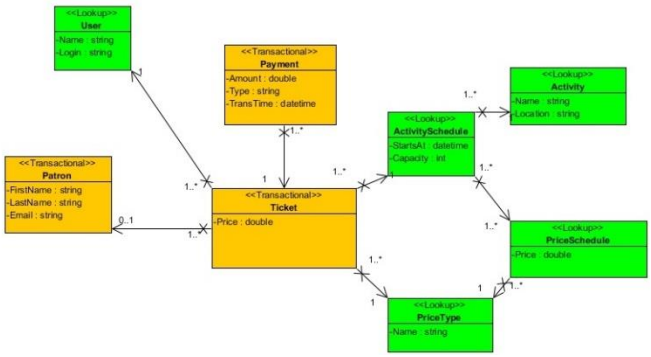


Figure 1. Normalized Transaction Model

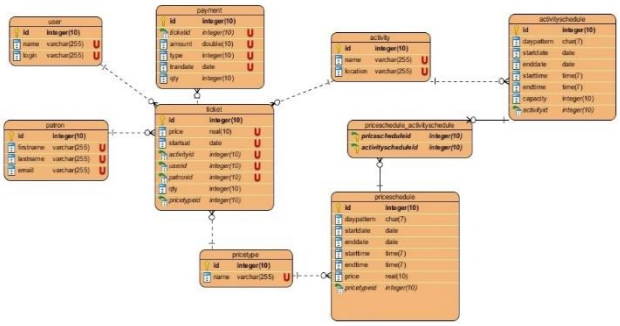


Figure 2 Denormalized Transaction Model

the organization 500,000 tuples in the enterprise data storage option. The data collected for the normalized data model would allow the Gettysburg organization to store around nine months' worth of data. With the anonymous transaction, the ticket and the payment data is only important on the original transaction level for auditing. For example, the accounting department may want to see the details behind a specific ticket agent's cash total for the day. Another example would be the marketing department wants to see the ticket price patterns within an hour of the day.

The force.com [2] platform uses an Oracle relational database to deliver the data storage services but adds a journal feature so history can be stored on all changes to an object over time. This journal can be used at no additional data storage cost. The field level changes stored in the journal would allow aggregate data to be stored for anonymous transactions and still have the detail to perform the audits mentioned earlier.

If an object is used between two other objects where the middle object is the "many" side of the one-to-many relationship and the one side of the other relationship, then the same data can be represented by moving the attributes to the object on the composition side of the relationship. The middle object is then able to be removed, reducing the number of tuples representing the same amount of data. In Figure 1, the "Activity Schedule" object fits this profile and can be horizontally merged with the "Ticket" object. In our previous work [10], we study UML data model freshness requirements and document the relationship between data changes and location in the UML graph. In our findings, we see that middle object nodes are less predisposed to changes than leaf nodes. The lower amount of data changes reduces the change of update anomalies.

In Figure 1, we also designate objects that are updated in transactions differently than objects that are navigated for transactional lookup values. Two stereotypes are added to the diagram:

- Transactional – The classes designated with the orange color and the <<Transactional>> tag are updated during transactional activities.
- Lookup – The classes designated with the green color and the <<Lookup>> tag are not updated during transactional activities. The data in these classes are created by administrative activities. During transactions, the data is searched for the proper values.

The Denormalization Algorithm, Algorithm 1, transforms a normalized model stored in a UML class diagram into a denormalized model represented as an entity-relationship diagram. The algorithm assumes input and output of the models in the XMI [14] format. XMI is a standard exchange format used to represent structural models in a non-proprietary way.

The algorithm first loops through each object in the normalized model and adds the object and attributes as entities in the denormalized entity-relationship diagram. If the object has the transactional stereotype, then the attributes are marked unique. Surrogate identifier fields are added to each object's definition to be used as an auto-incrementing primary key.

The next pass of the algorithm is to find objects that can be eliminated from the middle relationship of two "one-to-many" relationships. The original model, in Figure 1, had an activity schedule object that consumed a lot of data space by storing a lot of tuples to represent the occurrences an activity can take place. We use a stereotype of "PK" applied to attributes in the original model to designate the primary identifier for instances of an object. This designation allows us to shift the attribute down the association and swap the positioning of the objects. In this iteration over the objects, we also look for date-time data

Algorithm 1. Denormalization Algorithm

INPUT: normalizedObjects (XMI representation of UML class diagram)
OUTPUT: denormalizedEntities (XMI representation of denormalized entities)

```
foreach object in normalizedObjects
   add entity to denormalizedEntities
   foreach attribute in object
      add attribute to entity
      if object is transactional
         mark attribute as unique
   add id attribute as primary key
   mark id as autoincrement

foreach entity in denormalizedEntities
   if entity is both a many side and a one side of two
relationships
     and  a lookup object
      foreach attribute in object
         if attribute is PK
            add attributes to many side entity
            if attribute is a datetime type expand date pattern
      swap graph location entity of the one sides

foreach association in normalizedObjects
   if association is one-to-may and many side is transactional
      add foreign key to many side entity
      add quantity field to entity on many side
```

 ISBN: 978-1-61208-460-2

types that are part of the primary key. When we locate an occurrence, we replace the attribute with a date-time specification occurrence. The date-time specification includes a starting date, ending date, starting time, ending time and day of the week pattern.

The final pass of the algorithm adds foreign keys and aggregation counters. The aggregation significantly reduces the count of tuples stored. An example of this is shown in Figure 1. Instead of having an instance of each ticket, we add the quantity field to store the aggregate count for the unique attributes.

Figure 2 shows an entity-relationship diagram of a transformed model of Figure 1. Unique attributes have been applied where aggregations should be performed. The activity schedule entity has been shifted out in the graph, and the quantity fields have been added to the aggregated transactional objects.

IV. BUSINESS RULES

Business rule engines have sprung up to allow the separation of business rules from the core application code. The systems are designed to allow the end users to change the business rules freely without changing the original application code. In 2007, International Data Corporation implemented a survey where they asked 'How often do you want to customize the business rules in your software?'. Ninety percent of the respondents reported that they changed their business rules annually or more frequently. Thirty-four percent of the respondents reported that they changed their business rules monthly [15].

Figure 2 shows two tables that implement business rules:

- Activity Schedule – This table implements the date-time pattern mentioned earlier to store the business rules for when a particular activity is valid.
- Price Schedule – This table implements the date-time pattern mentioned earlier to store the business rules for when a particular price is available.

In each case objects in Figure 1, that inserted instances to represent availability, are replaced with rule instances to represent the availability. So instead of having a tuple per availability instance, a single tuple can represent the pattern. In the case of activity schedules, the example year had over 26,000 instances of availability that were replaced with 30 instances of the business rule.

V. OBJECT HISTORY ANALYSIS

One of the main reasons enterprises develop or purchase software solutions to allow the organization to increase their knowledge of their operations through the analysis of the data collected in the software solution. The denormalization solution presented earlier may limit the data available from the denormalization process. The data is presented to the users through dashboards, reports or exports. A dashboard is presented as a graphical chart to measure where the organization stands compared to a goal. Examples of these would be sales to date compared to same period last year. A report has a set of input parameters that control the data displayed. The data displayed in the report tends to include tables with aggregated values. Exports allow for the exporting of data into a two-dimensional table saved as a comma separated value (CSV) format. In this format, attributes represent the columns of the data. Columns are escaped with double quotes and separated by commas. For our purposes, we will refer to all three categories generically as reports.

Current state and historical comparison are the two categories of reports a user may want to pull in their analysis. In current state reports only the latest version of the object is needed. In historical comparison reports, all versions of an object may be needed depending on the level of aggregation. An example of a historical comparison report would be a report that compares sales for the month compared to sales last year in the same month.

In our work, we developed Algorithm 2 to create an in-memory copy of all historical versions of a specific object. We use code to generate the data and then generate the report output. If the organization wanted to allow end users to report on historical versions, they could modify Algorithm 2 to write records as temporary tuples and then call the reporting tool.

Algorithm 2. History Creation Algorithm

INPUT: object
OUTPUT: collection of object's version history

```
Set thisObject = newest version of object
Set objectVersions = empty list
Set fieldVersions = distinct saveDates values from object journal
Sort fieldVersions by saveDate descending
Set lastDate = maximum(saveDate)
Foreach version in fieldVersions
    If lastDate = version.saveDate
      objectVersions.add(thisObject)
   Set thisObject.[version/attribute] = version.value
   Set lastDate = version.saveDate
Return objectVersions
```

VI. EMPIRICAL RESULTS

The empirical results demonstrate the success of representing the example transaction data with significantly lower cloud storage costs. TABLE 1 shows the tuple counts for the original data model and the denormalized data model. Both data models represent the complete 2014 calendar year of visitor transactions for the Gettysburg National Battlefield. The denormalized model creates a 78% reduction in the number of tuples. In the specific case of the

 ISBN: 978-1-61208-460-2

TABLE 1. EMPIRICAL RESULTS

Table	*Normalized Tuples*	*Denormalized Tuples*
user	*31*	*31*
patron	*17,610*	*17,610*
ticket	*738,981*	*157,780*
activity schedule	*26,697*	*30*
price schedule	*220*	*24*
activity	*17*	*17*
Total	*783,556*	*175,492*

force.com [2] platform, the reduction in number of tuples allows the organization to store nearly three years of transaction data in the data storage provided without additional subscriptions costs. In the minimal data storage provided to an enterprise customer of force.com [2], the organization receives 500,000 tuples. Additional data storage is available to the organization for a monthly subscription price of 2,000 tuples per dollar. Using the normalized data model, the organization would not be able to store a complete year of data without needing to purchase more data storage.

VII. CONCLUSION

In this paper, we propose an algorithm for object denormalization when transforming an application domain object model to a data model used in a cloud PAAS data store. Our solution is based on navigating the relationships in a UML class diagram and horizontally compressing classes between multiple one-to-many relationships, aggregating relationships on anonymous relationships and using temporal offering patterns.

In this research, we studied a specific application domain related to humanities organizations. The algorithms can be applied to similar application domains that contain entity objects representing transactions and customers. Future work needs to test our algorithms in other application domains to ensure the work applies across different application domains.

REFERENCES

[1] brainsell blog, "Salesforce, SugarCRM and SalesLogix — Data Storage Costs Compared," 2016. [Online]. Available: https://www.zoho.com/creator/pricing-comparison.html. [Accessed 03 02 2016].

[2] Salesforce.com, inc, "Run your business better with Force.," 2006. [Online]. Available: http://www.salesforce.com/platform/products/force/?d=70130000000f27V&internal=true. [Accessed 03 02 2016].

[3] Zoho Corporation, "Creator Pricing Comparison," 2016. [Online]. Available: https://www.zoho.com/creator/pricing-comparison.html. [Accessed 03 02 2016].

[4] Zoho Corporation, "Compare Zoho CRM editions," 2016. [Online]. Available: https://www.zoho.com/crm/comparison.html. [Accessed 03 02 2016].

[5] C. J. Date, "Denormalization," in *Database Design and Relational Theory*, O'Reilly Media, 2012.

[6] G. L. Sanders and S. Shin, "Denormalization Effects on Performance of RDBMS," in *Proceedings of the 34th Hawaii International Conference on Systems Sciences*, 2001.

[7] J. D. Conley and R. P. Whitehurst, "Automatic and transparent denormalization support, wherein denormalization is achieved through appending of fields to base relations of a normalized database". USA Patent US5369761 A, 29 November 1994.

[8] A. Bisda , "Salesforce Denormalization Delivers New Power for Nurtures," DemandGen, 29 07 2014. [Online]. Available: http://www.demandgen.com/salesforce-denormalization-delivers-new-power-nurtures/. [Accessed 09 11 2015].

[9] D. Taber, Salesforce.com Secrets of Success: Best Practices for Growth and Profitability, Prentice Hall, 2013.

[10] A. Olmsted, "Fresh, Atomic, Consistent and Durable (FACD) Data Integration Guarantees," in *Software Engineering and Data Engineering, 2015 International Conference for*, San Diego, CA, 2015.

[11] Object Management Group, "Unified Modeling Language: Supersturcture," 05 02 2007. [Online]. Available: http://www.omg.org/spec/UML/2.1.1/. [Accessed 08 01 2013].

[12] C. L. Forgy, "Rete: A fast algorithm for the many pattern/many object pattern match problem," *Artificial Intelligence*, vol. 19, no. 1, p. 17–37, 1982.

[13] Object Management Group, "Unified Modeling Language: Supersturcture," 05 02 2007. [Online]. Available: http://www.omg.org/spec/UML/2.1.1/. [Accessed 08 01 2013].

[14] Object Management Group, "OMG Formal Versions of XMI," 06 2015. [Online]. Available: http://www.omg.org/spec/XMI/. [Accessed 11 11 2015].

[15] Ceiton Technologies, "Introducing Workflow," [Online]. Available: http://ceiton.com/CMS/EN/workflow/introduction.html#Customization. [Accessed 15 09 2014].

 ISBN: 978-1-61208-460-2

Modeling Non-Functional Requirements in Cloud Hosted Application Software Engineering

Santoshi Devata, Aspen Olmsted
Department of Computer Science
College of Charleston, Charleston, SC 29401
e-mail: devatas@g.cofc.edu, olmsteda@cofc.edu

***Abstract*- Functional Requirements are the primary focus of software development projects for both end users and developers. The Non-Functional Requirements (NFR) are treated as a secondary class requirement, ignored until the end of the development cycle. They are often hidden, overshadowed and therefore, frequently neglected or forgotten. NFRs are sometimes difficult to deal with and are the most expensive in certain cases. NFRs become even more important with cloud architectures because the concurrent load and response latency are more vulnerable using public networks than they were on private networks. More wok is needed on mapping NFR models into software code. Developing a cloud based system with functional requirements only is not enough to build a good and useful software. NFRs should become an integral part of software development. In this paper, we focus on the modeling of NFRs and the transformations from UML models into the source code. Specifically, we choose three NFRS: response time, concurrency, user response time for a Theater Booking system.**

Keywords - Non-Functional Requirements; NFRs; Response time; Concurrency

I. INTRODUCTION

The software engineering process is intended to produce software with specific functionality that is delivered on time, within budget and satisfying customer's need. Bruegge and Dutoit [1] dedicate the first chapter of their text book to these outcomes and budget constraints. These demands mean that the software development is focused and driven by the functional requirements. The software market is changing every day increasing its demands for providing best quality software that not only implements all the desired functionality but also satisfies the NFRs. Including NFRs in the model, leads to a complete software capable of handling not just the requirements associated with the product but also provides the usability according to the current standards. NFRs (sometimes also referred to as software qualities) indicate how the system behaves and includes requirements dealing with system performance, operation, required resources and costs, verification, documentation, security, portability, and reliability. Thus, satisfying NFRs is critical to building good software systems and expedites the time-to-the-market process, since errors due to not properly dealing with NFR, which is usually time-consuming and complex, can be avoided. However, software engineers need to know whether the performance cost of the algorithms that deal with the various NFRs will violate the basic performance requirements or conflict among themselves.

Failing to address NFRs in the design phase can lead to a software product that may meet all the functional requirements but fail to be useful because it cannot be used. In the United State, the federal government contracted a 3rd party vendor to develop an application for individuals to register for health care coverage. The designers failed to specify the NFRs for concurrency, and the application could not be used because of the mistake. [2]

NFRs are sometimes not intuitive to the developers, so implementing and including them in the development cycles is challenging. There are different approaches to handling the NFRs, but the best way is to model them and implement for each case. Rahman and Ripon [3] describe a use case and the challenge of integrating the NFRs into the design models.

The organization of the paper is as follows: Section 2 describes the related work and the limitations of current methods. In Section 3 we give a motivating example and explain our proposal, present our algorithms and show how they are used in our research. We conclude and discuss future work in Section 4.

II. RELATED WORK

Functional requirements are defined and represented in many ways. These functional requirements are the basis of software development, but NFRs are the ones that supply the rules when implementing the code. Many authors have looked at NFRs and the problems of their inclusion in the design process. Pavlovski and Zou [4] define NFRs as specific behaviors and operational constraints, such as performance expectations and policy constraints. Though there are many discussions about the NFRs, they are not taken as seriously as they should be. Glinz [5] suggest the notion of splitting both functional and NFRs into a set of categories and make groups of

 ISBN: 978-1-61208-460-2

them so that they are inherently considered while developing the applications. Alexander [6] suggests looking at the language used to describe the requirements. Words ending in '-ility' are often the NFRs. Examples of these words are reliability and verifiability. All of their work focuses on identification of the NFRs. Our work builds on theirs by applying domain specific models using extensibility mechanisms built into standard modeling notations.

Ranabahu and Sheth [7] explore four different modeling semantics required when representing cloud application requirements. These include data, functional, non-functional and system. Their work focuses on functional and system requirements. There is a small overlap with our work, but only with nonfunctional requirements from the system perspective. They build on work done by Stuart [8] in his workshops where he defined semantic modeling languages to model cloud computing requirements in the three phases of the cloud application life cycle. The three phases are development, deployment, and management. Our work adds to the missing semantic category of NFRs.

In Ranabahu and Sheth [7] work they use Unified Markup Language (UML) [9] to model the functional requirements only. UML is a standardized notation for representing interactions, structure and process flow of software systems. UML consists of many different diagram types. Individual diagrams can be linked together to model different perspectives of the same part of a software system. We utilize UML to express the NFRs also.

Additional semantics for models can be added by the integration of the matching UML Activity and Class diagrams. UML provides an extensibility mechanism that allows a designer to add new semantics to a model. A stereotype is one of three types of extensibility mechanisms in the UML that allows a designer to extend the vocabulary of UML to represent new model elements [10]. Traditionally the semantics were consumed by the software developer and manually translated into the program code in a hard coded fashion.

Object Constraint Language (OCL) [11] is part of the official Object Management Group (OMG) standard for UML. An OCL constraint formulates restrictions for the semantics of the UML specification. An OCL constraint is always true if the data is consistent. Each OCL constraint is a declarative expression in the design model that states correctness. Expression of the constraint happens on the level of the class, and enforcement happens on the level of the object. OCL has operations to observe the system state but does not contain any operations to change the system state.

Our contribution in the domain of cloud computing software modeling is in the use of modeling standards such as UML and OCL with their extensibility mechanism of stereotypes to model the NFRs. We demonstrate these models can be transformed into application source code through the application of three application domain constraints.

III. RESEARCH PROPOSAL

Our contribution in this work is to look at application domain specific NFRs that are useful for cloud based application architectures. We model the NFRs using the extensibility mechanisms built into the standard modeling notations of UML and OCL to specify those NFRs. We then auto-generate code for NFRS using Java. We demonstrate the NFRs using a theater booking system example**.** A theater booking system is an online application used by theatres to sell their entrance tickets. Figure 1 shows an activity diagram for a theatre booking system where the NFRs are represented using UML stereotypes. We chose to focus on three NFRs for this study: response time, concurrency, and pick seat time to implement the theater booking system. For each NFR, we model in UML and OCL utilizing stereotypes to apply the additional required semantics. We then generate code from the model to enforce the NFRS. The code is generated for these NFRs for use in a cloud application that uses the threads on the server side for each client.

Request response time is one of the key performance measures in a theater booking system. It is an NFR that is represented as a 'Response time' stereotype in the UML activity diagram (Figure 1). This stereotype is related to every interaction between the client and the server. In general terms, response time can be defined as the amount of time system takes to process a request after it has received one. A control flow in an activity diagram can be assigned the stereotype. In the algorithm that is used to enforce for this stereotype, the time is noted right before the request is sent to the server and the difference is measured once the response is received. The difference between the send and receive time gives the response time for the request. Specific stereotypes are used to represent different latency requirements. Examples are "low latency: and "high latency". Runtime configuration can define the allowable time for each stereotype. Response time for each request from the users is measured, and the average response is calculated for the overall system. This is particularly useful to measure the overall system performance and compare it over time. If the average response time increases, we can further get the average response of each module/type of requests and find the bottlenecks. Algorithm 1 shows the algorithm implemented to guarantee this NFR. In the algorithm, the client notifies the server when the timeout occurs to enable the server to rollback any partial work completed.

 ISBN: 978-1-61208-460-2

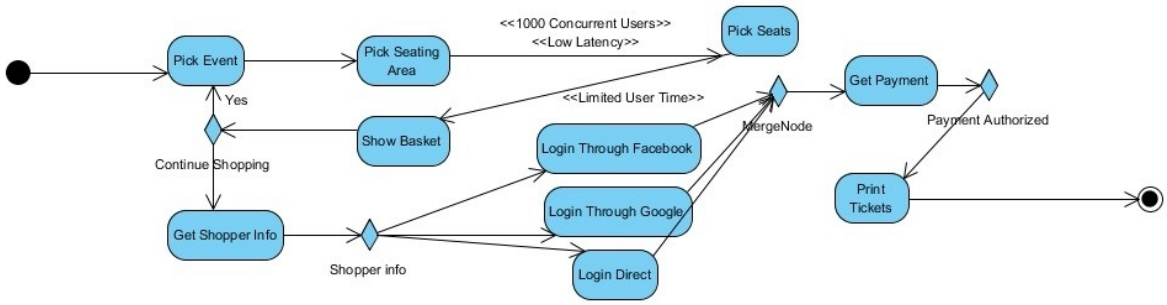


Figure 1 Activity Diagram for A Theater Booking System

Concurrency is a robustness measure of applications, and especially for any online booking system. This is represented as 'Concurrent Users' stereotype in the UML activity diagram (Figure 1). We represent the threshold concurrency as the 'Concurrent Users' stereotype in the UML activity diagram (Figure 1). We implemented this by spawning a thread pool with the size based on stereotype. The server then handles the request by creating a queue. Requests are pulled from the queue and assigned a thread from the pool to process the request. In the implementation of this stereotype, we measure the latency of the request by noting the time right before the request is sent to the server also when the response is received. This difference between the two values gives the latency for that request. This latency for each request is measured and the queue time is appended to the log. The log of measurements is particularly useful to measure the overall system performance and compare it over time. If the average latency time increases, we can further get the average latency of each module/type of requests and find the bottlenecks. When handling the concurrency stereotype, the bottleneck is often caused by a pool of threads that is smaller than the demand on the server. Algorithm 2 shows the algorithm implemented to guarantee this NFR.

Handling the situation where a user does not respond to a form in an appropriate amount of time is another important NFR for many systems. In the theater booking system, while the user is picking a seat, resources are locked from other users. The time the locks are held needs to be minimized. We represent the form response time requirement as 'Limited user time' stereotype in the UML activity diagram (Figure 1). The stereotype is specific to the pick seats activity in the ticketing application domain. The user should be given a limited time to respond when selecting the seats. We implemented this by binding an event to the request submission of the client. When the user tries to pick the seats, the client application polls continuously to check if the request is sent during the specified time. When there is a delay of more than the time specified by the stereotype, then the user gets a message indicating that

Algorithm 1. Request Response Timeout

INPUT: XML of Send to Server, timeout
OUTPUT: XML of response with server

```
Send request to server
Set timer to fire every second
Set timeExpired = 0
Do
   Check if response is received
While timeExpired < timeout or response received
If not response received
   Set response to time expiration error
   Set response to timeout error
    Notify server of timeout
End if
Return response
```

Algorithm 2. Concurrency

INPUT: XML of request, timeout
OUTPUT: XML of data entered or XML with error

```
Check if anythreads in pool
If no threads in pool
   Set timer to fire every second
   Set timeExpired = 0
   Do
      Check if thread available in the pool
   While timeExpired < timeout or thread received
If not thread received
   Set response to timeout error
ELSE
    Execute request in thread
End if
Return response
```

 ISBN: 978-1-61208-460-2

Algorithm 3. User Response Timeout

INPUT: XML of form to display, timeout
OUTPUT: XML of data entered or XML with error

```
Show form to user
Set timer to fire every second
Set timeExpired = 0
Do
   Check if response is received
While timeExpired < timeout or response received
If not response received
   Notify user of time expiration
   Set response to timeout error
End if
Return response
```

there locks have been released. If the request is sent before the specified time, then the user will proceed to next activity. Algorithm 3 shows the algorithm implemented to guarantee this NFR.

IV. CONCLUSION/FURTHER RESEARCH

In this work, we show that it is possible to model many cloud based software NFRs using UML stereotypes. UML diagrams have been used for years to model the functional requirements of the application. We extended the modeling of functional requirements by using UML stereotypes to model the NFRs in the same design model. The UML stereotype is transformed to application code that guarantees the NFR will be enforced. Future work will enhance our work to include OCL constraints to broaden the type of NFRs that can be modeled and transformed into cloud application code.

REFERENCES

[1] B. Bruegge and A. Dutoit, Object-Oriented Software Engineering, Prentice Hall, Inc, 2010.

[2] T. Mullaney, "Obama adviser: Demand overwhelmed HealthCare.gov," USA Today, 06 10 2013. [Online]. Available: http://www.usatoday.com/story/news/nation/2013/10/05/health-care-website-repairs/2927597/. [Accessed 24 02 2016].

[3] M. M. Rahman and S. Ripon, "Elicitation and Modeling Non-Functional Requirements – A POS Case Study," *International Journal of Future Computer and Communication,* vol. 2, no. 5, pp. 485-489, 2013.

[4] C. J. Pavlovski and J. Zou, "Non-functional requirements in business process modeling," *Proceedings of the Fifth on Asia-Pacific Conference on Conceptual Modelling, vol. 79,* 2008.

[5] M. Glinz, "Rethinking the Notion of Non-Functional Requirements," *Third World Congress for Software Quality, Munich, Germany,* 2005.

[6] Alexander, I, "Misuse Cases Help to Elicit Non-Functional Requirements," *Computing & Control Engineering Journal, 14, 40-45,* 2003.

[7] R. Ajith and A. Sheth, "Semantic Modeling for Cloud Computing, Part I," *Computing,* vol. May/June, pp. 81-83, 2010.

[8] S. Charlton, *Model Driven Design and operations for the Cloud,* Towards Best Practices in Cloud Computing Workshop, 2009.

[9] Object Management Group, "OMG Formal Versions of UML," 06 2015. [Online]. Available: http://www.omg.org/spec/UML/. [Accessed 11 09 2015].

[10] Object Management Group, "Unified Modeling Language: Supersturcture," 05 02 2007. [Online]. Available: http://www.omg.org/spec/UML/2.1.1/. [Accessed 08 01 2013].

[11] Object Management Group, "OMG Formally Released Versions of OCL," 02 2014. [Online]. Available: http://www.omg.org/spec/OCL/. [Accessed 09 11 2015].

 ISBN: 978-1-61208-460-2

Enabling Resource Scheduling in Cloud Distributed Videoconferencing Systems

Álvaro Alonso, Pedro Rodríguez, Ignacio Aguado, Joaquín Salvachúa
Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid
Madrid, Spain
email:{aalonsog, prodriguez, iaguado, jsalvachua}@dit.upm.es

***Abstract*—When deploying videoconferencing systems in Cloud based infrastructures, one of the most complex challenges is to distribute Multipoint Control Units (MCUs) among different servers. By addressing this challenge, we can improve the flexibility and the performance of this type of systems. However, to actually take advantage of the Cloud possibilities we have also to introduce mechanisms to dynamically schedule the distribution of the MCUs across the available resources. In this paper, we propose a resource scheduling model for videoconferencing systems and, starting from an existing MCU distribution architecture, we design a solution to enable the resource scheduling basing on custom criteria. These criteria can be based on the characteristics of each server or in their status in real time. We validate the extended model by setting up a typical videoconferencing deployment among a set of Cloud providers and testing a decision algorithm. We conclude that the proposed model enables the use of a wide range of algorithms that can be adapted to the needs of different Cloud deployments.**

***Keywords*—cloud computing; videoconferencing; distributed MCU; scheduling**

I. INTRODUCTION

Nowadays, a very important part of the applications and services we consume over Internet are provided in Cloud [1] infrastructures. The main advantage of using this technology is that one can adapt the amount of provisioned resources for a service based on the demand. In traditional deployments, one has to forecast the demand before obtaining the hardware and this involves a lack of flexibility that can result in a waste of resources. However, using Cloud based deployments one can dynamically scale a service in almost real time by adding or removing computing capacity. In other words, Cloud Computing provides the illusion of infinite computational power on a pay-per-use basis.

In addition to this advantage, this flexibility enables easier and more efficient ways to distribute the services among different servers. Thus, one can balance the system load, replicate instances or geographically distribute them. These operations are done almost instantaneously with a *click* or by calling an API. Even some public Cloud providers offer components that automatically distribute the load between servers. However, these mechanisms are usually designed to be used by request-response services such as RESTful or web services.

In videoconferencing systems, the communication between participants is usually performed via a central server called Multipoint Control Unit (MCU). MCUs are used to address the signalling and to interchange the media streams between peers. Furthermore, in advanced configurations they are used to record sessions or to transcode video and audio flows.

Today, thanks to technologies such as Adobe Flash [2] and HTML5 [3] with its real-time communications standard Web Real Time Communications (WebRTC) [4], videoconferencing systems are accessible from web applications and mobile devices and its use is open to a higher number of users than ever before. On the other hand, the demand of these services can vary dynamically in short periods of time. The fluctuation in the number of users and the managed sessions results in substantial changes in the computing capacity consumed by the MCUs. Hence, and this is strengthened in [5], deploying MCUs in Cloud infrastructures offers several advantages. One of them is the already mentioned idea of distributing a service among several servers. But, as anticipated, the standard solutions for resource scheduling do not apply to the specific case of MCU distribution.

In traditional web services based on HTTP, it is usually enough with a distribution of the users requests using load balancers. These components are adapted to the requirements of complex web services. However, we argue no general solution for distributed videoconferencing systems is available. Here, the resource use in each of the distributed nodes may depend on several parameters besides the number of users. This can be understood with a very easy example. Suppose an scenario with an MCU distributed in two servers and managing six users each. In the first MCU the six users are connected to the same videoconferencing room and in the second one each user is connected alone to a different room. Obviously, the first MCU is consuming more resources because it is receiving packets from each user and broadcasting them to all of the rest while the second one is only receiving packets from the users without any processing to do. Furthermore, to address the scheduling challenge in this scope we first need to understand which type of criteria we need to take into account and then we must enable a way to schedule the load distribution taking those criteria into account.

In the next Section, we analyse the existing solutions regarding resource scheduling in the Cloud and why they do not cover the videoconferencing scenario. Then, in Section III we describe our scheduling model and extend an existent distributed architecture to cover the needed requirements. Then, in Section IV we validate the solution with a real implementation and a typical use case. Finally, in Section V we enumerate the main conclusions obtained and we analyse the different research lines we open to continue the work.

II. RELATED WORK

Scheduling Cloud Computing resources has always been a subject under study. Being able to dynamically adapt and change the available resources depending on the demand of the users is a key point in services deployed in the Cloud. Many strategies have been designed in order to solve this

 ISBN: 978-1-61208-460-2

problem, using techniques like the Genetic Algorithm [6], Particle Swarm Optimization (PSO) [7], Ant Colony Optimization (ACO) [8] or Load Balancing Ant Colony Optimization (LBACO) [9]. All of them try to find an optimal resource allocation for workload in a generic Cloud architecture.

In spite of all of this work, there is not any solution specifically designed for a videoconferencing service. However, there exists a model which, if properly developed, could be useful. The model described in [10] could be the first step of Scheduling a videoconference service in the Cloud. There, the authors propose an architecture that allows to easily divide an MCU in atomic parts called One To Many (OTMs). An OTM is a software component that basically broadcasts a video/audio stream to many participants. And these OTMs can be distributed among different servers without introducing extra latency in the communications.

It is important to define a valid and efficient algorithm for this kind of services to obtain better performance and more flexibility. However, in order to achieve this goal it is essential to propose a mechanism that allows us to implement it in the model described above. No other solutions have been defined for this new layer of functionality in a videoconference service, but this work defines, implements and tests one.

We extend the OTM model by adding the layer that allows us to decide how to schedule the resources among the distributed OTMs (dOTMs). These decisions are customisable and can be based on the characteristics or the status of the servers where the OTMs are deployed. As an example case we also provide a basic algorithm. With this mechanism, specifically designed for a videoconferencing service architecture, it is possible to define new algorithms, grouping different criteria.

III. VIDEOCONFERENCING RESOURCE SCHEDULING

We have concluded that the first thing we need to start working for an efficient videoconferencing services distribution is to design a model that allows us to introduce custom decisions to schedule the resources. In this section, we describe the model we propose by analysing the main characteristics it has to cover. We also design an architecture that extends the existing MCU distribution model dOTMs to comply with those requirements.

A. Model description

As introduced above, distributed MCU servers differ from common web services in many aspects. Videoconferencing is not a request-response service and the amount of resources needed per user depends on the size of the videoconferencing sessions. According to this, we cannot distribute the users homogeneously among distributed MCUs. When connecting a new user, we need to know certain information about the current status of each of the available servers to decide which to use. Once the decision is taken, we need to be able to assign the connection to the selected MCU. Thus, the model we are proposing needs to cover three main aspects:

1) Decision layer: Having a distributed MCU, we need a central component in charge of deciding where to allocate each connection. This component has to be configurable with custom decision algorithms that take as input the list of available MCUs and their information and return the selected one. Once the decision is taken this component has to be able to communicate directly with the selected MCU to allocate the resources there. Moreover, the algorithms can be automatically modified in real-time taking into account the feedback received from the MCUs.

2) MCU registration: To be able to communicate with a specific MCU, the decision layer must have a list with all the available ones. So, an MCU has to be registered in the central component when it is added to the pool of available MCUs. In the registration, the MCU can specify a set of fixed characteristics of itself that can be used by the algorithms at decision time.

3) MCU report: The fixed information set at registration time is not enough. In the decision layer we also need real-time information about the status of the MCUs. Thus, the algorithms can decide also basing on parameters such us the CPU or Memory use of each MCU. Therefore, we also need a returning channel between the decision layer and each MCU.

B. Architecture design

To put these three requirements in a real MCU distribution model we have designed an architecture that extends the one described in [10]. The key of this architecture is the design of a mechanism to split a traditional software MCU into smaller components called OneToManys (OTMs). A OTM receives packets from a source and forwards it to many destinations, usually participants in a videoconferencing session. If all the participants of a session are sharing their media with the others, the MCU receives the media packets from each participant and forwards them to the rest. Thus, with the dOTMs model, we can manage a session using a OTM for each participant. Running each OTM in a single process we can distribute the same session among different servers. To achieve this it is necessary to isolate the media layer of the MCU.

The dOTMs model proposes the division of a traditional MCU into three layers: signalling, control and media. Therefore, as detailed in [11], this separation results in an architecture with three main components:

- *OTM*: the software unit in charge of receiving media packets from a participant and broadcast it to many.
- *Agent*: the component in charge of managing OTMs. We have an Agent per each machine in which we want to host OTMs.
- *Controller*: manages videoconference rooms and interchanges the signalling messages between clients and OTMs. It also communicates with Agents to start and stop OTMs.

To perform the signalling, these components communicate between them using a Message Bus. When a new participant joins a videoconferencing session and wants to publish media, the Controller asks an Agent to create a new OTM and establishes the signalling between the client and the created OTM. When they are connected the media communication takes place directly between them. When other participant wants to subscribe to the published media, the Controller searches the corresponding OTM and establishes the connection in the same way.

If we start Agents in different servers we can distribute even the same videoconferencing session in different infrastructures taking advantage of the Cloud benefits exposed before. However, the dOTMs model does not specify how to schedule the resources between the available Agents. We have to adapt the model to the requirements explained above modifying the way

 ISBN: 978-1-61208-460-2

in which the Controller and the Agents communicate between them.

Reviewing the three requirements explained in the model description and analysing the current dOTMs architecture we can observe the following:

1) Agent decision: When a new OTM is required in a session, the Controller selects the Agent in round-robin mode sending the creation request to one of the available Agents each time. We need to support the creation of OTMs in specific Agents basing on custom algorithms. In order to enable that, we have to introduce a way to allow the Controller to communicate directly with a specific Agent.

2) Agent registration: The Controller does not really have awareness about the existing Agents. To communicate with them sends a message to a message bus and the bus is the one that redirects it to one of the subscribed Agents. We need to add a mechanism to provide the Controller a list of the available Agents in every moment. In other words, every Agent has to be registered in the Controller.

3) Agent report: To take the decision of which Agent select to create a new OTM, the Controller needs information of the available Agents. It is specially interesting to have information of the status of each Agent in real time. In the current configuration, once an OTM is created the Controller sends messages directly to it and does not need to communicate with the Agent anymore. To support a real time report from the Agents we need to enable a persistent communication channel between each Agent and the Controller.

Figure 1 illustrates the extensions we propose in this paper. We introduce a new message queue for each of the Agents (*Agent id*). These queues are configured in a direct unicast mode and used by the Controller to communicate directly with an Agent. When a new Agent is added to the environment it creates a new queue to be able to receive messages. All the Agents are still subscribed to the common queue (*Agent*) but now that queue is able to send broadcast messages to all the Agents. This queue is used by the Controller to send a periodic message that will be answered by the existing Agents. In the response of these messages each Agent includes three types of information:

- Contact information: the needed information to contact the Agent. It basically includes the id of the queue created by the Agent.
- Fixed information: constant information that is configured when creating the Agent.
- Realtime information: information about the state of the Agent in each moment.

With this information sent periodically, the Controller has an updated list of the available Agents. Furthermore, when a new OTM is needed it can use the information (both fixed and real-time) of the Agents to decide to which of them delegate the creation. Once decided it sends the creation message using the specific queue. The broadcast queue is still configured to be able to send messages in round robin mode. Thus, if one does not want to specify which Agent to use and wants to evenly distribute the connections between all of them, the system can be used as before. Finally, the periodic broadcast messages are also used as a heartbeat to ensure that an Agent is still available and reachable. The Controller stores a count of the not responded messages to determine when an Agent is not usable anymore.

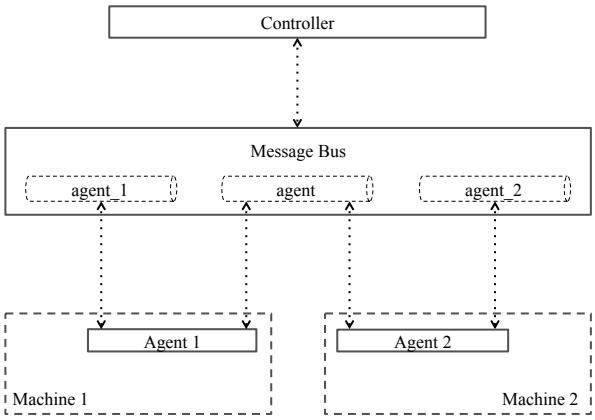


Figure 1: Message Bus configuration

TABLE I: MESSAGES FROM THE CONTROLLER TO THE AGENTS

createOTM	
Requests a new OTM to an Agent. The Agent is selected in round robin mode.	
Queue	Agent
Type	unicast
Parameters	-
Returns	OTM id
createOTM	
Requests a new OTM to a specific Agent.	
Queue	Agent_id
Type	unicast
Parameters	-
Returns	OTM id
deleteOTM	
Forces the destruction of an OTM. The message is sent to all the Agents and only the Agent that owns the OTM deletes it.	
Queue	Agent
Type	broadcast
Parameters	OTM id
Returns	Result code
getAgents	
Requests information to all the Agents available in the system.	
Queue	Agent
Type	broadcast
Parameters	-
Returns	Stats object

In Table I, we can see the detailed specification of the messages we need to enable the proposed mechanism. The two original messages (*createOTM* and *deleteOTM*) are the same, but we need a new *createOTM* message to create in a specific Agent (sent to the unicast queue of the Agent) and the *getAgents* broadcast message to obtain the status of each Agent.

 ISBN: 978-1-61208-460-2

IV. VALIDATION

We have proposed a mechanism that, having a videoconferencing system distributed between several nodes, allows us to dynamically decide how to schedule the load among them. In this section, we validate the mechanism testing a working implementation of the model in a real videoconferencing deployment.

A. Implementation

To validate the solution we use an open source project named Licode [12]. Licode is developed by the authors of this paper and provides a WebRTC compatible videoconferencing system with three main parts.

The first one is an MCU built on top of the WebRTC standard that implements a signalling protocol based on SDP exchange. For the establishment of the media connection it implements the Interactive Connectivity Establishment (ICE) [13] standard and for the encryption of all the media data Secure Real-time Transport Protocol (SRTP) [14] and Datagram Transport Layer Security (DTLS) [15] protocols. The second part is a JavaScript client API that wraps the WebRTC API facilitating the development of videoconferencing applications and adding the necessary modules to communicate with the MCU. Finally, to ensure the security in the signalling between clients and MCU, Licode includes an authorisation module based on Nuve [16]. This module is also in charge of the management of rooms and it is able to start entire MCUs in different machines in order to scale the system.

The current Licode version implements the mechanism exposed in this work using the Advanced Message Queuing Protocol (AMQP) [17] protocol for the message bus, more specifically the RabbitMQ implementation. To send the unicast messages we have implemented an Remote Procedure Call (RPC) mechanism using *direct* exchanges and for the broadcast messages we use *topic* exchanges.

When the Controller sends the *getAgents* broadcast message to get the information about the existing Agents, they respond with the following information:

- *Agent_id*: a unique identifier of the Agent.
- *rpc_id*: the identifier of the AMQP queue to which the Agent is subscribed.
- Metadata object: a JSON object used to store fixed information when starting the Agent.
- Stats object: a JSON object with realtime information about the state of the Agent. It includes CPU and memory usage.
- Keep Alive counter: a counter used to ensure that the Agent is still responding to the requests. When the counter reaches an established limit, the Agent is unregistered in the Controller.

To take the decision of which Agent choose to create a new OTM, the Controller uses the round robin mode as default. Custom policies can be configured by introducing programmatic scripts. A custom script receives as a parameter an object with the Agents information, performs the decision based on it and returns the selected Agent.

B. Experiment description

Using Licode project and its implementation of the model proposed, we have performed an experiment to prove that the solution works in a real use case. We propose a very common scenario in which we (as supposed videoconference as a service provider) need to provide videoconferencing rooms on demand to a variable number of users.

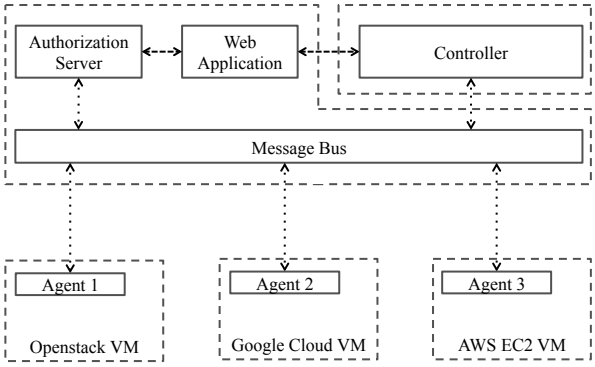


Figure 2: Deployment configuration

TABLE II: INSTANCES USED FOR THE AGENTS

Cloud Platform	Instance Type	CPU	Memory
Openstack	m1.small	1 vCPU	2 GB
Google Cloud	f1-micro	1 vCPU (Shared GCPU)	0.60 GB
Amazon EC2	t2.micro	1 vCPU (Variable ECU)	1 GB

To host the needed resources we own a set of physical computers in which, to facilitate the deployment, we have set up a private cloud environment to deploy virtual machines. If we eventually need more resources we have the possibility of hosting them in two different public cloud providers. The component that we will deploy in the public cloud providers is the Agent in charge of creating new OTMs. In this case the Agent is the bottleneck when it comes to resources use. As seen in [10], dOTMs are limited by CPU, bandwidth or memory are not a limiting factor in public clouds' low processing power instances. Thus, for the purpose of this experiment we will replicate the conditions and simplify load metrics to only take into account CPU measurements.

The deployment of resources in those public Clouds implies an economic cost so the criteria is to prioritise the use of the private Cloud and only when we do not have compute capabilities there, we use the public ones. However, in this case we always want to reserve a part of the computing power of the private Cloud to host private meetings so we will configure a use threshold below the maximum computational level.

Once the set up is ready, we start connecting clients to videoconferencing rooms. First clients will be handled in the private cloud and when the configured threshold is reached, we will start handling the new ones in the public Clouds. To decide which of the two available public Clouds we will select, we will follow load criteria, using in each moment the one that is consuming less resources. We also define a weight factor between both public clouds that can be used to set a priority taking into account different criteria.

C. Deployment set up

To deploy the needed component for the experiment we have chosen Openstack Compute [18] as the private cloud and Amazon Web Services EC2 [19] and Google Compute Engine [20] as the public ones. In Figure 2 we can see a diagram of the configuration.

 ISBN: 978-1-61208-460-2

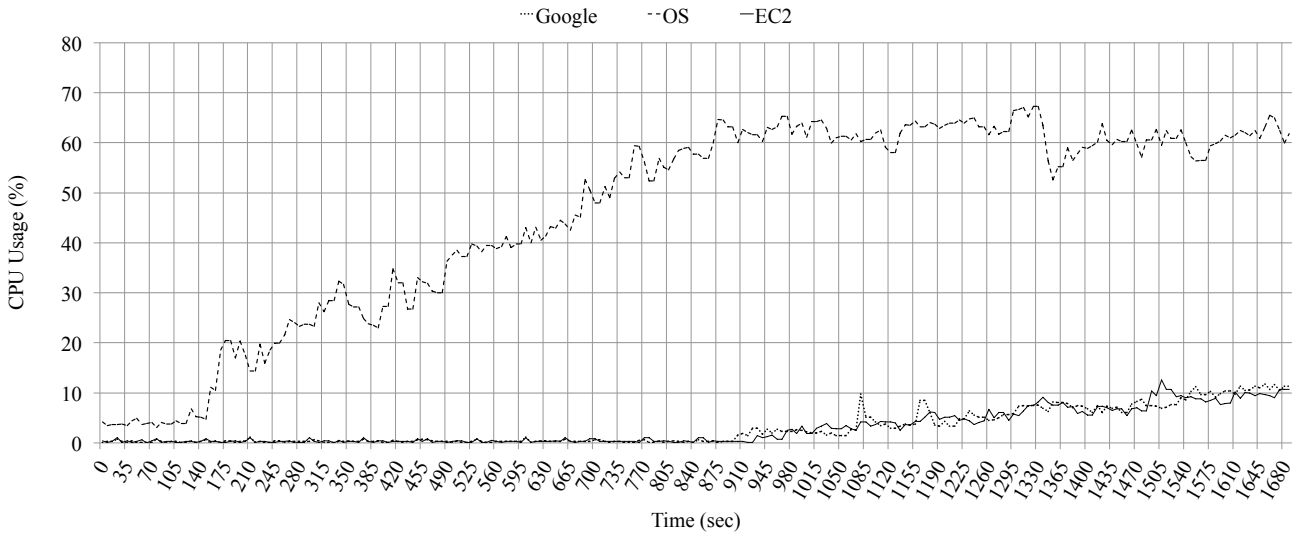


Figure 3: Cloud Providers CPU Usage

Algorithm 1 Agent decision

Require: $cpuThreshold$, $weightFactor$, $osAgent$, $googleAgent$, $ec2Agent$

```
osCPU ← getCurrentCPU(osAgent)
if osCPU < cpuThreshold then
    return osAgent
else
    googleCPU ← getCurrentCPU(googleAgent)
    ec2CPU ← getCurrentCPU(ec2Agent)
    if googleCPU * weightFactor < ec2CPU then
        return googleAgent
    else
        return ec2Agent
    end if
end if
```

We have deployed the Controller in a separate virtual machine. In other virtual machine we have deployed the Authorisation Server (Nuve), the Web Application server and the RabbitMQ server. The infrastructure in which these components have been deployed is not relevant for this experiment. Nor the capabilities of the virtual machines. On the other hand we have deployed an Agent in a virtual machine of each of the mentioned cloud providers, Openstack, Google Cloud and Amazon EC2. The characteristics of the selected virtual machines are described in Table II. The operating system used is Ubuntu 14.04 LTS for all of them.

The small size and computing power of the selected virtual machines is not a problem in the scope of the scenarios we designed. It is very convenient to have a low enough processing power that we can saturate easily to test the ability to allocate OTMs. Regarding the amount of bandwidth available for this type of instance, we have tested it is enough to handle the amount of users we are going to connect.

To host the clients we have used the same instance type as the Amazon EC2 Agent. They connect using Chromium [21] browser (the open-source project behind the Google Chrome) version 45.

Algorithm 1 shows the decision policy we have configured in the Controller. The algorithm is executed everytime we need a new OTM. In the Metadata object of each Agent, we can find the cloud provider where it is running. This way we obtain the values of $osAgent$, $googleAgent$ and $ec2Agent$. We set this parameter in each Agent at boot time. On the other hand, the Stats object allows as to get the current CPU consumption of each Agent ($getCurrentCPU()$ method). In this experiment we have configured the following constants:

- $cpuThreshold$: 60 %
- $weightFactor$: 1

With this threshold value we ensure that a 40% of the private cloud computing capability is reserved for special rooms that we need to host internally. In this experiment we have set a neutral weight factor. This means that there is no priority between the public cloud providers. When the public part is needed, we always create the Agent in the infrastructure that is consuming less CPU. However, using this factor we could design algorithms that establish priorities taking into account advanced criteria such us the pricing, the geographical location, etc. Furthermore, we can use algorithms with feedback that modify the value of that factor taking into account real-time aspects, such as the fluctuation of the pricing.

D. Results

We have created six videoconferencing rooms and connected clients to a random one every 35 seconds, starting in second number 110. This dynamic is only changed in second number 1320, when five clients disconnect from their rooms. The results of the experiment are showed in Figure 3.

As it can be observed, on one hand the CPU usage in the Openstack vitual machine keeps growing as new clients connect to the service. On the other hand, the activity in the Google Cloud and Amazon EC2 ones is almost null. The first key point can be found at second number 875, where the threshold of 60% is first exceeded. From that moment the CPU Usage in Google Cloud and Amazon EC2 virtual machines starts growing, while the Openstack one suffers minor variations but remains almost constant. The growth of

 ISBN: 978-1-61208-460-2

Google Cloud and Amazon EC2 stops at second 1330 as a consequence of the disconnection of five clients explained before. From that moment, the CPU usage of the Openstack virtual machine stops exceeding the threshold, so new clients start connecting again to it instead of connecting to Google Cloud or Amazon EC2 ones. Finally, in second number 1410 the threshold is exceeded again, so the Openstack CPU usage remains constant and the Google Cloud and Amazon EC2 starts growing again until the end of the measures in second 1680.

V. Conclusions and future work

Cloud Computing provides numerous benefits to scalable and distributed services. There are several studies regarding how to efficiently distribute web services or databases taking advantage of the Cloud. But in relation with videoconferencing systems and MCU servers the literature is not so extensive. This type of systems has particular characteristics that make the traditional load balancers not optimal.

In this paper, we have defined a model for scheduling resources in videoconferencing Cloud deployments and designed an architecture that enables this scheduling basing on the realtime status of the MCUs that are managing the sessions. Thanks to this new model, we can design cloud-based scenarios in which we distribute the load according to advanced decision algorithms. To validate that the model actually covers the requirements of videoconferencing scalable systems, we have set up a real deployment using a set of well known Cloud providers and illustrating a very common use case. For this we have used a complete implementation of the model configuring it with an algorithm prototype that simulates the requirements of a videoconferencing as a service provider.

The main conclusion after the experiment is that the proposed model actually works and offers a customisable way to distribute a videoconferencing service according to the specific requirements of each deployment. Once we have the tools to enable this custom decisions, the main challenge is to study algorithms that could standardise the requirements of typical videoconferencing scenarios. Thus, multiple lines of research are opened from this point. The immediate one is to test other kind of algorithms that take into account different criteria than the load of the virtual machines. For instance, a scheme based on the pricing of each Cloud provider should be explored.

If we geographically distribute the Agents, we can also take into account latency constraints to improve the quality of the communications connecting each client to the closer Agent. Furthermore, if the Agents can be connected between them using trees we can improve the latencies even more. As we can see in [22] using hybrid clouds also offers cost saves in many cases.

Other interesting line is the design of algorithms with feedback that are modified in real-time taking into account the status of the Agents. Finally and putting all of these things together, it is interesting the study of common characteristics of videoconferencing deployments that allow us to generalise the requirements and design a global procedure to schedule the resources.

References

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf [retrieved: March, 2013], 2009.

[2] Adobe Flash Player. [Online]. Available: http://get.adobe.com/en/flashplayer/ (retrieved: January, 2016)

[3] HTML5 W3C. [Online]. Available: http://dev.w3.org/html5/spec/ (retrieved: January, 2016)

[4] A. Bergkvist, D. C. Burnett, C. Jennings, and A. Narayanan, "Webrtc 1.0: Real-time communication between browsers," August 2012.

[5] A. Alonso, P. Rodriguez, J. Salvachua, and J. Cerviño, "Deploying a multipoint control unit in the cloud: Opportunities and challenges," in CLOUD COMPUTING 2013, The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization, 2013, pp. 173–178.

[6] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu, "Independent tasks scheduling based on genetic algorithm in cloud computing," in Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on, Sept 2009, pp. 1–4.

[7] S. Pandey, L. Wu, S. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, April 2010, pp. 400–407.

[8] X. Lu and Z. Gu, "A load-adapative cloud resource scheduling model based on ant colony algorithm," in Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, Sept 2011, pp. 296–300.

[9] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in Chinagrid Conference (ChinaGrid), 2011 Sixth Annual, Aug 2011, pp. 3–9.

[10] P. Rodriguez, A. Alonso, J. Salvachua, and J. Cervino, "dOTM: A mechanism for distributing centralized multi-party video conferencing in the cloud," in The 2nd International Conference on Future Internet of Things and Cloud (FiCloud-2014). IEEE, 2014, pp. 61–67.

[11] P. Rodríguez, A. Alonso, J. Salvachúa, and J. Cervino, "Materialising a new architecture for a distributed mcu in the cloud," Computer Standards & Interfaces, pp. –, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0920548915001014 (retrieved: January, 2016)

[12] Licode. [Online]. Available: http://lynckia.com/licode (retrieved: January, 2016)

[13] J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," Internet Requests for Comments, RFC Editor, RFC 5245, April 2010. [Online]. Available: http://tools.ietf.org/html/rfc5245 (retrieved: January, 2016)

[14] M. Baugher, "The Secure Real-time Transport Protocol (SRTP)," Internet Requests for Comments, RFC Editor, RFC 3711, March 2004. [Online]. Available: http://tools.ietf.org/html/rfc3711 (retrieved: January, 2016)

[15] E. Rescorla, "Datagram Transport Layer Security," Internet Requests for Comments, RFC Editor, RFC 4347, April 2006. [Online]. Available: http://tools.ietf.org/html/rfc4347 (retrieved: January, 2016)

[16] P. Rodríguez, D. Gallego, J. Cerviño, F. Escribano, J. Quemada, *et al.*, "Vaas: Videoconference as a service," in Collaborative Computing: Networking, Applications and Worksharing, 2009. CollaborateCom 2009. 5th International Conference on. IEEE, 2009, pp. 1–11.

[17] "OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0," OASIS Standard, Tech. Rep., October 2012.

[18] Openstack. [Online]. Available: http://www.openstack.org/software/openstack-compute/ (retrieved: January, 2016)

[19] Amazon EC2. [Online]. Available: http://aws.amazon.com/ec2 (retrieved: January, 2016)

[20] Google Compute Engine. [Online]. Available: http://cloud.google.com/compute (retrieved: January, 2016)

[21] Chromium. [Online]. Available: http://www.chromium.org/ (retrieved: January, 2016)

[22] J. Cervino, P. Rodriguez, I. Trajkovska, F. Escribano, and J. Salvachua, "A cost-effective methodology applied to videoconference services over hybrid clouds," Mobile Networks and Applications, vol. 18, no. 1, pp. 103–109, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11036-012-0380-4 (retrieved: January, 2016)

 ISBN: 978-1-61208-460-2

Energy Saving in Data Center Servers Using Optimal Scheduling to Ensure QoS

Conor McBay, Gerard Parr and Sally McClean
School of Computing and Information Engineering
Ulster University
Northern Ireland, United Kingdom
Email: mcbay-c1@email.ulster.ac.uk, {gp.parr, si.mcclean}@ulster.ac.uk

***Abstract*— With the rise in popularity of cloud computing the amount of energy consumed by the cloud computing data centres has increased dramatically. Cloud service providers are aiming to reduce their carbon footprint by reducing the energy their data centres produce, while maintain an expected Quality-of-Service adhering to set Service Level Agreements. In this paper, we present our suggested approach for using previously researched energy efficiency techniques, particularly Dynamic Voltage/Frequency Scaling and sleep states, more efficiently through the aid of an SLA-based priority scheduling algorithm, and the results we expect from our research.**

Keywords—cloud computing; energy; DVFS; sleep mode; scheduling; quality-of-service.

I. INTRODUCTION

As the use of ICT continues to become an essential aspect of modern life its increased usage has caused the high production of greenhouse gases, contributing 2-3% of global emissions, and is rising each year [1]. The recent popularisation of cloud computing is a contributing factor to this heightened usage. Cloud service providers aim to maintain an expected Quality-of-Service (QoS) and meet set Service Level Agreements (SLAs) while attempting to be cost effective. One of the biggest costs to providers is energy consumption in cloud data centers. Servers, essential for operations within the data center, consume vast amounts of energy. At times large amounts of energy is being wasted on servers that are not operating at their full capacity and may in fact be in an idle state, doing nothing.

The problem faced by cloud service providers is how to reduce the energy consumption within their data centers, while also maintaining the expected QoS and SLA requirements. There has been a lot of research into ways of dealing with the relationship between energy consumption and expected performance, but there are few incentives for providers to use the methods suggested. Reasons for this are that most have not been tested on real world applications and the providers fear that such approaches may lead to breaches in SLAs with corresponding loss of customers [2].

Our proposed solution to this problem is to create a cloud data center infrastructure taking advantage of existing energy saving techniques. Our solution will feature servers that have had these techniques applied to them, while there will be a small number of servers operating as standard as an incentive for providers, by allowing high priority jobs to be completed quickly and by acting as a buffer in case of losing operating servers. An algorithm will be created to allocate incoming tasks to either the standard or energy efficient servers dynamically based on priority scheme.

The rest of this paper is structured as follows. In Section II, we discuss the existing literature and research in this area. In Section III, we expand on our planned architecture and explain our proposed approach. In section IV, we discuss our simulation setup. In Section V, we display the results we have found thus far and what results we expect from our final version. Finally, in Section VI, we present our conclusions found at this point and outline our planned future work.

II. LITERATURE REVIEW

Various studies have been carried out into improving the energy efficiency of cloud computing [1, 3-4], some of which mentioned in Table I. Among the most frequently suggested methods are sleep modes, dynamic speed scaling, DVFS, virtualization, resource allocation, virtual machine migration, green routing, and workload optimization [3]. Our research focuses on a combination of sleep modes and DVFS, as well as introducing a scheduling algorithm to help optimize the process.

Sleep modes, wherein servers can be put into low power states, are one of the most common approaches in order to reduce energy consumption. The basis for this approach is saving energy by powering down servers when they are not needed. By turning off idle servers within a data centre, energy consumption is reduced. However, continually switching servers on and off can cause a time delay and energy penalties. Testa et al. propose a controller in addition to IEEE 802.az Energy Efficient Ethernet to manage transitions between low powered and standard powered states [5]. Using traffic forecasting to manage sleep modes has been suggest by Morosi et al. [6]. Their forecasting based algorithm, forecasting based sleep mode algorithm (FBSMA), allows for daily calculations of approximate traffic.

Another common suggested method of improving energy efficiency is DVFS. This technique allows for the frequency of the CPU to be reduced in times when the CPU load is low, meaning less voltage of power can be consumed [7]. Meisner et al. propose the PowerNap system using DVFS in conjunction with dynamic power management [8]. The Power Aware List-based Scheduling and the Power Aware Task Clustering algorithms suggested by Wang et al use DVFS to propose that non-critical jobs can be run slowly over time to allow the CPUs' frequencies to be reduced [9].

There are various scheduling schemes suggested in research aiming to improve energy efficiency in data center servers. Dong et al. propose a scheduler that will select the most energy

 ISBN: 978-1-61208-460-2

TABLE I. TABLE OF ENERGY EFFICIENCY METHODS IN LITERATURE

Method	Uses			Ref
	Sleep Mode	DVFS	Scheduling	
IEEE 802.az Energy Efficient Ethernet	Yes	No	No	[5]
Forecasting Based Sleep Mode Algorithm	Yes	No	No	[6]
Dynamic Voltage/Frequency Scaling	No	Yes	No	[7]
PowerNap	Yes	Yes	No	[8]
PALS & PATC Algorithms	Yes	Yes	No	[9]
Most Energy Efficient Server First	No	No	Yes	[10]
Vacation Queueing Model	Yes	No	Yes	[11]
Separate Scheduling Algorithms	No	No	Yes	[12]
Intelligent scheduling with DVFS	Yes	Yes	Yes	[13]

efficient server first [10]. This method allocates tasks based on server energy profiles, and is shown to outperform random scheduling and least-allocated-server-first scheduling. Cheng et al. suggest a method that uses a vacation queueing model to model task schedule, then analyse task sojourn time and energy consumption of computation nodes to create algorithms [11]. Reddy and Chandan's method of using three processors to each run a different scheduling algorithm (earliest-deadline-first, earliest-deadline-late, and first-come-first-served) found energy savings when compared to existing stand-by sparing for periodic tasks, but their savings do not show in all their experiments [12]. Calheiros and Buyya propose a method that combines intelligent scheduling with DVFS for minimum frequency and power consumption, and for completion before user-defined deadline [13]. This approach focuses on urgent, CPU intensive tasks, and their results show between a 2% and 29% improvement on the baseline energy consumption [14].

III. PLANNED ARCHITECTURE

As mentioned previously, our proposed solution is to create a cloud data center infrastructure taking advantage of existing energy saving techniques. Within our infrastructure, we plan to have a combination of servers using energy efficiency techniques, along with servers that operate as normal as fall back to incentivize cloud service providers. Using the literature, we have decided to tackle the issue using a combination of sleep modes, DVFS, and energy efficient scheduling, however unlike the intelligent scheduling with DVFS method, our solution will be applied to all tasks as opposed to CPU intensive tasks only.

Our planned architecture will be made up of two sets of servers: standard servers, that is servers running as normal with no energy efficiency methods applied, and 'green' servers, that will be running using sleep modes and DVFS in order to reduce their energy consumption. A simple outline can be seen in Fig 2. Sleep modes will activate if the incoming traffic to the data center is light and a smaller number of servers can be utilized to handle the workload. DVFS will be used in much the same way, where the traffic allows for the frequency of the processers to be reduced, thereby lowering the energy consumption of the servers. In addition, the green servers will be running with lower processing speeds as we also believe that this will lead to reduced energy consumption.

Currently, we plan on having 90% of the servers within the data center operate using our green model. The final 10% will be running as normal. We believe that with most of the servers using our reduced energy method the energy savings should be sizeable. The remaining 10% will help to ensure that the QoS does not suffer. These values are just a starting point, from which we plan to experiment with different ratios to find the most beneficial setup.

In addition to the energy efficiency techniques applied to the servers, we plan to implement a scheduling algorithm that will direct tasks to the appropriate servers based on a priority scheme within the task's SLA requirements and maintain expected QoS rates for users. Outlined in Fig. 1, at a basic level we envision that tasks will have at least one of the following requirements:

- High priority requirement, meaning the task is of the highest urgency and to be completed as soon as possible with no regards to energy consumption.
- Time requirement, meaning the task can either be on a quick time limit or that it can be run slowly over time on a 'slow burn'.
- Energy requirement, meaning the task is to be as energy efficient as possible while meeting all its other SLA requirements.

The scheduling algorithm will use these requirements to decide which server the task should be assigned to as follows. As each task arrives on the network, information on whether it is a priority task, has a time limit, or has an energy limit will be found. If the task is a priority task then it is immediately assigned to the next available server in order to meet its priority requirements. If the task has a time limit, or can be run at a lower processing rate over a longer time period, then first the target completion time will be found. For each server currently on the network, its 95 percentile completion time will be found. The first server to return a completion time that is within range of the target completion time will be assigned the task. Finally, if the task has a preferred energy limit then the energy target will be found first. Following that, the 95 percentile energy consumption per task will be found, however unlike for the time limit, only low energy servers will be used. The task will be assigned to the first low energy server to return an energy consumption per task within range of the target consumption.

 ISBN: 978-1-61208-460-2

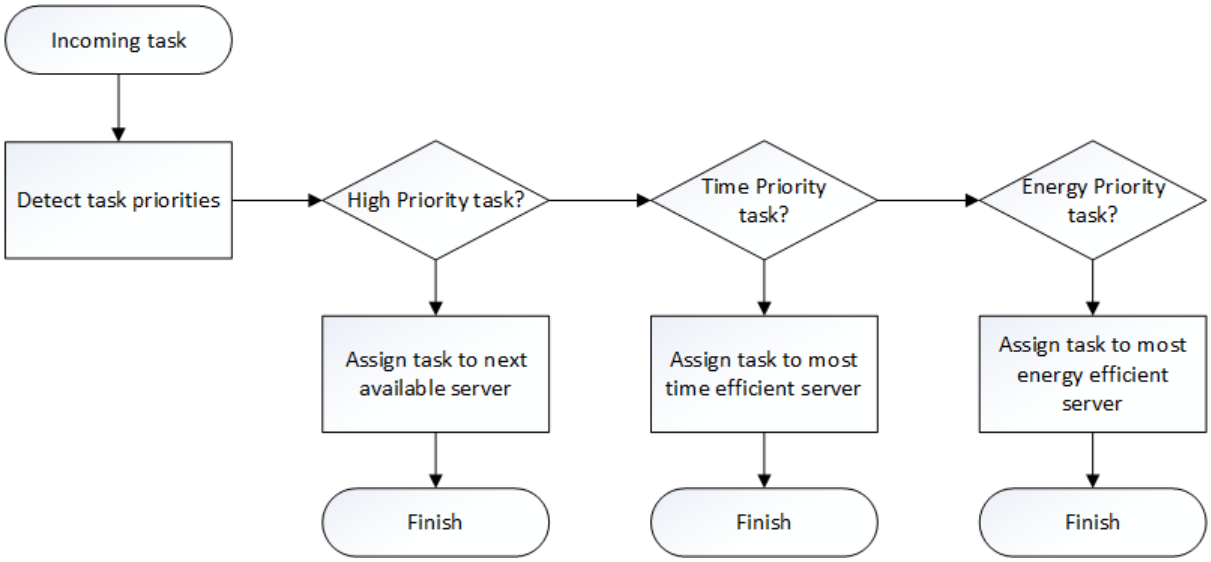


Figure 1. Flow chart of proposed algorithm functionality

IV. Simulation Setup

After testing out a range of simulation software, including CloudSim, DCSim, and iCanCloud [14], for our experiments we decided to use the Greencloud simulator [15]. Greencloud is a packet-level simulator for energy-aware cloud computing in data centers and an extension of the ns-2 simulation software.

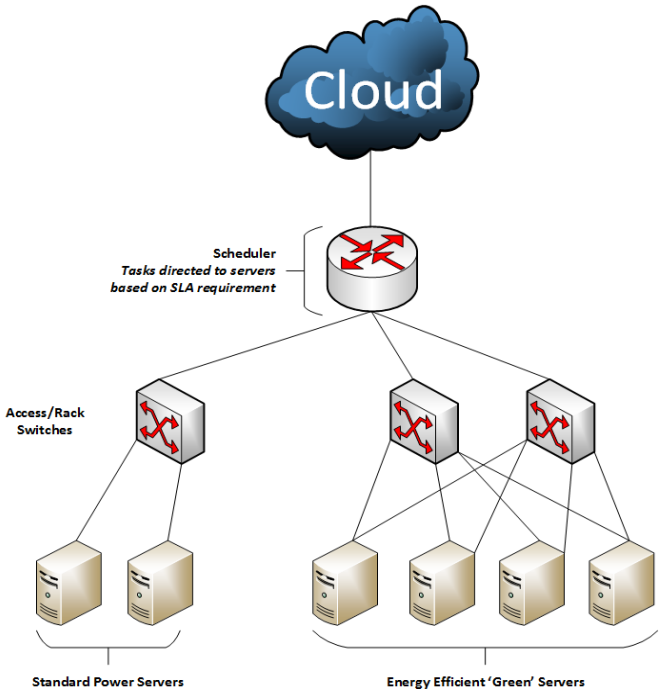


Figure 2. Outline of planned architecture.

We decided to use Greencloud as it is a simulator designed specifically for measuring the energy efficiency of cloud computing data centers. Greencloud uses a three-tier data center design, seen in Fig. 3, consisting of a core layer, an aggregation layer, and an access layer.

Currently our simulation setup consists of 144 servers, all of the same specification, with 1 core switch, 2 aggregation switches, and 3 access switches. The servers' specification contain commodity processors with 4 cores, 8GB of memory, and 500GB of disk storage, giving the data center a total of 576057600MIPS (Millions of Instructions Per Second) of processing power.

To examine QoS in our simulation, we are using round-trip time. We have self-imposed a limit of 2 seconds on single tasks. Tasks exceeding the 2 second limit are in breach of QoS, however we have also decided on a 5% trade-off, allowing for 5% of tasks within the experiments to exceed this limit without being in breach of SLAs.

V. Results

Using the simulator, we have tested the energy consumption of the data center when servers are run as normal and when DVFS and sleep modes have been applied. For this experiment, we used a random assignment scheduler and aimed or a data center workload of around 30%. This created 32689 tasks for the data center to process.

We found that using DVFS and sleep modes in a data center can reduce the energy consumption. As seen, the data center consumes 495.3 W*h using standard servers, with the servers consuming 332 W*h of that. In comparison, when DVFS and sleep modes are applied, the data center consumes 472.1 W*h, with the servers consuming 308.8 W*h. That is a percentage difference of 7.24% without any optimisation applied.

VI. Conclusions And Future Work

In this paper, we presented our idea to introduce a cloud data center infrastructure taking advantage of existing energy saving techniques combined with a priority scheme based scheduling algorithm in order to optimise the process and maintain QoS. We have suggested an approach wherein a major number of servers within the data center are run using

 ISBN: 978-1-61208-460-2

energy efficiency methods, such as DVFS and sleep modes, and a small minority are left to run as standard as an incentive to cloud service providers. We outlined the structure for our scheduling algorithm based upon SLA requirements and how we think it can improve the energy efficiency of cloud data centers.

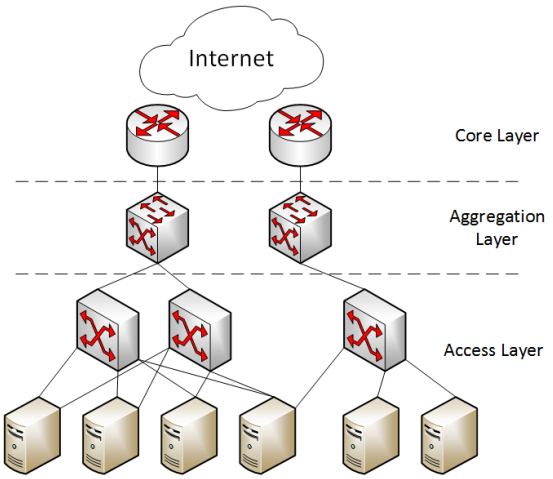


Figure 3. Example of a simple three-tier data centre architecture in Greencloud.

The results from our experiments thus far have shown that by using DVFS and sleep modes in cloud data centers the energy consumption of the servers can be reduced, and that there is still room for improvement, and we aim for our scheduling algorithm to help do this.

In the future, we aim to accomplish several key issues. Firstly, we will expand our simulation model to create our two distinct types of servers. Using our original 144 servers, approximately 10% will be left to run using the initial specifications with no energy efficiency methods applied. The remaining 90% will be run with slower processing speeds, as well as DVFS and sleep modes to allow for the servers to be placed into a lower power state then the incoming traffic is low.

The next step in our planned work is to amend our simulated tasks with our new SLA priority scheme. Tasks will be assigned at least one of the three priority requirements discussed previously. Once the priority scheme is in place we will work to implement our scheduling algorithm that will assign incoming tasks to the relevant servers based upon the priority scheme. Using this algorithm we hope to find that the energy efficiency methods enacted can be optimised to achieve the best energy reduction results possible while maintaining the QoS through measuring task completion time, failed tasks, and dropped packets.

Finally, as another incentive for cloud service providers, we are aiming to create realistic simulated workloads based on Google cluster data that was released in 2011 [16]. This data has been collected over a period of three months in one of Google's cloud data centers and offers real world information that can be adapted for our simulations. We hope that the inclusion of this data in our workload generation will show how our optimised approach would react if placed in a real world situation. This would act as another incentive for cloud service providers, along with the standard powered servers, to implement more 'green' energy applications.

REFERENCES

[1] A. Hameed et al., "A Survey and Taxonomy on Energy Efficient Resource Allocation Techniques for Cloud Computing Systems," in Computing, 2014, pp. 1–24.

[2] Y. Georgiou, D. Glesser, K. Rzadca, and D. Trystram, "A Scheduler-Level Incentive Mechanism for Energy Efficiency in HPC," in 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2015, pp. 617–626.

[3] A. Kulseitova and A. T. Fong, "A Survey of Energy-Efficient Techniques in Cloud Data Centers," in 2013 International Conference on ICT for Smart Society (ICISS), 2013, pp. 1–5.

[4] G. L. Valentini et al., "An overview of energy efficiency techniques in cluster computing systems", Cluster Computing, vol. 16, no. 1, pp 3-15, 2013.

[5] P. Testa, A. Germoni, and M. Listanti, "QoS-aware sleep mode controller in 'Energy Efficient Ethernet,'" in 2012 IEEE Global Communications Conference (GLOBECOM), 2012, pp. 3455–3459.

[6] S. Morosi, P. Piunti, and E. Del Re, "Sleep mode management in cellular networks: a traffic based technique enabling energy saving," Trans. Emerg. Telecommun. Technol., vol. 24, no. 3, pp. 331–341, 2013.

[7] A. Hammadi and L. Mhamdi, "A survey on architectures and energy efficiency in Data Center Networks," Comput. Commun., vol. 40, no. 1, pp. 1–21, Dec. 2013.

[8] D. Meisner, B. T. Gold, and T. F. Wenisch, "The PowerNap Server Architecture," ACM Trans. Comput. Syst., vol. 29, no. 1, pp. 1–24, Feb. 2011.

[9] L. Wang et al. , "Energy-aware parallel task scheduling in a cluster," Futur. Gener. Comput. Syst., vol. 29, no. 7, pp. 1661–1670, Sep. 2013.

[10] Z. Dong, N. Liu, and R. Rojas-Cessa, "Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers," J. Cloud Comput., vol. 4, no. 1, pp. 1-14, 2015.

[11] C. Cheng, J. Li, and Y. Wang, "An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing," Tsinghua Sci. Technol., vol. 20, no. 1, pp. 28–39, 2015.

[12] H. K. S. Reddy and S. P. Chandan, "Energy aware scheduling of real-time and non real-time tasks on cloud processors (Green Cloud Computing)," in 2014 International Conference on Information Communication and Embedded Systems (ICICES), 2014, no. 978, pp. 1–5.

[13] R. N. Calheiros and R. Buyya, "Energy-Efficient Scheduling of Urgent Bag-of-Tasks Applications in Clouds through DVFS," in 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, 2014, pp. 342–349.

[14] W. Zhao, Y. Peng, F. Xie, and Z. Dai, "Modeling and simulation of cloud computing: A review," in Proceedings - 2012 IEEE Asia Pacific Cloud Computing Congress, APCloudCC 2012, 2012, pp. 20–24.

[15] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: A packet-level simulator of energy-aware cloud computing data centers," J. Supercomput., vol. 62, pp. 1263–1283, 2012.

[16] C. Reiss, J. Wilkes, and J. J. L. Hellerstein, "Google cluster-usage traces: format+ schema," 2011.

 ISBN: 978-1-61208-460-2

Model Driven Framework for the Configuration and the Deployment of Applications in the Cloud

Hiba Alili, Rim Drira and Henda Hajjami Ben Ghezala
RIADI Laboratory, National School of Computer Sciences,
University of Manouba, 2010 la Manouba, Tunisia
E-mail: {alilihiba,drirarim,hhbg.hhbg}@gmail.com

***Abstract*—Cloud computing offers a distributed computing environment where applications can be deployed and managed. Many companies are seeing substantial interest to extend their technical infrastructure by adopting cloud infrastructures. Although the choice of such an environment may seem advantageous, users are faced with many challenges, especially with regard to deployment and migration of applications in the Cloud. To address some of these challenges, we propose a new approach based on model-driven engineering techniques (MDE), called MoDAC-Deploy, for the assistance to the configuration and the deployment of applications in the Cloud. This paper focuses on the design and the implementation of our approach. In fact, we developed a model-driven Framework with generative mechanisms to simplify and to automate cloud services deployment process, to overcome APIs heterogeneity, to minimize the vendor lock-in and to enable application portability among different cloud infrastructures by reusing configurations/deployments "Model a configuration once and deploy it anywhere". We conducted also a case study in order to validate our proposed approach. Our empirical results demonstrate the effectiveness of our MDE Framework to seamlessly deploy services in the cloud and to migrate easily between different Cloud Service Providers (CSPs) without any programming efforts.**

Keywords-Deployment; Cloud Computing; Model Driven Engineering.

I. INTRODUCTION

Cloud Computing is a paradigm shift that involves dynamic provisioning of shared computing resources on demand. It is a pay-as-you-use billing model that offers computing resources as a service in an attempt to reduce IT capital and operating expenditures [1].Particularly, Infrastructure as a Service (IaaS) allows users to allocate computational, storage and networking resources from Cloud Service Providers (CSPs). It offers to users the ability to customize the environment to suit their applications and even it supports the deployment of legacy applications without any modification in their source code. In order to make efficient use of such an environment, tools are needed to automatically deploy, configure and run services in a repeatable way. In this context, we focus in this paper on the deployment of applications in IaaS environment.

Deploying applications in cloud infrastructures is not a trivial task, as it relies on handcrafted scripts and it requires increased complexity and additional effort. Doing so is time consuming and error prone, especially for deployments with a large number of nodes. Moreover, the growing trend towards migrating applications and services to the cloud has led to the emergence of different CSPs, in turn leading to different specifications of provided resources and to heterogeneous APIs. These challenges make it hard for cloud customers to seamlessly transition their services to the cloud or migrate between different CSPs. These challenges can be classified into three main categories:

- Deployment Complexity: the deployment in the cloud is a very complex process given the large number of operations required to finish with a successful deployment (e.g., the restructuring of each application layer for the cloud, the auto-scaling of services, the monitoring and the optimization of the application services to take advantage of the cloud benefits) [2]. In fact, to successfully deploy an application in the cloud, a good preparation of the target environment is essential to be compatible with its architecture.
- Programming and Deployment Heterogeneity: CSPs such as Amazon Web Services [3], Google Cloud Platform [4], Rackspace, and Microsoft Azure [5] provide different APIs to their customers to manage their resources on the cloud, which is often carried out programmatically using this APIs. This API heterogeneity imposes a steep learning curve for cloud customers [6]. To overcome this concern, CSPs often provide a web-based management console. Unfortunately, these user interfaces are very specific to the CSP and hence do not resolve the original problem.
- Vendor lock-in and Portability: The fear of vendor lock-in is often cited as a major impediment to cloud service adoption. In fact, the proprietary APIs provided by each CSP are incompatible with those of other CSPs and as a result it limits the ability of cloud customers to seamlessly migrate their services between different CSPs. For that reason, many customers stay with a provider that doesnt meet their needs, just to avoid the cumbersome process.

Addressing these challenges requires a framework that holistically focuses on the core set of the deployment problems. In parallel, MDE has emerged as a software engineering paradigm for dealing with the problem of system interoperability and portability across different execution platforms. Model Driven Architecture (MDA) does this separating business and technical concerns and proposing techniques to integrate them. In addition, MDA techniques allow generating automatically code from models. Thus, we believe that MDE techniques are promising to address the challenges outlined above (automating the deployment process and ensuring the portability across different cloud infrastructures).

In this context, we propose in this paper an intuitive abstraction, based on MDE standards, to cloud customers to model software deployment in the cloud and to enable various CSP-agnostic. This abstraction is realized as a modeling tool based on a domain specific modeling language (DSML) with

 ISBN: 978-1-61208-460-2

generative capabilities. Our proposal, which we call MoDAC-Deploy (Model Driven Framework for the Assistance to Cloud Deployment), includes three key artifacts: (1) IaaSEditor, a modeling tool which provides an intuitive user interface that allows cloud customers to define the deployment model of their applications in the cloud. It presents an abstraction layer isolating applications from the underlying cloud provider and hiding APIs. This tool is based on (2) IaaSMetaModel, a meta-model that captures all the concepts needed to specify an accurate cloud deployment. And finally (3) ScriptGenerator, a generative tool that concludes automatically the deployment script from the deployment model created within IaaSEditor. Our framework shields cloud users from having to manually write scripts using low-level APIs and enables applicaion portability among different CSPs.

This paper is organized as follows: Section 2 briefly discusses scientific works closely related to ours. In Section 3 , we introduce the MDE basis, especially we describe the MDA process. Section 4 describes our model driven framework for the configuration and the deployment of applications in the cloud. In Section 5, we illustrate a case study to evaluate and to demonstrate the effectiveness of our deployment framework and finally, Section 6 provides concluding remarks and outlines future works.

II. Related works

Our work has taken shape in the context of a rich literature focused on simplifying the deployment of applications in the cloud. In fact, several works have shown an interest to automate the deployment process and to deal with API heterogeneity. We propose in this section to analyze the state of the art about software deployment and identifying the good practices to be reused in our own solution.

Juve and al. [7] have developed a system called Wrangler to provision, configure and manage virtual machine deployments in the cloud. Wrangler allows users to specify the layout of their application declaratively using an eXtensible Mark-up Langage (XML) format and then to send this deployment description to a web service that manages the provisioning of virtual machines, the installation and the configuration of software and services. This system is able to interface with different resource providers,as it currently supports only Amazon EC2 [8], Eucalyptus [9] and OpenNebula [10]. But authors haven't talk about the possibility to extend this system in order to support other CSPs. While our solution is designed specifically to support multiple CSPs and to easily add new cloud artifacts. Our approach intends also to completely shield software designers from any programming efforts contrary to Wrangler that requires the preparation of an XML description of the deployment model.

Caglar and al. [11] have proposed a solution based on MDE, including a domain-specific modeling language (DSML) for automating deployment of applications in the cloud and generative technologies. The meta-model of the deployment model in this DSML was designed in order to overcome the challenges resulting from heterogeneity in CSP APIs and deployment policies. It consists of Print, Sleep, Upload, Download, RunApp, Terminate, CreaeInstance, WaitforStartup, Connect, Entity, and Keyfile model components, which are used during the deployment process. Connections between components are also defined in the meta-model. The interpretation of the created deployment model generates the appropriate deployment script in Python, which contains and execute the deployment steps. Just like Wrangler [11], this work is limited to VM management, while our work supports also storage and network connectivity management. In addition, it allows users to specify resources into groups (availability group, security group and auto-scaling group).

In [12], the authors describe their automatic deployment platform that they developed for the Microsoft Azure cloud, driven by the need of a chemistry application performing Quantitative Structure-Activity Relationship (QSAR) analysis. The main goal was to enable the execution of existing non-.Net software in the Azure infrastructure which was designed only for applications based on the .Net framework, and which supports a specific, queue-based software architecture. By using the proposed deployment framework, the QSAR application was successfully running in the Azure infrastructure. However, this solution is dedicated only to the Azure cloud and it needs to be generalized.

Shekhar and al. [13] have proposed a framework for conducting price/performance tradeoffs in executing MapReduce jobs at various CSPs, selecting the best option and deploying and executing the job on the selected CSP infrastructure. All of these capabilities are driven by an MDE framework. However, the MDE abstractions are being developed and the realization as a web-hosted service is still under development. While this efforts is promising, they need to be tested and evaluated.

Other recent efforts like Deltacloud [14], Libcloud [15] and jclouds [16] have been developed to deal with API heterogeneity. These libraries hide away differences among multiple cloud provider APIs and allow users to manage different cloud resources through a unified common API. This has solved the multi-cloud problem in a very detailed manner, but the complexity is therefore even larger (i.e., users need to learn how to program using these APIs).

A model-driven approach for automating cloud deployment is also presented in [17]. Hamdaqa et al. have proposed a (5+1) architectural view model, where each view corresponds to a different perspective on cloud application deployment. This view model enables cloud stakeholders (e.g., providers, developers, administrators and financial managers) to leverage cloud platform capabilities. The (5+1) view model has been realized as a layered, domain specific modeling language (DSML), called StartusML, and the capabilities of this language have been illustrated using a representative domain example. The model was derived by investigating the process of architecting cloud applications, and then providing a set of meta-models to describe cloud applications within their ecosystem: an availability meta-model, an adaptation meta-model, a performance meta-model, a service meta-model, a workflow meta-model and finally a provider meta-model. Each meta-model in the (5+1) view model is dedicated a layer in StratusML. Our work has synergies with this work in the context of providing a user interface in order to facilitate the configuration and the description of the deployment model of applications in the cloud. Our deployment framework presents a fairly comprehensive DSML that allows the users to describe their applications deployment architecture in terms of services and interactions. It clarifies the cloud service model and its requirements in terms most cloud customers would understand. We developed also generative technologies to automate the de-

 ISBN: 978-1-61208-460-2

ployment process and to resolve the problem of the repetition of tedious tasks.

In the reminder of this paper, we will focus on presenting and evaluating our proposed contribution with respect to related work.

III. DEFINITIONS

This section gives a short overview of Model-Driven Engineering and its related concepts.

A. Model Driven Engineering

MDE is becoming an emergent software engineering paradigm to specify, develop and maintain software systems. In MDE, models are the primary artifact of the engineering process and are used, for instance, to (semi)automatically generate the implementation of the final software system.

According to the Object Management Group [6] MDE is a specific approach to software engineering that defines a theoretical framework for generating a code from models using successive model transformations [18]. The main goal of this approach is to separate the business side of a system from its implementation. The business model of a system can therefore drive its implementations on different platforms. In this way, we can expect to obtain better coherence between implementation and interoperability.

In brief, MDE aims to raise the level of abstraction in program specification and increase automation in program development. The best-known MDE initiative is the MDA proposed by the OMG [19].

B. Model Driven Architecture

MDA states that it models the environment and the requirements for a system in a Computational Independent Model (CIM). A CIM does not show the details of system structure. Thus, a CIM can be used to build a Platform Independent Model (PIM). A PIM focuses on the operation of the system while hiding details related to the use of a particular platform. PIM maintains platform independence in order to be suitable for use with different platforms. The transformation of a PIM into a Platform Specific Model (PSM) is based on the associated Platform Model (PM). A PSM is a system model for a specific platform. It combines PIM specifications with the details that specify how that system uses a particular platform. Figure 1 shows the main concepts used in MDA.

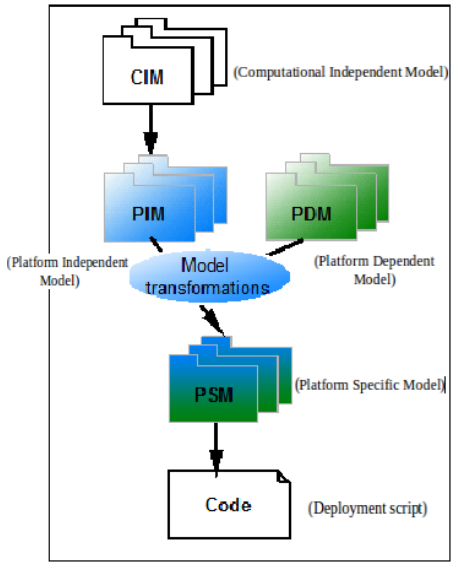


Figure 1. Main concepts of the MDA approach

C. MDE for the Cloud deployment

Considerable attention has been focused recently on MDE as an alternative solution to overcome some of the deployment concerns in the cloud. In fact, the MDE approach holds promise in:

1) **Simplifying and (semi)automating the process of deployment of applications in the cloud**, by creating specific modeling languages/ tools that hide development complexity while also significantly reducing the learning curve involved in moving to a cloud platform. They allow accurate descriptions with a semantic precision.
2) **Ensuring portability and interoperability of systems across different platforms**, by developing generic and extensible cloud artifacts. This shields the users from the variabilities in CSPs.

In the literature, a number of recent papers have already explored this possibility as StartusML [17] and works done in [11][12][13]. Nevertheless, all of these works have focused only on resolving the heterogeneity problem. Our current research focuses on resolving all of the challenges mentioned above in Section 1 and on providing a complete solution for an automated deployment in the cloud.

IV. MODAC-DEPLOY: A MODEL DRIVEN FRAMEWORK FOR THE ASSISTANCE TO CLOUD DEPLOYMENT

Our goal is to provide a complete solution that assists software designers to configure and to deploy successfully their applications in the Cloud. In this section, we present more details about the MoDAC-Deploy architecture, giving the main steps required for deploying an application in the cloud and its capabilities.

A. Overview

The key idea of the MoDAC-Deploy framework is to simplify as much as possible the deployment process in the cloud by proposing an abstraction layer isolating applications from the underlying environment and hiding API details. In fact, shielding users from having to manually write scripts using low-level APIs hides the deployment complexity and dramatically reduces manual efforts and the time required to configure cloud resources. In addition, our framework has been designed specifically to support multiple CSPs in order to enable application portability among different CSPs. So applications can be easily moved from one cloud infrastructure to another which would satisfy more their needs without any additional efforts: ”model the software deployment once and deploy it anywhere”. In fact, users have only to change the selected provider from the available list presented in our framework and then reuse the same deployment model to

 ISBN: 978-1-61208-460-2

deploy their applications in the new chosen cloud infrastructure. Figure 2 shows our framework architecture. It includes three main modeling tools: *IaaSEditor*, *IaaSMetaModel* and *ScriptGenerator*. We have used Eclipse Modeling Framework to develop the DSML and the generative capabilities within our framework.

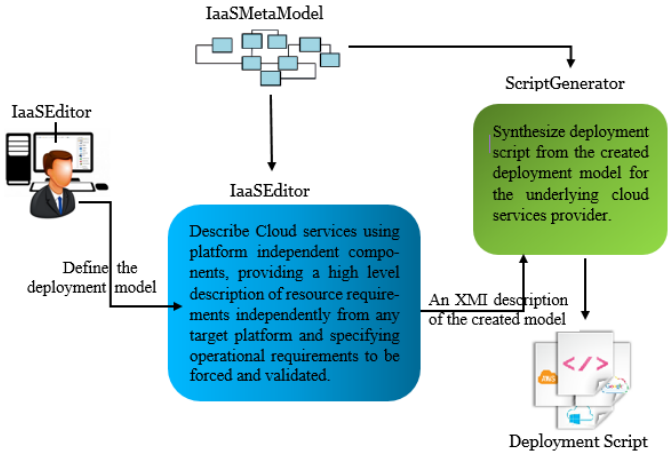


Figure 2. Overview of the MoDAC-Deploy approach

Both *IaaSEditor* and *ScriptGenerator* use the deployment meta-model *IaaSMetaModel* in order to guarantee the creation of a valid model and the generation of an executable deployment script.

B. Deployment process

Following MDA practices, an application deployment is achieved in three-step process:

First, users specify and define the hosting architecture of their applications through creating a new deployment model under *IaaSEditor*. The deployment model is saved as an XMI description consisting of several nodes (components). Each node may correspond to a virtual machine or to a storage medium, then it would be associated to a named group, namely a *SecurityGroup*, an *AutoScaling-Group* or an *AvailibilityGroup*. This deployment model is defined using the meta-model IaaSMetaModel. A simple validation should be done at this level to guarantee a valid deployment model and then an executable deployment script.

Second, a model transformation engine with specific rules is used to transform the preceding model into a CSP-specific model.

Finally, *ScriptGenerator* ensures the generation of the deployment script script.sh from the XMI document generated within *IaaSEditor*. The generated script presents an executable bash script which should be token later and executed on the command line interface of the underlying provider. Users have access to the generated script and they can identify possible values as wanted. Figure 3 illustrates the process that the MoDAC-Deploy framework goes through to facilitate and semi-automate the deployment into cloud infrastructures.

C. IaaSMetaModel

IaaSMetaModel is depicted in Figure 4. This meta-model captures all the concepts that are needed to specify a cloud

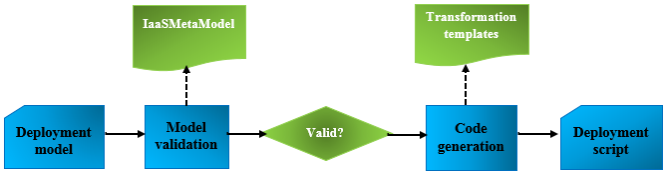


Figure 3. Deployment Process

deployment. It comprises multiple extensible and customizable classes, all of these classes are generic cloud artifacts, describing the functioning and the dependence of the different components and application services to be deployed on the cloud. The meta-model components and their responsibilities are as follows:

- *Hosting_Architecture*: is the main class of our deployment meta-model. It presents the hosting architecture of the application to be deployed in the cloud. Through this class, users can specify the cloud provider and their authentication credentials required by the provider. A good design of the hosting architecture is essential to have a successful deployment.
- *Application*: presents the name of the application to be deployed, its version, the URL designated by the developer to access to this application. It contains also entities named File such as text file, executable file, or any other library files to be uploaded onto the VMs that it is connected to. The application's set up and log files are copied from a local directory to another directory on a VM in the cloud.
- *VirtualMachine*: is used to define requested VMs from clients and to specify their characteristics such as the image ID, the VM size, the availability zone and the number of instances required to execute the application in the cloud. Through this class, we can also enable the monitoring of our instances VM.
- *StorageMedium*: we classify the storage mediums into three categories: *DataBaseStorage*, *SimpleStorage* and *VolumeStorage*. *DataBaseStorage* offers both relational and NoSQL database infrastructure. *SimpleStorage* provides a persistent storage of large amounts of distributed object, highly scalable, sustainable and available while *VolumeStorage* provides disk support, we can associate multiple disks to a virtual machine.
- *Group*: designates a collection of virtual machines or storage mediums with common characteristics, we distinguish between three group categories: *AvailibilityGroup*, *AutoScalingGroup* and *SecurityGroup*. A *SecurityGroup* consists of a set of access control rules that describe traffic filters to our VMs. It is analogous to an inbound network firewall, for which we specify the protocols, ports, and source IPs ranges that are allowed to reach the VM instances. An *AutoScalingGroup* presents scaling factors to apply on a set of virtual machines. The number of running VM instances can be dynamically scaled out and in, according to certain conditions in order to handle changes in traffic: it is possible to increase the size of a group of instances to meet a load peak or to

 ISBN: 978-1-61208-460-2

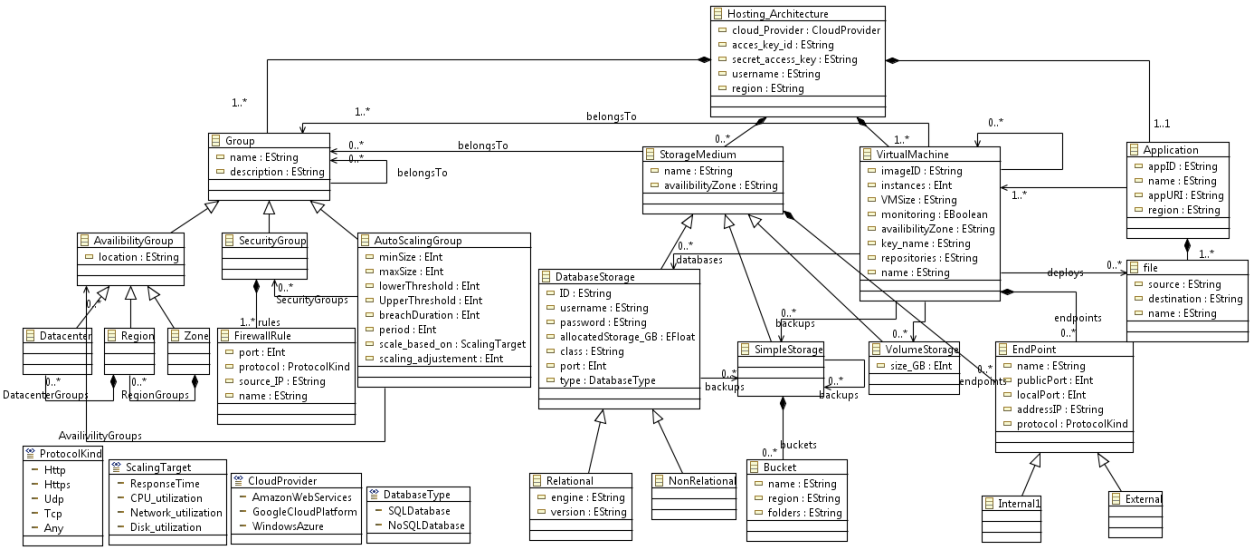


Figure 4. IaaSMetaModel

reduce the executing park in the case of the decrease of traffic. Finally, an *AvailibilityGroup* nests components which need to be hosted in the same location. It is a superclass for the three geolocation groups (i.e., Zone Region and Datacenter).

- *EndPoint*: handles incoming network traffic to cloud components. An endpoint is a URL that is the entry point for a web service. For each endpoint, we associate a range of IP addresses and the port through which a cloud component/ task can connect to others. An endpoint uses a specific protocol that determine the syntax and semantics of the messages that are exchanged between the two communication parties. An endpoint can be external if it is publicly visible or internal if it is only accessible within the cloud application. Also, each endpoint has a public port and a private port: the public port is used to listen for incoming traffic to the virtual machine from the Internet while the private port is used to listen for incoming traffic, typically destined to an application or service running on the virtual machine.

This meta-model was developed after inspecting manually three cloud infrastructures, namely Amazon Web Services, Windows Azure and Google Cloud Platform. Furthermore, adding additional provider concepts is designed to be relatively simple.

D. IaaSEditor

As described above, IaaSEditor provides an intuitive user interface that allows cloud customers to define their application services and to configure the target environment through a simple graphical modeling, shielding them from programming efforts. Once users have created the deployment model of their applications, they can choose any CSP supported by our framework and the created deployment model can then be reused to move the application into another Cloud infrastructure by changing only the selected CSP under IaaSEditor and some CSP-specific properties such as the VM image ID.

Figure 5 presents the editor *IaaSEditor*. it is composed of three layouts:

- **Design workspace :** Here users can design and validate their deployment models, and ask for the generation of the script deployment.
- **Palette :** it contains the different components to use in creating the deployment model, grouped together in different categories (Resources, Groups, Relations) according to their role.
- **Configuration Tabs :** Each tab opens a view that displays the properties of the selected element in the design workspace. These properties can be edited to change or set the parameters related to a particular component.

E. ScriptGenerator

We have used Acceleo, a code generation tool under the framework Eclipse, to implement our *ScriptGenerator*.

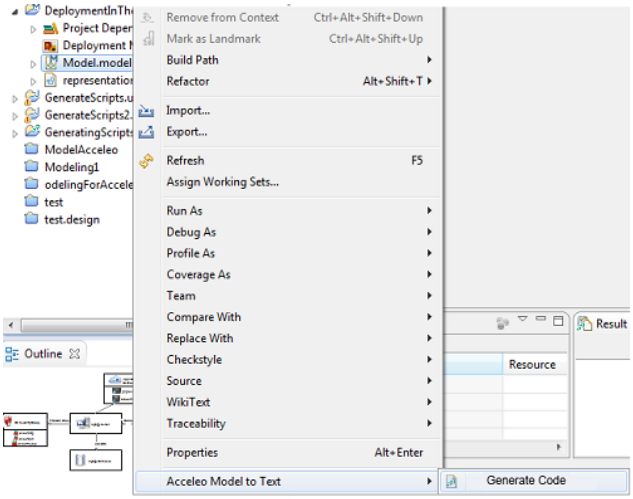


Figure 5. Generation of the deployment script

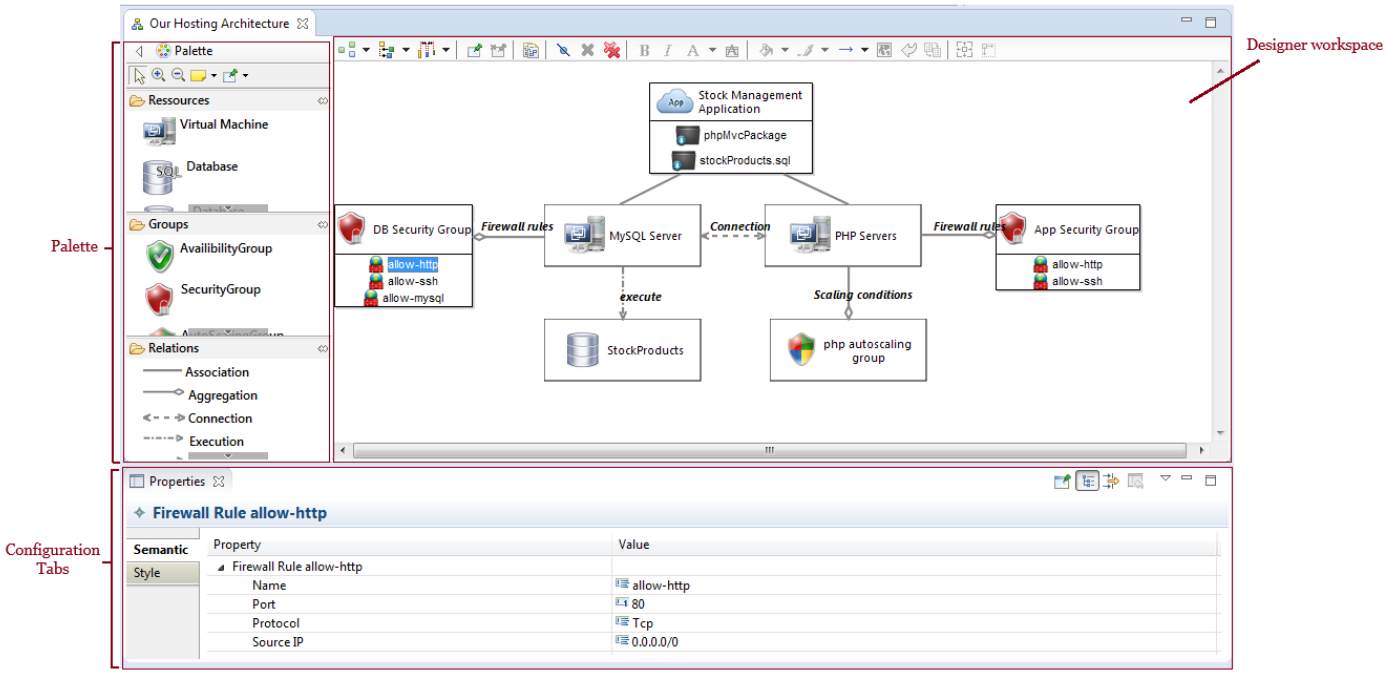


Figure 6. IaaSEditor

Users launch the generation of the deployment script with a simple click as shown in Figure 5.

Through a set of model transformation templates, *Script-Generator* synthesizes the deployment script from the deployment model created within the editor *IaaSEditor* for the underlying CSP (cf. Figure 6). In fact, templates convert data from the input model into the deployment script to configure cloud environment and to deploy the application.

In this Section, we described how the modeling capabilities are used to implement our framework and how the developed modeling tools can help to facilitate the deployment of services into cloud infrastructures. In the next Section, we will illustrate a case study in order to evaluate our deployment framework and to demonstrate its effectiveness.

V. CASE STUDY

We consider here a stock management application, sufficient to demonstrate the effectiveness of our deployment solution. we deployed this application across two different cloud infrastructures: Amazon Web Services (AWS) and Google Cloud Platform in order to underline the deployment portability among different CSPs.

To provision and to configure all necessary infrastructure, we need first to design the application architecture in order to ensure that it meets our requirements. This application is structured into logical tiers. The first tier is the web browser, which is responsible for presenting the user interface. The middle tier is an application server, which is responsible for the application's functionality. The third tier is a database server, which is responsible for data storage.

To this end, we defined two virtual machines and the number of instances to create from each one, one instance for MySQL (the database server) and 3 instances for Apache/PHP (application servers). In fact, we decided to deploy the MySQL database on an independent machine to properly manage the scalability of the application. Then we created a security group for each virtual machine to control and to filter the traffic allowed to reach the instances and we specified rules to each security group. In our case, we enabled inbound HTTP access from anywhere and inbound SSH traffic from our computer's public IP address so that we can connect to our instances. MySQL port was only opened for the Apache/PHP instances. In addition, an *AutoScalingGroup* is associated to PHP Servers in order to launch or terminate instances as demand on the application increases or decreases. So we configured auto-scaling to launch an additional Apache/PHP instance whenever CPU usage exceeds 60 percent for ten minutes and to terminate an instance whenever CPU usage under 30 percent. Every new added instance connect to the same MySQL database. Once our system was setup and configured, we installed needed repositories such as Apache 2 and PHP and we deployed our code to the application servers (PHP servers) and finally we associated a domain name with our web application. The final deployment model is presented in Figure 6.

Thanks to MoDAC-Deploy, all these steps are simply conducted by drag and drop operations and by filling properties in the tab "Configuration Tabs".

Figure 7 depicts the deployment script of the underlying application in AWS. The captured lines of code creates a MySQL instance and defines the characteristics of the virtual machine to be provisioned. We chose a linux image-32 bits for the MySQL server. It creates also a Key Pair, which presents the credentials we used to SSH into the box. Then, we opened the SSH port (22) and the HTTP port (80) for the

 ISBN: 978-1-61208-460-2

MySQL server. And finally, we imported the database backup file "stockProduits.sql" and we started the MySQL instance.

```
...
#Create a Key Pair for the VM MySQL Server :
aws ec2 create-key-pair --key-name key-pair-mysql-server --query 'KeyMaterial' --output text
    | out-file -encoding ascii -filepath key-pair-mysql-server.pem
chmod 400 key-pair-mysql-server.pem
#Create a Security Group for the VM MySQL Server:
aws ec2 create-security-group --group-name db-security-group-security-group
#Add  firewall rules:
aws ec2 authorize-security-group-ingress --group-name  db-security-group --protocol Tcp --
    port 80 --cidr 0.0.0.0/0
aws ec2 authorize-security-group-ingress --group-name  db-security-group --protocol Tcp --
    port 22 --cidr 0.0.0.0/0
#Create and Start the VM instances  "MySQL Server":
NUMBER_OF_INSTANCES=1
for i in $(seq 1 $NUMBER_OF_INSTANCES)
do
#Start an instance and Get her PublicDnsName:
PublicDnsName=$i=`aws ec2 run-instances --image-id ami-0c87ad78 --count 1 --instance-type 1
    --region us-east-1 --key-name key-pair-mysql-server.pem \
...security-groups  | grep PublicDnsName | awk -F": " '{print $2}' | sed s/\"//g |sed s/\,//g
```

Figure 7. The deployment script in Amazon Web Services

We reused the same deployment model to move the management stock application into the Google Cloud Platform. All what we did is to change the CSP as illustrated in Figure 8.

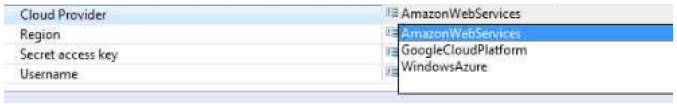


Figure 8. Cloud Services Provider Selection

We modified also the VM imageID from "ami-0c87ad78" in AWS to "https://www.googleapis.com/compute/v1/projects/ubuntu-os-cloud/global/images/ubuntu-1404-trusty-v20150316 " in Google Cloud Platform.

```
#!/bin/bash
#Download and install gcloud:
curl https://sdk.cloud.google.com | bash
sudo apt-get install googlecl
#Authenticate gcloud:
gcloud auth login
gcloud config set project application-de-gestion-de-stock
gcloud config set compute/region us-central1
# Virtual Machine:
#Create and Start the VM instances  "MySQL Server":
NUMBER_OF_INSTANCES=1
for i in $(seq 1 $NUMBER_OF_INSTANCES)
do
gcloud compute instances create mysql-server-$i --image https://www.googleapis.com/compute/v1
    /projects/ubuntu-os-cloud/global/images/ubuntu-1404-trusty-v20150316 \
--machine-type f1.micro  --zone us-central1-a --boot-disk-type "pd-standard"  \ ...
```

Figure 9. The deployment script in Google Cloud Platform

The generated deployment script is depicted in Figure 9.

VI. CONCLUSION AND FUTURE WORK

This paper presented the results of investigations on the main challenges of the deployment of applications in the cloud and on the modeling capabilities that can help to implement our proposed solution.

So, we have developed a model-driven framework for cloud deployment, which facilitates and semi-automates the deployment of services to cloud infrastructures. This reduces the deployment complexity and errors that can occur during manual configurations as well as costs. It helps also to shield users from complex programming efforts, the low- level API details and from the heterogeneity in cloud providers. In addition, our solution enables application portability between different clouds and allows to minimize the vendor lock-in. Generated script can be executed only on unix machine, we are currently working on generating deployment scripts running on windows.

As a minority of providers that offer autoscaling capabilities to automatically add or remove virtual machines from an instance group based on increases or decreases in load as Amazon Windows Azure (within an Auto Scaling group), Google Cloud Platform (i.e., define the autoscaling policy and the autoscaler performs automatic scaling) and Windows Azure (through configuring the autoscale status), our next research thread will definitely revolve around this feature, we plan to develop algorithms and techniques for dynamically scaling deployments in response to application demand in other IaaS and for re-configuring deployments. Cloud computing comes with a cost where the accounting is based on a utility model. Making decisions on how many cloud resources to use to host a service, and when and how much to autoscale is a significant challenge for the cloud customers. Understanding what will the impact of these decisions be on both the expected performance delivered to the service and cost incurred by the customer is even harder. In that context, developing mechanisms for estimating deployment performance and cost and selecting the proper cloud deployment is an issue to be addressed in ongoing work.

Besides, the current capabilities presented by this framework can be extended further to handle complex architectures such as network applications by adding new cloud artifacts and why not make it able to deploy multi-cloud architectures (i.e., deploying applications across multiple cloud providers, e.g., deploy a single workload on one provider, with a backup on another). We plan also to move in the direction of making the deployment DSML as mature and complete by covering new CSPs as well as private IaaS.

REFERENCES

[1] M. Hamdaqa, T. Livogiannis, and L. Tahvildari, "A reference model for developing cloud applications," in CLOSER, 2011, pp. 98–103.

[2] R. Gadhgadhi, M. Cheriet, A. Kanso, and S. Khazri, "Openicra: Towards a generic model for automatic deployment and hosting of applications in the cloud," in IJ-CLOSER, 2013, pp. 249–275.

[3] Amazon Web Services, http://aws.amazon.com, [Accessed 09 November 2015].

[4] Google Cloud Platform, http://cloud.google.com, [Accessed 24 October 2015].

[5] Windows Azure, http://azure.microsoft.com, [Accessed 04 November 2015].

[6] OMG, Object Management Group, http://www.omg.org.

[7] G. Juve and E. Deelman, "Automating application deployment in infrastructure clouds," in CLOUDCOM '11, pp. 658–665.

[8] Amazon, Elastic Compute Cloud (EC2), http://aws.amazon.com/ec2, [Accessed 09 November 2015].

[9] D. Nurmi et al., "The eucalyptus open-source cloud-computing system," in CCGRID '09, 2009, pp. 124–131.

[10] OpenNebula, http://www.opennebula.org, [Accessed 27 October 2015].

[11] F. Caglar, K. An, S. Shekhar, and A. Gokhale, "Model-driven performance estimation, deployment, and resource management for cloud-hosted services," in DSM '13, 2013, pp. 21–26.

[12] J. Cala and P. Watson, "Automatic software deployment in the azure cloud," in Distributed Applications and Interoperable Systems, 2010, vol. 6115, pp. 155–168.

 ISBN: 978-1-61208-460-2

[13] S. Shekhar et al., “A model-driven approach for price/performance tradeoffs in cloud-based mapreduce application deployment,” in MODELS, 2013, pp. 37–42.

[14] Deltacloud, https://deltacloud.apache.org/, [Accessed 18 September 2015].

[15] Libcloud, http://libcloud.apache.org/, [Accessed 22 September 2015].

[16] jclouds, http://jclouds.apache.org/, [Accessed 28 September 2015].

[17] M. Hamdaqa and L. Tahvildari, “The (5+1) architectural view model for cloud applications,” in 24th CSSE, 2014, pp. 46–60.

[18] J. Bézivin, “Model driven engineering: An emerging technical space,” in Generative and Transformational Techniques in Software Engineering, 2006, pp. 36–64.

[19] MDA, Model Driven Architecture, http://www.omg.org/mda, [Accessed 27 November 2015].

 ISBN: 978-1-61208-460-2

Modeling Workflow of Tasks and Task Interaction Graphs to Schedule on the Cloud

Mahmoud Naghibzadeh
Department of Computer Engineering
Ferdowsi University of Mashhad
Mashhad, Iran
Email: naghibzadeh@um.ac.ir

***Abstract*-Many composite computational activities are modeled as directed acyclic graphs called workflows in which each vertex is a task and each directed edge represents both precedence and possible communication from its originating vertex to its ending vertex. When the execution of a task is completed, the communication with its successor(s) starts and anticipated data are transferred. Only after all parents of a task are completed and their results (if any) are received by the task its execution can start. These constraints restrict a more general case in which some tasks could communicate during their executions. In this paper, a task-model composed of both interaction and precedence of tasks is introduced. It is shown that this kind of graph can be transformed into an extended directed acyclic graphs, called hybrid directed acyclic graph, composed of tasks and super-tasks. Super-tasks need not be recognized manually and the proposed method automatically finds them. This can simplify the design of complex workflows. Validity conditions of hybrid directed acyclic graphs are investigated and a verification algorithm is developed. Also, scheduling aspects of hybrid workflows on the cloud is highlighted and some results are reported. This inventive idea can open a whole new area of research and practice in the field of workflow modeling and scheduling.**

Keywords-Task interaction-precedence graph; hybrid DAG; hybrid workflow; Cloud computing.

I. INTRODUCTION

Activities of many composite processes such as likelihood propagation using Bayesian network, family trees (actually graphs) in genealogy, and scientific and industrial workflows, are modeled as *Directed Acyclic Graphs* (DAGs) in which many vertices are connected via directed edges [1]. As the name suggests, an important property of such graphs is that there is no cycle in a DAG. This model is applicable where there is control and data dependencies and a partial precedence relation between tasks and if a task has to transfer data to one (or more) of its successors it can do so at the end of its execution, i.e., in the middle of execution it is not possible for two or more tasks to interact. This resembles an assembly line in which one station has to complete its job and then pass the object to the next station. In any case, a DAG represents a partial ordering of tasks that must be obeyed for their executions. Figure 1 shows a DAG composed of 11 vertices and 15 edges. The number next to a vertex shows its required execution time averaged on all applicable resource types. The real resource type for the execution of each task will be determined during the actual scheduling of the workflow, before the execution of the workflow starts. Consequently, the exact execution time of each task will be computed using the average execution time given on the workflow and the relative processing power of the resource to be used to the average processing power of the resource types. Likewise, the number on an edge shows the average data transfer time from the originating vertex of the edge to the ending vertex of the edge. The exact transfer time will be computed during the scheduling time when the actual resources for source and destination tasks are determined, hence, the communication link and its baud rate is known. The transfer takes place at the end of the execution of the originating task of the edge.

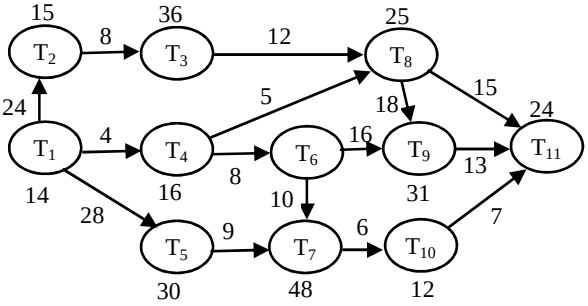


Figure 1. A DAG of precedence-communicating tasks

On the other hand, a Task Interaction Graph (TIG) is used to show which tasks of an application interact during their executions [2]. All tasks of a TIG can start simultaneously and each task can continue its execution so far as it does not need to synchronize or communicate with any other task. Only two directly connected tasks, i.e., neighbors, could interact and it is possible that they may not do so depending on the logic of the corresponding programs and the current values of data objects. Also, even the interaction may be one sided, in some circumstances. The nature of interaction depends on the parallel problem being solved; it may be an actual information transfer or an indirect communication to use a shared data object. For example, suppose a parallel program is designed to multiply a sparse Matrix *M* with *m* rows and *n* columns by vector *V* with *n* rows. Suppose Task *i's* responsibility is pairwise multiplication of elements of Row *i, i=1,2,…m* of Matrix *M* and corresponding elements of Vector *V* and computing their sum, Formula (1),

$$Task_i: \sum_{1 \le j \le n,\ M[i,j] \neq 0} M[i,j] * V[j] \qquad (1)$$

Here, Task *k, k=1,2,..n* and Task *l, l=1,2,…n, l≠k* conflict on elements of Vector *V* for which both *M[k,j]* and *M[l,j]* are nonzero [3]. These two tasks cannot simultaneously access the same memory location. The nature of task interaction in this example is indirect. In the context of scheduling DAGs and TIGs, and now the Task Interaction-Precedence Graph, a task is a piece of work that is completely assigned to one processor to do.

The cloud, provides wide varieties of resources and software in the forms of Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) for public use [4]. Users can lease these services for the needed periods of time and pay as they are used. This is a great opportunity for many users who are short of required resources to run their computational jobs. Scheduling scientific workflows on the cloud is an ongoing research activity with continuous improvements on their quality of services such as make-span, price to be paid, energy efficiency, and fairness. *Fairness* is related to workloads of workflows belonging to one enterprise where it wants to be nondiscriminatory in assigning resources to different workflows [5]. A workflow, if modeled as a DAG, is unable to handle task interactions that can happen any time during their executions. In this paper, a new task modeling approach called *Task Interaction-Precedence Graph* (TIPG) is introduced in which all types of task precedence, dependency, and interaction is possible, subject to passing validity tests. Difficulties involved in the scheduling of applications which can be modeled using TIPG is studied. Some scheduling results are presented which shows the success rate of scheduling is improved. However, the originality of this work is on the introduction of a new task model which allows tasks of a workflow to directly or indirectly interact during their execution.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 formulates the problem being studied. Section 4 discusses the validity verification of a given TIPG and its scheduling on the cloud, including the designed algorithms for both verification and scheduling. Section 5 concludes.

II. RELATED WORK

In scheduling workflows on the cloud, we are faced with different quality of services needed by users. Optimal scheduling of a workflow represented by a general DAG is an NP-hard problem [6]. Therefore, different approaches such as integer linear programming [7], genetic algorithms [8], and heuristic algorithms [9], are proposed to produce close to optimal solutions. The objective is often meeting a user-defined deadline, minimizing the computational cost of the workflow, minimizing the make-span, and/or maximizing the success rate of the proposed algorithm. Of course, some algorithms are multi-objective which means that they are designed to optimize more than one objective, such as minimizing timespan and cost at the same time [10]. Here, *make-span* is defined to be the time length from when the workflow is submitted to the cloud to the time when its execution is completed. *Success rate* is the ratio of the number of workflows successfully scheduled to the total number of workflows examined. There are many other aspects to scheduling workflows such as data and computation privacy [11] and simultaneous scheduling of many workflows belonging to one organization [12].

Scheduling TIGs is another field which makes the foundation of this paper's scheduling extended workflows. In a connected TIG, it is not possible to complete one (or many but not all) task at a time but the whole TIG must complete at one time. For example, the whole sparse matrix and vector multiplication is one super-task and computing each element of the resulting vector can be organized as one task. Tasks of such a super-task can simultaneously run on different hardware resources of the cloud. However, it is possible to assign more than one task to one processor to be executed in some specified order. Figure 2 shows a TIG graph for accomplishment of the parallel multiplication of a small-size sparse matrix and a vector. A TIG may represent a complete independent application or it may be part of a larger application. For example, it can be matrix multiplication using matrix-vector product to reduce the number of operations as part of solving a system of linear equations. The latter is widely used in many applications.

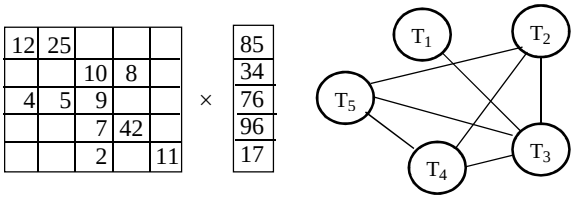


Figure 2. The TIG of matrix-vector multiplication

TIGs are traditionally scheduled on multiprocessors and computer clusters. Nowadays, distributed systems and the clouds provide favorable platforms for solving parallel applications [13]. A TIPG is neither a simple DAG nor a TIG but a new graph model in which both interaction and precedence is allowed. The closest model to this could be a DAG (or workflow) of simple (sequential) tasks and parallel tasks. Nevertheless, there are many differences between the two. (1) Before the production of a DAG of tasks and parallel tasks (which will be called TPDAG for short) parallel tasks of the application being modeled have to be recognized and to be considered as indivisible units, similar to simple tasks within the model. However, a TIPG model starts with (simple) tasks only and there is no composite task in the beginning. Two different types of relations are allowed between each pair of tasks, interaction and communication. Interaction resembles concurrency of tasks and information exchange during execution, while communication resembles precedence and data transfer from one task to the other at the end of execution of the former. The recognition of super-tasks in the TIPG is automatic and without the interference of users. This increases ease of workflow design and at the same time the possibility to recognize parallel tasks in their smallest possible size. (2) Parallel tasks of a TPDAG are co-scheduled to run in parallel, i.e., all subtasks of a parallel task start simultaneously and the whole parallel task

 ISBN: 978-1-61208-460-2

completes at one time. This means, a child of a parallel task cannot start until the whole parallel task is completed and its data (if any) is received [8]. On the other hand, tasks of a super-task within a TIPG can run concurrently, i.e., they can start at different times and complete at different times as long as interactions are possible. Besides, any child of a super-task can start its execution as soon as its parent tasks (not the whole super-task) are completed. This can have a great impact on the time span of the TIPG and as a result on the price to be paid for running the TIPG on the Cloud. (3) Inputs to a parallel task are receive by the parallel task itself and outputs are also sent by the parallel task. For a super-task of a TIPG, any input from external tasks of a super-task is independently received by the receiving task within the super-task and any output from a task within the super-task is sent directly by the sending task within the super-task. As a result, parallel communication from/to tasks of a super-task is the norm. This is another flexibility, which can have a great impact on the quality of scheduling services. Our work could be considered as an extension of workflows of tasks and parallel tasks which have been used by many applications. A well-known such application is Proteomics workflow [14]. Many workflow management software such as Pegasus have the capability to handle parallel tasks.

III. PROBLEM FORMULATION

A directed acyclic graph, when used to model workflows, prohibits tasks to directly or indirectly interact during their executions. We propose an extended model called TIPG which allows interaction, communication, and precedence capabilities, simultaneously.

A TIPG, *G(V, U, D)*, consists of a set of vertices, *V*, a set of undirected edges, *U*, and a set of directed edges, *D*. Each edge, directed or undirected, connects one vertex to another and there could be at the most one edge between any pair of vertices. Note that, the transfer of information between two interacting tasks could occur any time during their executions or even, at the end of the execution of one task it could send information to another task (which is not completed yet) for the last chance. Because we want TIPGs to be extensions of DAGs, considering directed edges only, a TIPG must be acyclic, i.e., it should not be possible to start from some vertex and follow a sequence of directed edges and reach back to the same vertex. For undirected edges only, cycles are not forbidden. For a TIPG graph to be valid it must be acyclic with respect to directed edges and also conflict-free.

Definition 1: A TIPG is *conflict-free* if there is no directed path between any pairs of vertices of any connected component of the TIPG where only undirected edges are considered to obtain these connected components.

Definition 2: In the rest of this paper, wherever we talk about connected components of a TIPG graph *G(V, U, D)* we mean, a sub-graph $G'(V', U') \subseteq G(V, U, D)$, $V' \subseteq V$, *and* $U' \subseteq U$, i.e., U' is composed of only undirected edges of the TIPG.

Definition 2 complies with the classical definition of connected components for undirected graphs when all directed edges of the TIPG are ignored. Recall that, in computer science, for a directed graph, the concept of strongly connected components is used rather than connected components.

If there is any conflict in a TIPG graph it means that there is at least one pair of vertices that one vertex precedes the other and at the same time they can interact during their executions. This is definitely impossible because parallel and precedence constraints may not intermix, hence the design of such a system is incorrect and it should be fixed before going about running it. Therefore, conflict detection, which will be discussed later, could be thought of one type of design verification.

For example, sparse matrix and vector multiplication is seldom a complete application and it can be part of solving a greater problem such as matrix multiplication using matrix-vector product to reduce the number of operations and solving a system of linear equations. Although, in some workflows, cycles composed of two or more tasks are possible, whenever the control flow reaches one of these tasks it should start executing from the beginning. This is not the case for a TIG which is part of a workflow. If an undirected edge connects two vertices v_i and v_j it is represented by either (v_i, v_j) or (v_j, v_i), with no distinction. However, if a directed edge connects two vertices v_k and v_ℓ it is represented by $<v_k, v_\ell>$, where v_k is the starting vertex and v_ℓ is the ending vertex of the edge. A directed edge between a pair of tasks, $<v_k, v_\ell>$, means that the execution of task v_k must precede the execution of task v_ℓ and that v_k can transfer information to v_ℓ once at the end of v_k's execution.

Suppose for the DAG of Figure 1, tasks T_3 and T_4, T_4 and T_5, and T_9 and T_{10} have to interact during their executions. The resulting TIPG is shown in Figure 3. In this figure, all execution and communication times are removed only to increase clarity of the graphical representation of the TIPG, but they are implicitly in place.

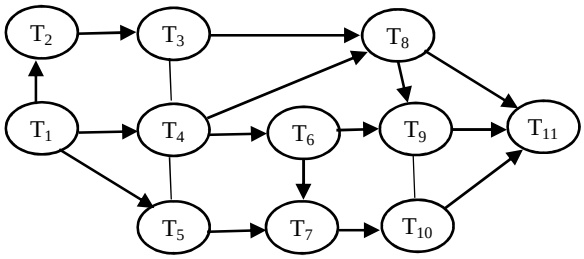


Figure 3. A sample task interaction-precedence graph

Later it will be shown that this TIPG is conflict-free, which means it could be a valid model for tasks of a real application. Such an application cannot be modeled by neither a DAG nor a TIG graph and nor a workflow of tasks and parallel tasks. To represent a TIPG graph an $N \times N$ matrix, *M*, where *N* is the number of vertices, is used. In this matrix, which is usually very sparse, if there is no edge between v_i and v_j, $i, j = 1,2,3,\ldots N$ then $M[i,j] = Null$; if there

is an undirected edge from vertex v_i to vertex v_j then $M[i, j]=0$; otherwise $M[i, j]$ is set to the average data transfer time from v_i to v_j, i.e.,

$$M[i,j] = \begin{cases} null, & if\ no\ edge\ between\ V_i\ and\ V_j \\ 0, & if\ undirected\ edge\ between\ V_i\ and\ V_j \\ data\ transfer\ time\ from\ V_i\ to\ V_j, & otherwise \end{cases}$$

IV. VALIDITY VERIFICATION AND SCHEDULING

In the context of this research a TIPG must pass three test to become a valid *Hybrid Directed Acyclic Graph* (HDAG), i.e., A DAG of tasks and super-tasks. (1) The original TIPG must be acyclic with respect to directed edges, i.e., ignoring all undirected edges. (2) The TIPG must be conflict-free. (3) Considering each connected component as being an indivisible super-task with one entry and one exit, such a graph must be acyclic. Condition 2 is an absolute must satisfy statement, otherwise the model is not valid. Condition 1 is the restriction of our research meaning that, this research aims at eventually scheduling workflows which can be modeled as DAGs and no other models of workflows such as iteration structure. Condition 3 is a complementary to Condition 1.

An algorithm has to be designed to systematically perform these tests. After the first test is successful, it has to find all connected components of the graph considering only undirected edges. A connected component should include at least two vertices. For every pair of vertices of every connected component we have to make sure there is no directed path from one to the other, i.e., considering only directed edges of the TIPG in this stage. This is called conflict-freeness of the TIPG. Using notations of Definition 2, if for any two vertices $(v_i, v_j) \subseteq U'$ there is at least one path, $P \subseteq U'$, from v_i to v_j the TIPG is not conflict-free and hence the model is not valid. Then the whole TIPG is restructured in the form of a directed graph by considering each whole connected component as one indivisible super-task. For this phase, communications to all tasks of a super-task are received by the super-task itself and all communications from the tasks of a super-task are from the super-task itself. The final check is to make sure that the new directed graph of tasks and super-tasks is acyclic. It is worth mentioning that restructuring action is only for the purpose of making sure that there is no cycle in the model. After this checking action is completed, the communications to/from a task within a super-task would directly go to/from the task itself. Algorithm 1 shows the steps to be taken for transformation of a TIPG to a directed graph and verifying its validity as a hybrid DAG. At the start of the algorithm, Matrix M is the representation of the TIPG as it is explained earlier. This matrix will be augmented with new rows and columns corresponding to the connected components found. The number of elements in vector V is equal to the number of vertices on the original TIPG. At the end, Vector V represents which vertex belongs to which component, i.e.,

$$V_i = \begin{cases} 0, if\ vertex\ i\ is\ not\ in\ any\ connected\ component \\ id\ of\ connected\ component, otherwise \end{cases}$$

Since algorithms for finding cycles (if any) in a directed graph (Test 1) is not a new topic, we will assume that the given TIPG graph is already acyclic. Algorithm 1 first produces all connected components and then proceeds with the validity tests. See Figure 4.

```
---------------------------------------------------------------------------
1:Boolean procedure HDAG_Production&Validity(M, V)
2:    while (exists new connected component, C)
3:       Update vector V to represent this component
4:    end while /*from here on components are connected*/
5:    for every (pair of vertices (vi,vj) of every component) do
6:       if (exists a directed path from vi to vj or vj to vi)
7:             return false
8:       end if
9:    end for
10:   for every (component, C) do
11:      add a new row and column to M and
               fill its entries using Rules 2 to 5
12:   end for
13:   for every (vertex v of every component) do
14:      replace all positive values of row v by null
15:      replace all positive values of column v by null
16:   end for
17:   if (cycle (M, V)) return false
18:   else return true
19:   end procedure
---------------------------------------------------------------------------
```

Figure 4. Algorithm 1- producing a potential Hybrid DAG (HDAG) from a TIPG and checking its validity

In Lines 2 to 4 all connected components of the given TIPG are found, one by one. From Line 4 on any reference to component means connected component. Vector V is set to represent which tasks of the original TIPG are parts of which component, if any. The conflict-freeness of each connected component is then checked in Lines 5 to 9 and if there is at least one conflict in any connected component there is a design error and the original TIPG must be redesigned. Conflict-freeness of all connected components implies conflict-freeness of the whole TIPG. Each component is called a super-task, in its entirety, is now a new object in the model. To represent connections of each of these new objects with other objects, for every component a new row and a new column is amended to the Matrix M, i.e., super-tasks are represented in the same structure where tasks are represented. The values of elements of these rows and columns are filled with respect to communication times between tasks and super-tasks to this super-task (and vice versa) using Rules 2 to 5 that are discussed later. Lines 10 to 12 of the algorithm have this responsibility. There should not be any edge from a task or a super-task outside a given super-task to a task inside this super-task and vice versa. Such edges have to be changed to/from the super-task itself. Lines 13 to 16 are intended to serve this purpose. What remains is that we have to make sure the resulting graph of tasks and super-tasks, where each super-task, as a whole, is considered an indivisible unit of work in this stage, is acyclic. Line 17 will take care of this job and if a violation is diagnosed the graph is not acyclic. Otherwise, the hybrid DAG is valid. In the body of Algorithm 1 the following rules are used.

Rule 1: All vertices and undirected edges of every connected component are separately grouped and encapsulated as a super-task, before going about checking the correctness of the model.

 ISBN: 978-1-61208-460-2

Rule 2: Any data sent from an external task or super-task to a task within a super-task is sent to the super-task itself.
Rule 3: Any data to be sent from a task within a super-task to an external task or super-task is sent by the super-task.
Rule 4: If an external task or super-task wants to send data to more than one task within a super-task the average of the data volumes over all receiving tasks is sent to the super-task. It is assumed that there are independent communication links between each pair of resources. This assumption enables us to consider average of data instead of sum of data. There are two points to be cleared with respect to sending data to more than one task within a super-task: (1) message processing time is negligible compared to message transfer time, hence the sender can handle parallel sends simultaneously, and (2) each receiver task can immediately start its execution after the expected data from its parent(s) is received. These two points justify using the average data volume instead of the sum of data volume on a link which connects an external task or super-task to many tasks within a super-task.
Rule 5: If more than one task within a super-task want to send data to an external task or super-task the average of the data volume is sent by the super-task. The justification for using the average of data volume, instead of the sum of data volume, is similar to the reasoning used for Rule 4.

A. Time complexity of Algorithm 1

The number of operations needed for lines 2 to 4 of Algorithm 1 is proportional to *(N+E), i.e.,* $C_1(N+E)$, where C_1, N, and E, are a constant, the number of vertices, and the number of edges in the original TIPG, respectively. The number of operations for Lines 5 to 9 is proportional to the number of pairs of vertices times *NlogN+E, i.e.,* $C_2N^2(NlogN+E)$. The number of operations for line 17 is $C_3(N+E)$. Therefore, the time complexity of the algorithm is dominated by the time complexity of Lines 5 to 9. Consequently, the time complexity of the algorithm is $O(N^2(NlogN+E))$. For the TIPG of Figure 3, Vector *V* will be filled as in Formula (2).

$$V^{'}=(0,0,0,1,1,1,0,0,0,2,2,0)^{'} \quad (2)$$

There are two connected components, (1) the TIG of tasks *T3*, *T4*, and *T5*, and (2) the TIG of tasks *T9* and *T10*. There are no directed paths from *T3* to *T4*, *T4* to T5, *T3* to *T5*, *T4* to *T3*, *T5* to *T4*, *T5* to *T3*, *T9* to *T10*, or *T10* to *T9*. Therefore, the TIPG of Figure 3 is conflict-free. The algorithm checks and makes sure that there is no cycle in the whole new graph of tasks and super-tasks when each super-task is considered indivisible. The new graph is a valid hybrid workflow. The resulting hybrid DAG is shown in Figure 5.

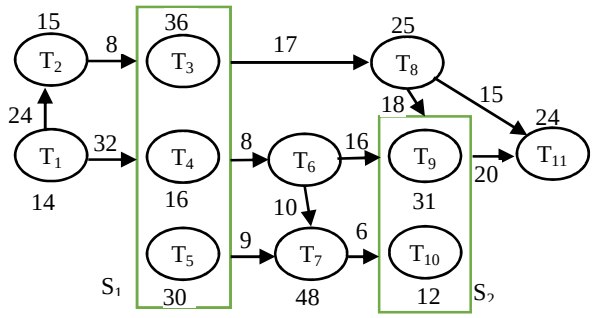


Figure 5. Hybrid DAG obtained from TIPG of Figure 3

An actual implementation of Algorithm 1 could include a preliminary stage of finding the reachability matrix of the graph by considering only directed edges. This can be done using the Floyed-Warshall algorithm [15]. The matrix can be accessed in Line 6 of the algorithm many times. The complexity of the preliminary stage of the algorithm is $O(N^3)$, where N is the number of vertices. This is a lower time complexity than the time complexity of the algorithm which we produced earlier, hence the time complexity of the algorithm is still $O(N^2(NlogN+E))$.

B. Scheduling hybrid workflows

Conventionally, workflow schedulers assume that tasks are non-preemptable hence, each processing resource runs the current task in a single-programming fashion. With the extension of workflows to include super-tasks called TIGs multiprogramming becomes more attractive and beneficial to the cloud users. Tasks of a TIG may have quite different execution times, on the one hand and, they should be co-scheduled to be able to interact during execution. We are faced with three options, (1) allocation of as many processing resources as there are tasks and assigning each task to one resource, (2) assigning all tasks of each TIG to one fast enough processing resource and running then in a multiprogramming fashion, and (3) clustering tasks and allocation of as many processing resources as there are clusters and assigning each cluster to one resource then running each cluster in a multiprogramming fashion.

By selecting Option 1 we are aware that all tasks should start simultaneously in order to be able to interact during execution since otherwise resources' time could be wasted due to tasks wanting to interact with others which are not available. It is not always the case that whenever we want to schedule a TIG in the midst of scheduling a workflow it would be beneficial to lease new resources from the cloud. Furthermore, at the time of assigning a TIG, the finish time of already leased resources may not be the same hence waste of some resource's time is inevitable. Depending on the nature of the application being scheduled, we might want to start a descendent task or super-task when all tasks of a TIG are completed. For this case the scheduler should make use of simultaneous completions of tasks if it is beneficial towards the scheduling objectives. These kind of applications are called parallel TIGs.

Option 2 can be useful in many cases for example, when the objective is to minimize cost of running the workflow on the cloud and the given relative deadline is not very short. A *relative deadline* is a duration of time-interval that is given to a workflow to complete as opposed to an *absolute deadline* which is an exact moment of time before or at which the workflow must be completed. A major benefit of this approach is that the interaction time is negligible because the common main memory could be used for data and results sharing.

Option 3 is applicable when Option 2 is not useable with respect to the scheduling objectives. In any case, we cannot simply say that the time-span of executing a super-task is for example equal to the sum of execution times of its included tasks. It depends on the approach that the TIG

 ISBN: 978-1-61208-460-2

is modeled considering aforementioned models. An estimate of execution time is needed before we can go about designing a hybrid workflow scheduler. Although there are differences between scheduling hybrid workflows and simple workflows, the guidelines are already developed.

To make the paper short, a brief summary of what has been done with respect to scheduling TIPGs and the results obtained up to now follows. The base of our two experiments was the graph of Figure 1. Primarily, Edges $<T_4, T_3>$, $<T_5, T_4>$, and $<T_9, T_{10}>$ were added to the graph. Six hundred TIPGs were randomly generated from the topology of this modified graph. That is, all execution times and communication times were first removed from the graph but vertices and edges were not changed. The execution time of each vertex was then randomly selected from the interval [0, 50] and communication time of each edge was randomly generated from the interval [0, 30]. An integer number between zero to eight (exclusive) was randomly selected and that many vertices of the graph were randomly selected and were changed to undirected edges. If the graph was a valid Hybrid DAG it was selected for scheduling. A list scheduling heuristic with the objective of minimizing timespan [16], while the deadline is respected was developed to schedule the workflow. The deadline parameter was randomly selected within the interval [CPL, 1.4*CPL], where CPL is the Critical Path Length of the workflow. The CPL was calculated in a different way which is usually calculated. All communication times were ignored in the calculation of the critical path because the whole critical path could be assigned to one processor, hence communications are zeroed. This kind of critical path represents the absolute minimum timespan for running the workflow, with regards to the resource type being used. Five types of the cloud processing resources with performances 1.0, 1.2, 1.6, 2.4, and 3.0 with price factors equal to their performances were assumed. The ready task (or super-task) with the highest rank was scheduled next. The two experiments differ in the way a super-task is scheduled. The objective of the experiments was to compare parallel and concurrent execution of super-tasks. See Figure 6 (Algorithm 2.) In this algorithm, *lease_appropriate(1)* will lease one new resource with proper processing power, considering the current state of the scheduler. Similarly, *lease_appropriate(N_i-available)* leases as many as the difference of N_i, i.e., the number of tasks in the i^{th} super-task, and the number of available resources, if N_i is greater than the number of available resources. The procedure *assign_task(task)* assigns the task to the resource which completes it the soonest. On the other hand, *assign_stask(super-task)* assigns all tasks of the super-task to as many processors as needed by the super-task.

Experiment 1 followed Option 1 guideline. For this case, effective execution time of each task is equal to the maximum execution time of the tasks of the super-task. Experiment 2 also followed Option 1 with the possibility of each task of a super-task to complete and sent its data to its children vertices, independently. A simple heuristic (not necessarily optimal) was used to assign tasks within a super-task to resources. If Super-task S_i needed N_i simultaneous resources, N_i resources with earliest availability time were found, first. If there were not enough available resources new resources were leased. Then the task with the longest execution time (with respect to the processor's performance) was assigned to the resource which can complete it the earliest time. This task and the corresponding processor were removed and the process continued until allocation is completed.

```
---------------------------------------------------------------------------
1: Retrieve cloud resources' availability and prices
2: Rank tasks and super-tasks and enqueue (Q)
3: mark the entry task as ready
4: While (exists ready task in Q)
5:     remove (highest ranked ready task T (Q))
5:     case 1: regular task Ti (T, V) //A simple task
6:       begin
7:           lease_appropriate (1)
8:           call assign_task (Ti)
9:            remove father-son links of Ti
10:             if a child becomes orphan mark it ready
11:       end begin
12:    case 2: super-task Si (T, V) // A super-task
13:         begin
14:            if available resources are not enough
15:             lease_appropriate(Ni-available)
16:            end if
17:             call assign_stask(Si)
18:              remove father-son links of Si
19:              if a child becomes orphan mark it ready
20:           end begin
21: end while
22: end algorithm
---------------------------------------------------------------------------
```

Figure 6. Algorithm 2- scheduling hybrid workflows of tasks and TIGs

For these experiments the success rate of the second experiment was 7% higher (with respect to the total number of generated workflows) than that of the first experiment. From the 600 workflows with the assigned deadlines, 276 cases were successfully scheduled in the second experiment whereas 235 were successfully scheduled in the first experiment. The success rate of the second experiment was 0.46 and that of experiment one was approximately 0.39. It is a good indication that TIPGs should have their own schedulers rather than using schedulers of workflows of tasks and parallel tasks. It is worth mentioning that a TIPG can handle both parallel and concurrent super-tasks. Having applied the ideas on small-scale TIPGs it can be extended to large-scale scientific workflows with thousands of tasks and super-tasks. Scheduling, and then running, such workflows requires powerful supercomputers or a Cloud environment composed of thousands of Virtual Machines (VM).

V. SUMMARY AND FUTURE WORK

A new task model called task interaction-precedence graph is presented in this paper. Tasks, in this model, not only can communicate information at the end of their executions but they can also interact during their executions. A validity algorithm is developed to check the correctness of the design of such a model. Furthermore, a procedure for transforming a conflict-free TIPG into a hybrid DAG, i.e., a

 ISBN: 978-1-61208-460-2

DAG composed of simple tasks and super-tasks is established in order to be able to check that the final graph is acyclic. Problems involved in scheduling TIPGs on the cloud are investigated. This new modeling technique allows us to model many more complex applications which were not possible to model using DAGs or TIGs alone. The novelty claim of this paper is in the introduction of a new task model in which precedence, interaction, and communication are all simultaneously possible. Hybrid workflows, Hybrid DAGs, and even *hybrid graphs* become meaningful and very useful. The scheduling algorithm developed in this paper is the first step in this regard and much work has to be done. The experiments of this paper focused on the scheduling aspects of TIPG workflows. It is predictable that starting with a graph in which both directed and undirected edges, i.e., communications and interactions, are allowed and the granularity of all tasks are the same and the recognitions of TIGs are automated, the model designing is simpler and less time consuming. However, the actual performance is left to be evaluated in the future.

REFERENCES

[1] D. C. Kozen, The Design and Analysis of Algorithms. Springer, 1992.

[2] D. L. Long and L. A. Clarke, "Task interaction graphs for concurrency analysis," [1988] Proceedings. Second Work. Softw. Testing, Verif. Anal., 1988.

[3] A. Grama, A. Gupta, G. Karypis, and V. Kumar, "Introduction to Parallel Computing; 2nd Edition," Search, 2003, pp. 856.

[4] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds," Futur. Gener. Comput. Syst., vol. 29, no. 1, 2013, pp. 158–169.

[5] L. F. Bittencourt and E. R. M. Madeira, "Towards the Scheduling of Multiple Workflows on Computational Grids," J. Grid Comput., vol. 8, 2010, pp. 419–441.

[6] O. Sinnen, Task Scheduling for Parallel Systems. John Wiley & Sons, 2007.

[7] T. A. L. Genez, L. F. Bittencourt, and E. R. M. Madeira, "Using time discretization to schedule scientific workflows in multiple cloud providers," in IEEE International Conference on Cloud Computing, CLOUD, 2013, pp. 123–130.

[8] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," Sci. Program., vol. 14,, 2006, pp. 217–230.

[9] S. Abrishami and M. Naghibzadeh, "Budget Constrained Scheduling of Grid Workflows Using Partial Critical Paths," in International Conference on Grid Computing and Applications, GCA, 2011, pp. 1o8–114.

[10] E. Juhnke, T. Dörnemann, D. Bock, and B. Freisleben, "Multi-objective scheduling of BPEL workflows in geographically distributed clouds," in Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011, 2011, pp. 412–419.

[11] S. Sharif, J. Taheri, A. Y. Zomaya, and S. Nepal, "MPHC: Preserving Privacy for Workflow Execution in Hybrid Clouds International Conference on Parallel and Distributed Computing," in Applications and Technologies (PDCAT), 2013, pp. 272–280.

[12] A. Hirales-Carbajal, A. Tchernykh, R. Yahyapour, J. L. González-García, T. Röblitz, et al., "Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid," Grid Comput., vol. 10, no. 2, 2012, pp. 325–348.

[13] I. D. Falco, U. Scafuri, and E. Tarantino, "Two new fast heuristics for mapping parallel applications on cloud computing," Futur. Gener. Comput. Syst., vol. 37, 2014, pp. 1–13.

[14] G. Mehta, E. Deelman, J. A. Knowles, T. Chen, Y. Wang, J. Vöckler, et al., "Enabling data and compute intensive workflows in bioinformatics," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2012, vol. 7156, LNCS, PART 2, pp. 23–32.

[15] R. W. Floyed, "Algorithm 97: Shortest path," Commun. ACM, vol. 5, no. 6, 1962, pp. 345–345.

[16] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Trans. Parallel Distrib. Syst., vol. 13, no. 3,2002, pp. 260–274.

 ISBN: 978-1-61208-460-2

Instruments for Cloud Suppliers to Accelerate their Businesses

The Need for Unleashing Business Potential and Boosting Cloud Economics

Fred Kessler
InnovationLeaders Academy
fk@innovationleaders.ceo

Stella Gatziu Grivas, Claudio Giovanoli
Institute for Information Systems
University of Applied Science Northwestern Switzerland
{stella.gatziugrivas, claudio.giovanoli}@fhnw.ch

***Abstract*—Major IT suppliers like HP, IBM, Microsoft and Oracle are pushing their cloud services and technologies to the global markets. Their success varies considerably, especially in Europe. From a Microsoft perspective, the Scandinavian countries are far ahead in adopting cloud services, and the German speaking countries are far behind. Distribution channel partners (also IT-suppliers) and customers refuse to buy in. In addition, IT suppliers´ communication is mainly technical and as a result, customers do not understand how cloud service adoption would bring any profitable strategic advantage. This also inhibits the use of Electronic Marketplaces for Cloud Services (EMPCS), Cloud Service Brokerage (CSB) or Electronic Market Places (EMP). Suppliers need instruments and support for anticipating, communicating and measuring the business potential of and with their customers. Cloud transformation for suppliers means changing their approach to cloud services, in order to increase their competitiveness and cash flow, as well as that of their customers`.**

Keywords - business potential reference model; cloud services; customer value orientation; open innovation; transformation process.

I. INTRODUCTION

In Austria, Germany and Switzerland, Cloud business got stuck before it even really started. Besides other obstacles, like legal and security issues, customers do not buy cloud services, mainly because IT-suppliers fail to communicate the business value to business decision-makers.

One major reason for their dysfunctional attempts is their inability to identify and analyze customers´ business potential. In this context our empirical research and conversations with more than 300 executives of Microsoft partner companies in Austria, Germany and Switzerland reveal that they do not even try to measure the business impact of their IT-solutions, which means that they are not even interested in monetary customer business value [8]. Instead, suppliers talk (in their marketing) about technologies like azure, hyper-v, high availability, fault tolerance, big data, predictive analytics and so forth instead of projecting the achievable customer results of their service bundles in terms of cash flow and competitiveness. Customers are confused and do not sign cloud services contracts. Cloud business in these countries is far behind other European and global countries. There are also other reasons for not buying cloud solutions, which are linked to the customers themselves. The banking and insurance industries for instance, who would derive major and profitable strategic advantages from adopting cloud services, refuse to bring their sensitive information to the cloud.

With this positioning paper, we structure the problem and introduce an approach to resolve the situation. Starting with assessing the IT-suppliers with regard to their *customer-value-orientation* [8], we then introduce a *business potential reference model* as a common denominator for customers and IT-suppliers. With such a model in place, we then develop a sound innovation process as the central instrument for IT-suppliers to create, sell, implement, and measure monetarily valuable cloud solutions, with, for and to customers.

Various attempts have been made to define the term "Cloud Economics" in the literature [27][28][29]. The one that we perceive as closest to our understanding is: "Cloud economics is a branch of knowledge concerned with the principles, costs and benefits of cloud computing" [30]. In our research, we develop instruments that help leverage financial transactions between providers and consumers of cloud services.

The main objective of our research is to answer the question of how the business transformation of cloud suppliers can be performed successfully. Finally, we introduce a 9-stage end-to-end transformation process [8] with all relevant instruments to help IT-suppliers master the inevitable change, associated with the process.

In Section II, we focus on the definition of customer value orientation, regarding the expected advantages of cloud services usage, by defining a catalog of indicators to measure the advantages. In Section III, we discuss our model for defining the business potential on the customer side, which can be achieved by the enabling technologies on the IT-Supplier side. In Section IV, we introduce some initial elements of the transformation process that must be undertaken by IT-Suppliers, so that their services can help customers unleash business potential.

 ISBN: 978-1-61208-460-2

II. ASSESSING IT-SUPPLIERS´ CUSTOMER VALUE ORIENTATION

Initially, we need to define customer value orientation. The main components are *customer value* and *orientation*. The next step would be to revisit the conceptualization and operationalization of the customer value orientation to make it fit the modern needs of cloud economics. With this in mind, we can assess IT-suppliers´ current *customer value orientation* status, in order to help them focus on the right target (see Figure 1. Misaligned customer value orientation).

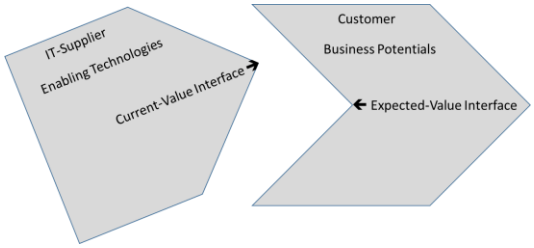


Figure 1. Misaligned customer value orientation

A. Customer value

We asked executives from the customer side for their criteria when judging capital expenditures (CapEx) or major operational expenditures (OpEx), which would both also apply to cloud services. Their answers were all about Return on Investment (ROI), Break Even, Net Present Value (NPV), etc., being very clearly determined and at the same time predictable. We conclude that the customer value they are looking for is something that helps them either generate a better interest rate on their invested capital or helps them increase competitiveness.

B. Orientation

In the literature, there are various definitions of the term "orientation" and, depending on the discipline, there are different perspectives. At least three relevant aspects refer to the main characteristics of orientation. First, there is the environmental positioning and information collection, second, there are values, guidelines and principles and third, there are intended behavior and targeted activities [8].

When it comes to organizational orientation with IT-suppliers, we have, without exception, encountered implicit technological orientations but no explicitly described models with other objectives.

There are various forms of orientation described in the literature and there are also very diverse definitions of value, but only one orientation form focusses on monetarily measurable customer value [8]. It seems reasonable to orient or realign an entire IT-suppliers´ organization around providing exactly the value that executive business decision-makers on the customer side are looking for.

C. Conceptualization and operationalization of customer value orientation

Organizational orientation is often conceptualized with the same pattern. There are success factors on a corporate level, on an interaction level and on a service level (see Figure 2. Levels and success factors of customer value orientation).

Each factor is subsequently described by formative and/or reflective indicators with underlying values, usually measured on a scale from 0-100.

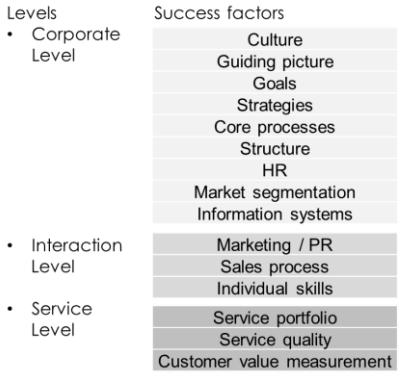


Figure 2. Levels and success factors of customer value orientation [8]

Implementing customer value orientation in an IT-supplier organization means synchronizing each individual success factor with the main objective. For instance, with *Goals*, the organization would express and document the customer value it wants to create and provide, *Strategies* would express how the organization will create the customer value, *HR* would define how members of the organization are hired, measured and fired in the process of creating customer value, *Marketing/PR* would communicate customer value, *Individual skills* would be developed with members of the organization to create and deliver customer value and, last but not least, *Customer value measurement* would ensure having leading indicators in place to control, prove and finally document the customer value.

The success factors´ individual level of sync with the customer value orientation is measured and depicted in a diagram (see Figure 3. Customer value orientation status).

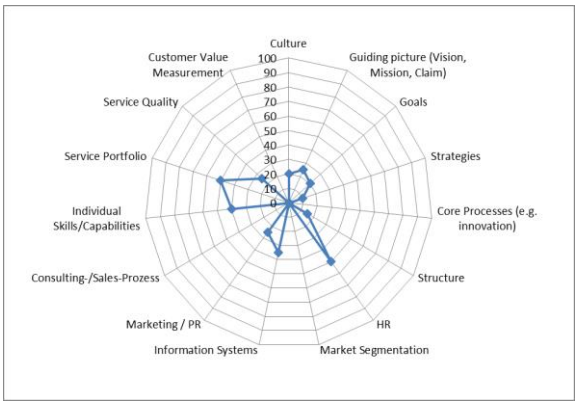


Figure 3. Customer value orientation status

D. Current customer value orientation status

We have created an initial catalogue of indicators to measure the current customer value orientation of IT-suppliers (see Figure 3. Customer value orientation status).

 ISBN: 978-1-61208-460-2

The results reveal the enormous potential for IT-suppliers to grow their businesses because typically, we encounter values below 50 for all success factors. Whenever we had the opportunity to conduct such measurement with members of different hierarchical levels within an organization, we could also observe that higher-ranking people would rate the orientation status more positively than the lower levels.

Finally, we found, with all IT-suppliers that they do not measure the customer value potentially obtained from using their solutions.

III. BUSINESS POTENTIAL REFERENCE MODEL

A "business potential" is a chance to generate a higher operating cash flow, either by reducing outgoing cash flows or increasing incoming ones. Investing and financing cash flows are designed to leverage the operating cash flow. Therefore, they are not important to the genuine business potential. Basically there are inhibitors (I) from I1 to In, preventing this currently happening. To unleash the business potential, these inhibitors need to be addressed individually and appropriately with enablers (E) from E1 to En. The total monetary value of both inhibitors and enablers can be calculated with the following formula: resources x time x value (in any freely convertible currency). This way, the total value of a business potential is the sum of the values that each Inhibitor (I1 to In) generates on the one side. On the other side, the total value of the offering is the sum of the values that each enabler (E1 to En) generates. Altogether, this constitutes a solution (see Figure 4. Business potential reference model).

The real value of this solution is the total value of the inhibitors minus the total value of the enablers. If this value is positive, it indicates that the solution generates a positive impact on the operating cash flow of the owner of the business potential. Ideally, this is the customer, but the IT-supplier should also be able to generate a higher operating cash flow for himself, if he is able to determine the customer value.

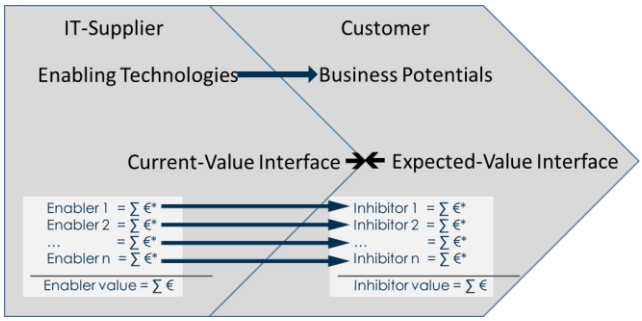


Figure 4. Business potential reference model

Looking at elements of a business potential, reveals the lever for making cloud services more effective and efficient. The customer business potential, with its current inhibitors, needs to be revealed and examined, making this the common denominator between IT-suppliers and customers. By contrast, our observation from over 12 years of working with over 11.000 people from over 450 IT-suppliers, is that suppliers and customers use technologies as a common denominator, although none of them has properly analyzed the inhibitors or even a defined business potential as a target. Looking at the web sites and marketing collateral of IT-suppliers and requests for proposals (RFP) or requests for information (RFI) from customers, it is evident that this has not changed at all over time. We will not try to figure out which side is responsible for such dysfunctional behavior. Instead, we will establish the business potential, with its inhibitors and enablers, as the most effective common denominator. Understanding, analyzing and quantifying the individual inhibitors of a business potential is crucial to defining appropriate individual enablers and to specifically addressing the inhibitors. Only this way can both sides of a solution be measured and controlled. Finally, a solution is intended to help the customer generate more operating cash flow and hence increase his competitiveness. The advantage for the supplier would be a value-added pricing approach.

We conclude that the current state of play requires both parties to fundamentally change the way they approach and create cloud services as intended solutions. A challenge is that leaders on both sides are not prepared. Neither transformational leadership, nor transformation methodology, nor innovation methodology, nor innovation processes, were taught at university or other training to the majority of people who are currently in leading positions on either side. There is a massive skill gap to close.

IV. TRANSFORMATION PROCESS

IT-suppliers are able to manage the complexity of IT, but are still unable to map it to customers´ business potentials, because they do not have insight into their target customers´ business processes. Fundamentally changing their approach to innovation management (creating, communicating/selling, implementing and controlling) of Cloud-Services is what we perceive as the core of the so-called cloud transformation, in order to boost cloud economics in our target countries.

A. The need for innovation processes

Cloud technology itself is not a solution and does not provide value. Identifying business challenges or desired results is essential before modifying technologies. IT-suppliers in our focus countries are still technology oriented. Innovative is, ex post, what has successfully penetrated a market. We recommend integrating the customers into the innovation process (Open Innovation) of cloud services. According to our conversations with the executives of IT-suppliers, this needs to be the focus of our research. From over 300 CEO conversations, we have observed that IT-suppliers do not have an innovation process in place. It may seem incredible, but while process management and process modeling techniques [10] have impacted on a lot of processes even in the academic world (see Figure 5. The Nobel price process), for example, by making them visible,

 ISBN: 978-1-61208-460-2

repeatable, measurable and controllable, the innovation process still evidently remains a black box.

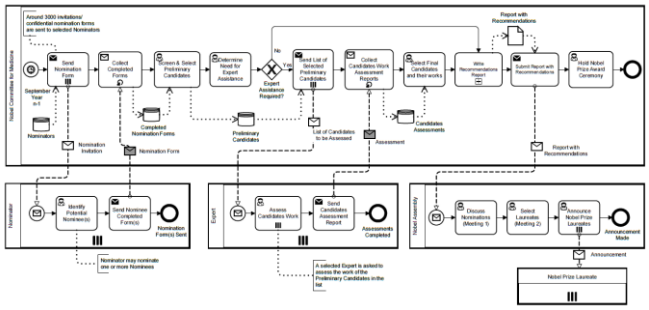


Figure 5. The Nobel price process [9]

In the literature, innovation processes [11][12] are only described as a series of specified sub-processes [13] and rarely end-to end [14][15], which ultimately contributes to failure in innovation [16][17]. There is a demand for new management priorities [18], management responsibilities [14], management models [19], core processes [20][21][22], process activities, process roles and process resources [16][23].

Designing a simple innovation process that IT-suppliers can use to create cloud-based hybrid service bundles is a research gap and thus the subject of our future research and publications.

B. Conducting successful transformation

The term *change* is used so variously in the literature that some misunderstanding is inevitable. There is no copyright on expressions like “Organizational Development”, “Evolution”, “Change Management” or “Transformation Management” [1][7]. An early version of a change management model introduced in 1947 proposed that change in an organization requires unfreezing, changing/moving and (re)freezing [4]. Although the theory was criticized for being too simplistic and for not being operationalizable, Lewin´s model still is relevant and can be found at the cores of many major approaches to change.

We will therefore introduce the major concepts (Transformation management, Enterprise Transformation and Transformation of Management) before introducing our own transformation model. In addition, the success of a transformation is fundamentally determined by its extent. Too big could make it too long and costly, whereas too small could mean failing to achieve the objectives. In general, there are three types of transformation (see Figure 6. Types of transformation)

- Type 1: Transforming the organization as an entity
- Type 2: Having a small team pilot and role model the transformation within the organization
- Type 3: Establishing a new entity outside the organization and starting from scratch [5].

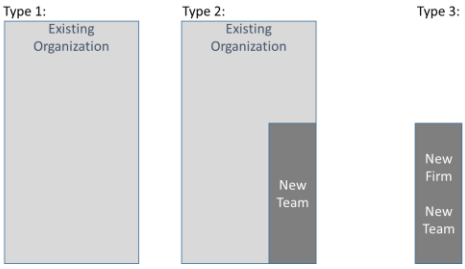


Figure 6. Types of transformation

We will have to analyze and develop recommendations, and determine with IT-suppliers which type of transformation will be most effective/efficient and under which circumstances. Also, we would need to define the three dimensions of transformation, *Ends*: Describing objectives that need to be achieved through initiatives, *Means*: Describing resources needed to achieve objectives, *Scope*: Describing the extent and direction of the transformation (see Figure 7. Dimensions of transformation)

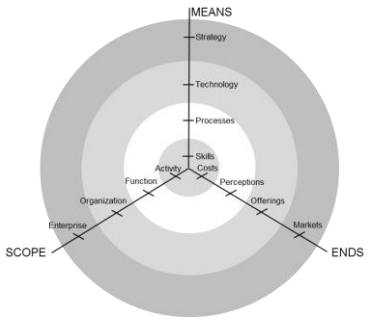


Figure 7. Dimensions of transformation [6]

The transformation approaches in both the literature and in practice have matured over the past decades. Nevertheless, leading or managing transformation is still not common in the curricula of graduate and post-graduate studies. When conducting transformations in the past, the author has identified this issue and already introduced a new stage in Kotter´s transformation process [3] “Stage 3: Empowering the guiding coalition” (Figure 8). Educating the key people in transformation methodology and about the original core of their transformation, has already helped hundreds of people to drive their transformations successfully.

Figure 8. The 9-Stage process [8]

 ISBN: 978-1-61208-460-2

V. SUMMARY AND RECOMMENDATIONS

Customers of cloud solutions want to achieve positive, monetarily measurable results. Making them aware of their business potentials would leverage the sale and use of cloud solutions. IT-suppliers do yet not speak the language of business. They are talking bits and bytes, which currently leads to poor business in the focal area. Business potential is the most effective common denominator for customers and suppliers of cloud solutions. The major objective of our research is therfore to create a methodology and tools, enabling IT-suppliers to conduct their transformation successfully, which will in turn contribute to driving cloud economics.

VI. FUTURE RESEARCH

On the one hand, we strongly recommend conducting future research on innovation processes in order to enable customers and suppliers to measurably and repetitively create innovations. An alignment of these processes would together lead to better and faster results. On the other hand, it is questionable whether the majority of suppliers and customers are really able to transform with a reasonable scope of objectives, within a reasonable period of time and at an acceptable level of costs. The critical element is time to market. Therefore, we recommend future research in methodology and cloud-based technology to help both parties exploit business potential as the most effective common denominator.

REFERENCES

[1] A. Janes et al., “Transformation-Management – Changing Organisations Inside Out,“ Vienna: Springer, 2001.

[2] W. B. Rouse, “Enterprise Transformation - Understanding and Enabling Fundamental Change,” Wiley, 2006.

[3] J. P. Kotter, “Leading Change,” Boston: Harvard Business Review Press, 1996.

[4] K. Lewin, “Change Management Model,” New York: McGraw Hill.

[5] J. P. Kotter, “Accelerate: Building Strategic Agility for a Faster Moving World (XLR8),” Boston: Harvard Business Review Press, 2013.

[6] W. B. Rouse, “Enterprise Transformation - Understanding and Enabling Fundamental Change,” Wiley, 2006, p. 39.

[7] R. Lessem et al., “Transformation Management - Towards the Integral Enterprise,” Farnham: Gower, 2009.

[8] F. Keßler, “Enterprise Transformation Management – Development of a Concept of Customer Value Orientation for IT_Companies, B2B, as a Path to Longterm Profitable Growth ”, 2011.

[9] Author not named, in http://www.omg.org/spec/BPMN/20100601/10-06-02.pdf, p. 26, retr.: Feb. 2016

[10] G. Schewe et al., „Innovation for Medium Size Companies – A Process Oriented Guideline for SME,“ Wiesbaden: Gabler, 2009, pp. 124.

[11] Fraunhofer Institut für Arbeitswissenschaft und Organisation “Cross-Industry Innovation,“ in http://wiki.iao.fraunhofer.de/index.php/Cross_Industry-Innovation, 2012, retrieved: Oct. 2012.

[12] Author not named, Gabler Wirtschaftslexikon, “Open Innovation,“ in http://wirtschaftslexikon.gabler.de/Archiv/81584/open-innovation-v3.html, 2012, retrieved: Oct. 2012.

[13] B. Wirtz, “Business Model Management – Design – Instrumente – Success Factors of Business Models,“ Wiesbaden: Gabler, 2011, pp. 214.

[14] J. Hauschildt et al., “Innovation Management,“ 5. überarb., erg. und akt. Aufl., Munich: Vahlen, 2011.

[15] T. Stern et al., “Successful Innovation Management – Success Factors, Basic Patterns, Cases,“ 4., überarb. Aufl., Wiesbaden: Gabler, 2010, p. 18.

[16] M. Disselkamp, “Innovation Management - Instruments and Methods for Implementation in Companies,“ 2., überarb. Aufl., Munich: Springer Gabler, 2012, pp. 56.

[17] M. Zollenkop, “Business Model Innovation,“ Wiesbaden: Deutscher Universitäts-Verlag, 2006, p. 49.

[18] T. Bieger et al., “Innovative Business Models - Fundamental Concepts, Creative Areas and Entrepreneurial Practice,“ Heidelberg: Springer, 2011, pp. 81.

[19] N. Tomaschek et al., “Management & Consulting in Transformation- und Innovation-Processes - Research in Progress,“ Munich: Rainer Hampp, 2010.

[20] C. Hentschel, et al., “TRIZ – Systematic Innovation,“ Munich: Hanser, 2010.

[21] B. Maurer et al., “World Champions in Innovation – How our Companies become Unbeatable,“ Weinheim: Wiley, 2011.

[22] J. O. Meissner, “Introduction to systemic Innovation-Management,“ Heidelberg: Carl-Auer-Systeme, 2011.

[23] O. Gassmann et al., “Practical Knowledge Innovation Management – From Idea to Market Success,“ 2. erw. und überarb. Aufl., München: Hanser, 2011.

[24] H. Österle et al., “Memorandum for Creation Oriented Wirtschaftsinformatik, in Österle et al.: Creation Oriented Wirtschaftsinformatik: A Pleadoyer for Rigor und Relevance,“ infowerk Verlag, 2010, pp. 1 – 6.

[25] A. R. Hevner et al., “Design Science in Information Systems Research,” in MIS Quarterly, Vol. 28, No. 1, (2004)3

[26] A. R. Hevner, “A Three Cycle View of Design Science Research,” Scandinavian Journal of Information Systems, 19(2007)2, pp. 87 – 92.

[27] M. Yamartino, “The Economics Of The Cloud,” in http://news.microsoft.com/download/archived/presskits/cloud/docs/the-economics-of-the-cloud.pdf, 2010, (retr. Feb. 2016)

[28] H. Okin, “Cloud Economics,” in https://www.kpmg-institutes.com/institutes/advisory-institute/events/2014/09-/podcast-cloud-economics.html, 2014, retr.: Feb. 2016

[29] D. Reeves et al., “The Truth About Cloud Economics,” in https://hbr.org/2012/04/the-truth-about-cloud-economic/, 2012, retr.: Feb. 2016

[30] o. V. TechTarget, in http://searchcio.techtarget.com/definition/ cloud-economics, 2013, retr.: Feb. 2016

 ISBN: 978-1-61208-460-2

How to Synchronize Large-Scale Ultra-Fine-Grained Processors in Optimum-Time

Hiroshi Umeo
School of Information Engineering
University of Osaka Electro-Communication
Neyagawa-shi, Hastu-cho, 18-8, Osaka
Email: umeo@cyt.osakac.ac.jp

***Abstract*—We introduce a new class of FSSP (firing squad synchronization problem) algorithms based on recursive-halving and construct a survey on recent developments in FSSP algorithms for one-dimensional cellular arrays. We present herein a comparison of the quantitative aspects of the optimum-time FSSP algorithms developed so far. Several state-efficient new implementations and new insights into synchronization algorithms and multi-dimensional expansions are also given.**

***Keywords*—cellular automata, synchronization**

I. INTRODUCTION

Synchronization of large-scale networks is an important and fundamental computing primitive in parallel and distributed systems. The synchronization in ultra-fine grained parallel computational model of cellular automata, known as firing squad synchronization problem (FSSP), has been studied extensively for more than fifty years, and a rich variety of synchronization algorithms has been proposed. In the present paper, we introduce a new class of FSSP algorithms based on recursive-halving and construct a survey on recent developments in FSSP algorithms for one-dimensional cellular arrays. The algorithms being compared are Balzer [1], Gerken [2], Waksman [20], a number of revised versions thereof, and their generalized versions such as Moore and Langdon [8], Settle and Simon [10], Szwerinski [11], all included in the proposed new class of FSSP algorithms. We present herein a survey and a comparison of the quantitative aspects of the optimum-time synchronization algorithms developed so far. Several state-efficient new implementations, new insights into synchronization algorithms and multi-dimensional expansions are also given.

Specifically, we attempt to answer the following questions:

- First, are all previously presented transition rule sets correct?
- Do these sets contain redundant rules? If so, what is the exact rule set?
- How do the algorithms compare with each other?
- Can we expand those 1D FSSP algorithms proposed so far to 2D, 3D arrays, or more generally to multi-dimensional arrays?
- How can we synchronize multi-dimensional arrays in optimum-time?

In Section 2 we give a description of the FSSP and review some basic results on the FSSP for 1D arrays. Section 3 introduces a new class of FSSP algorithms based on recursive-halving and presents multi-dimensional generalizations of the algorithms. In the last section we give a summary of the paper.

II. FIRING SQUAD SYNCHRONIZATION PROBLEM

In this section, we define the FSSP and introduce some basic results on FSSP.

A. Firing Squad Synchronization Problem

Figure 1 illustrates a finite one-dimensional (1D) cellular array consisting of n cells. Each cell is an identical finite-state automaton. The array operates in lock-step mode in such a way that the next state of each cell is determined by both its own present state and the present states of its left and right neighbors. All cells (*soldiers*), except one *general*, are initially in the quiescent state at time $t = 0$ with the property that the next state of a quiescent cell with quiescent neighbors is the quiescent state again. At time $t = 0$, one general cell C_1 is in the *fire-when-ready* state, which is the initiation signal for the synchronization of the array. The FSSP is to determine a description (state set and next-state function) for cells that ensures all cells enter the *fire* state at exactly the same time and for the first time. The set of states and the next-state function must be independent of n.

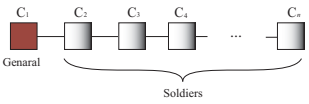


Fig. 1. One-dimensional cellular automaton

A formal definition of the FSSP is as follows: A cellular automaton $\mathcal{M}$ is a pair $\mathcal{M} = (\mathcal{Q}, \delta)$, where

1) $\mathcal{Q}$ is a finite set of states with three distinguished states `G`, `Q`, and `F`. `G` is an initial general state, `Q` is a quiescent state, and `F` is a firing state, respectively.
2) δ is a next state function such that $\delta : \mathcal{Q} \cup \{*\} \times \mathcal{Q} \times \mathcal{Q} \cup \{*\} \rightarrow \mathcal{Q}$. The state $\mathtt{*} \notin \mathcal{Q}$ is a pseudo state of the border of the array.
3) The quiescent state `Q` must satisfy the following conditions: $\delta(\mathtt{Q}, \mathtt{Q}, \mathtt{Q}) = \delta(*, \mathtt{Q}, \mathtt{Q}) = \delta(\mathtt{Q}, \mathtt{Q}, *) = \mathtt{Q}$.

A cellular automaton of length n, $\mathcal{M}_n$ consisting of n copies of $\mathcal{M}$ is a 1D array of $\mathcal{M}$, numbered from 1 to n.

 ISBN: 978-1-61208-460-2

Each $\mathcal{M}$ is referred to as a cell and denoted by C_i, where $1 \leq i \leq n$. We denote a state of C_i at time (step) t by $\mathtt{S}_i^t$, where $t \geq 0, 1 \leq i \leq n$. A *configuration* of $\mathcal{M}_n$ at time t is a function $\mathcal{C}^t : [1, n] \rightarrow \mathcal{Q}$ and denoted as $\mathtt{S}_1^t \mathtt{S}_2^t$ .... $\mathtt{S}_n^t$. A *computation* of $\mathcal{M}_n$ is a sequence of configurations of $\mathcal{M}_n$, $\mathcal{C}^0$, $\mathcal{C}^1$, $\mathcal{C}^2$, ..., $\mathcal{C}^t$, ..., where $\mathcal{C}^0$ is a given initial configuration. The configuration at time $t+1$, $\mathcal{C}^{t+1}$, is computed by synchronous applications of the next transition function δ to each cell of $\mathcal{M}_n$ in $\mathcal{C}^t$ such that:

$\mathtt{S}_1^{t+1} = \delta(*, \mathtt{S}_1^t, \mathtt{S}_2^t)$, $\mathtt{S}_i^{t+1} = \delta(\mathtt{S}_{i-1}^t, \mathtt{S}_i^t, \mathtt{S}_{i+1}^t)$, and $\mathtt{S}_n^{t+1} = \delta(\mathtt{S}_{n-1}^t, \mathtt{S}_n^t, *)$.

A *synchronized configuration* of $\mathcal{M}_n$ at time t is a configuration $\mathcal{C}^t$, $\mathtt{S}_i^t = \mathtt{F}$, for any $1 \leq i \leq n$.

The FSSP is to obtain an $\mathcal{M}$ such that, for any $n \geq 2$,

1) A synchronized configuration at time $t = T(n)$, $\mathcal{C}^{T(n)} = \overbrace{\mathtt{F}, \cdots, \mathtt{F}}^{n}$ can be computed from an initial configuration $\mathcal{C}^0 = \mathtt{G}\overbrace{\mathtt{Q}, \cdots, \mathtt{Q}}^{n-1}$.
2) For any t, i such that $1 \leq t \leq T(n) - 1$, $1 \leq i \leq n, \mathtt{S}_i^t \neq \mathtt{F}$.

The generalized FSSP (GFSSP) is to obtain an $\mathcal{M}$ such that, for any $n \geq 2$ and for any k such that $1 \leq k \leq n$,

1) A synchronized configuration at time $t = T(n)$, $\mathcal{C}^{T(n)} = \overbrace{\mathtt{F}, \cdots, \mathtt{F}}^{n}$ can be computed from an initial configuration $\mathcal{C}^0 = \overbrace{\mathtt{Q}, \cdots, \mathtt{Q}}^{k-1}\mathtt{G}\overbrace{\mathtt{Q}, \cdots, \mathtt{Q}}^{n-k}$.
2) For any t, i, such that $1 \leq t \leq T(n) - 1$, $1 \leq i \leq n, \mathtt{S}_i^t \neq \mathtt{F}$.

No cells fire before time $t = T(n)$. We say that $\mathcal{M}_n$ is synchronized at time $t = T(n)$ and the function $T(n)$ is a time complexity for the synchronization.

B. A Brief History of the Developments of Optimum-Time FSSP Algorithms

The problem known as the *firing squad synchronization problem* was devised in 1957 by J. Myhill, and first appeared in print in a paper by E. F. Moore [7]. This problem has been widely circulated, and has attracted much attention. The firing squad synchronization problem first arose in connection with the need to simultaneously turn on all parts of a self-reproducing machine. The problem was first solved by J. McCarthy and M. Minsky who presented a $3n$-step algorithm. In 1962, the first optimum-time, i.e., $(2n-2)$-step, synchronization algorithm was presented by Goto [3], with each cell having several thousands of states. Waksman [20] presented a 16-state optimum-time synchronization algorithm. Afterward, Balzer [1] and Gerken [2] developed an eight-state algorithm and a seven-state synchronization algorithm, respectively, thus decreasing the number of states required for the synchronization. In 1987, Mazoyer [5] developed a six-state synchronization algorithm which, at present, is the algorithm having the fewest states.

C. Complexity Measures for FSSP Algorithms

- **Time**
  Any solution to the original FSSP with a general at one end can be easily shown to require $(2n-2)$ steps for synchronizing n cells, since signals on the array can propagate no faster than one cell per one step, and the time from the general's instruction until the final synchronization must be at least $2n-2$.

  Theorem 1 The minimum time in which the firing squad synchronization could occur is $2n-2$ steps, where the general is located on the left end.

  Theorem 2 There exists a cellular automaton that can synchronize any 1D array of length n in optimum $2n-2$ steps, where the general is located on the left end.

- **Number of States**
  The following three distinct states: the *quiescent* state, the *general* state, and the *firing* state, are required in order to define any cellular automaton that can solve the FSSP. Note that the boundary state for C_0 and C_{n+1} is not generally counted as an internal state. Balzer [1] and Sanders [9] showed that no four-state optimum-time solution exists. Umeo and Yanagihara [17], Yunès [21], and Umeo, Kamikawa, and Yunès [14] gave some 5- and 4-state *partial* solutions that can solve the synchronization problem for infinitely many sizes n, but not all, respectively. The solution is referred to as *partial* solution, which is compared with usual *full* solutions that can solve the problem for all cells. Concerning the optimum-time full solutions, Waksman [20] presented a 16-state optimum-time synchronization algorithm. Afterward, Balzer [1] and Gerken [2] developed an eight-state algorithm and a seven-state synchronization algorithm, respectively, thus decreasing the number of states required for the synchronization. Mazoyer [5] developed a six-state synchronization algorithm which, at present, is the algorithm having the fewest states for 1D arrays.

  Theorem 3 There exists a 6-state *full* solution to the FSSP.

  Theorem 4 There is no four-state *full* solution that can synchronize n cells.

  Yunès [21] and Umeo, Yunès, and Kamikawa [14] developed 4-state partial solutions based on Wolfram's rules 60 and 150. They can synchronize any array/ring of length $n = 2^k$ for any positive integer k. Details can be found in Yunès [21] and Umeo, Kamikawa, and Yunès [14].

  Theorem 5 There exist 4-state *partial* solutions to the FSSP.

- **Number of Transition Rules**
  Any k-state (excluding the boundary state) transition table for the synchronization has at most $(k-1)k^2$ entries in $(k-1)$ matrices of size $k \times k$. The number of transition rules reflects a complexity of synchronization algorithms.

 ISBN: 978-1-61208-460-2

- **State-Change Complexity**
  Vollmar [18,19] introduced a *state-change complexity* in order to measure the efficiency of cellular automata, motivated by energy consumption in certain SRAM-type memory systems. The state-change complexity is defined as the sum of *proper* state changes of the cellular space during the computations. A formal definition is as follows: Consider an FSSP algorithm operating on n cells. Let $T(n)$ be synchronization steps of the FSSP algorithm. We define a matrix C of size $T(n) \times n$ ($T(n)$ rows, n columns) over $\{0, 1\}$, where each element $c_{i,j}$ on ith row, jth column of the matrix is defined:

$$c_{i,j} = \begin{cases} 1 & \mathrm{S}_i^j \neq \mathrm{S}_i^{j-1} \\ 0 & otherwise. \end{cases} \quad (1)$$

  The state-change complexity $SC(n)$ of the FSSP algorithm is the sum of 1's elements in C defined as:

$$SC(n) = \sum_{j=1}^{T(n)} \sum_{i=1}^{n} c_{i,j}. \quad (2)$$

  Vollmar [19] showed that $\Omega(n \log n)$ state-changes are required for synchronizing n cells in $(2n-2)$ steps.

  Theorem 6 $\Omega(n \log n)$ state-change is necessary for synchronizing n cells in minimum-steps.

  Gerken [2] presented a minimum-time, $\Theta(n \log n)$ minimum-state-change FSSP algorithm with a general at one end.

  Theorem 7 $\Theta(n \log n)$ state-change is sufficient for synchronizing n cells in $2n-2$ steps.

III. A Class of FSSP Algorithms Based on Recursive-Halving

Here we introduce a new class of FSSP algorithms based on recursive halving.

A. Recursive-Halving Marking

In this section, we develop a marking schema for 1D arrays referred to as *recursive-halving marking*. The marking schema prints a special mark on cells in a cellular space defined by the recursive-halving marking. It is based on a 1D FSSP synchronization algorithm. The marking will be effectively used for constructing multi-dimensional FSSP algorithms operating in optimum-time.

Let S be a 1D cellular space consisting of cells C_i, C_{i+1}, ..., C_j, denoted by $[i...j]$, where $j > i$. Let $|S|$ denote the number of cells in S, that is $|S| = j - i + 1$. A center cell(s) C_x of S is defined by

$$x = \begin{cases} (i+j)/2 & |S|\text{: odd} \\ (i+j-1)/2, (i+j+1)/2 & |S|\text{: even.} \end{cases} \quad (3)$$

The recursive-halving marking for a given cellular space S = [1...n] is defined as follows:

Recursive-Halving Marking: RHM

Algorithm RHM(S)
begin
  if $|S| \geq 2$ **then**
    if $|S|$ is odd **then**
      mark center cell C_x in S
      S_L:= [1...x]; S_R:= [x...n]
      RHM$_\mathrm{L}$(S_L); **RHM$_\mathrm{R}$(S_R)**;
    else
      mark center cells C_x and C_{x+1} in S
      S_L:= [1...x]; S_R:= [$x+1$...n]
      RHM$_\mathrm{L}$(S_L); **RHM$_\mathrm{R}$(S_R)**;
end

Left-Side Recursive-Halving Marking: RHM$_\mathrm{L}$

Algorithm RHM$_\mathrm{L}$(S)
begin
  while $|S| > 2$ **do**
    if $|S|$ is odd **then**
      mark center cell C_x in S
      S_L:= [1...x]; **RHM$_\mathrm{L}$(S_L)**;
    else
      mark center cells C_x and C_{x+1} in S
      S_L:= [1...x]; **RHM$_\mathrm{L}$(S_L)**;
end

The marking RHM$_\mathrm{R}$ for the right-side space can be defined in a similar way. As an example, we consider a cellular space S = [1...15] consisting of 15 cells. The first center cell is C_8, then the second one is C_4, C_5 and C_{11}, C_{12}, and the last one is C_2, C_3, C_{13}, C_{14}, respectively. In case S = [1...17], we get C_9, C_5, C_{13}, C_3, C_{15}, and C_2, C_{16} after four iterations.

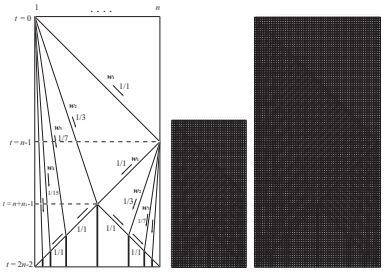


Fig. 2. Recursive-halving marking

Figure 2 (left) shows a space-time diagram for the marking. At time $t = 0$, the leftmost cell C_1 generates a set of signals $w_1, w_2, ..., w_k$, .., each propagating in the right direction at $1/(2^k - 1)$ speed, where $k = 1, 2, 3, ...,$. The $1/1$-speed signal w_1 arrives at C_n at time $t = n - 1$. Then, the rightmost cell C_n also emits an infinite set of signals $w_1, w_2, ..., w_k$, .., each propagating in the left direction at $1/(2^k - 1)$ speed, where $k = 1, 2, 3, ...,$. The readers can find that each crossing of two signals, shown in Figure 2 (left), enables the marking at middle points defined by the recursive-halving. A finite state realization for generating the infinite set of signals above is a well-known technique employed in Balzer [1], Gerken [2], and Waksman [20] for the implementations of the optimum-time synchronization algorithms on 1D arrays.

We have developed a simple implementation of the recursive-halving marking on a 13-state, 314-rule cellular automaton. In Figure 2 (middle and right) we present several

snapshots for the marking on 42 and 71 cells, respectively. We have:

Lemma 8 There exists a 1D 13-state, 314-rule cellular automaton that can print the recursive-halving marking in any cellular space of length n in $2n-2$ steps.

An optimum-time complexity $2n-2$ needed for synchronizing cellular space of length n in the classical WBG-type (Waksman [20], Balzer [1], and Gerken [2]) FSSP algorithms can be interpreted as follows: Let S be a cellular space of length $n = 2n_1+1$, where $n_1 \geq 1$. The first center mark in S is printed on cell C_{n_1+1} at time $t_{1\mathrm{D-center}} = 3n_1$. Additional n_1 steps are required for the markings thereafter, yielding a final synchronization at time $t_{1\mathrm{D-opt}} = 3n_1 + n_1 = 4n_1 = 2n-2$. In the case $n = 2n_1$, where $n_1 \geq 1$, the first center mark is printed simultaneously on cells C_{n_1} and C_{n_1+1} at time $t_{1\mathrm{D-center}} = 3n_1 - 1$. Additional $n_1 - 1$ steps are required for the marking thereafter, yielding the final synchronization at time $t_{1\mathrm{D-opt}} = 3n_1 - 1 + n_1 - 1 = 4n_1 - 2 = 2n - 2$.

$$t_{1\mathrm{D-center}} = \begin{cases} 3n_1 & |S| = 2n_1 + 1, \\ 3n_1 - 1 & |S| = 2n_1. \end{cases} \quad (4)$$

Thus, additional t_{sync} steps are required for the synchronization for a cellular space with the recursive-halving marks:

$$t_{\mathrm{sync}} = \begin{cases} n_1 & |S| = 2n_1 + 1, \\ n_1 - 1 & |S| = 2n_1. \end{cases} \quad (5)$$

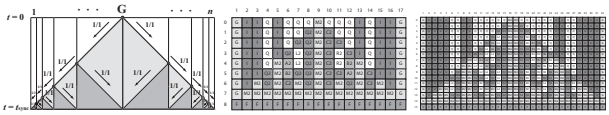


Fig. 3. Synchronization based on recursive-halving

In this way, it can be easily seen that any cellular space of length n with the recursive-halving marking initially with a general on a center cell or two generals on adjacent center cells can be synchronized in $\lceil n/2 \rceil - 1$ optimum-steps.

Lemma 9 Any 1D cellular space S of length n with the recursive-halving marking initially with a general(s) on a center cell(s) in S can be synchronized in $\lceil n/2 \rceil - 1$ optimum-steps.

In Figure 3, we illustrate a space-time diagram for synchronizing a cellular space with recursive-halving marking (left) and some snapshots for the synchronization on 17 (middle) and 32 (right) cells, respectively.

As was seen, the first marking of center cell(s) plays an important role. In order to use the marking scheme for the design of multi-dimensional FSSP algorithms, we print a special mark only on the first center cell(s) of a given cellular space, where the center cells thereafter will be marked with a different symbol from the first one.

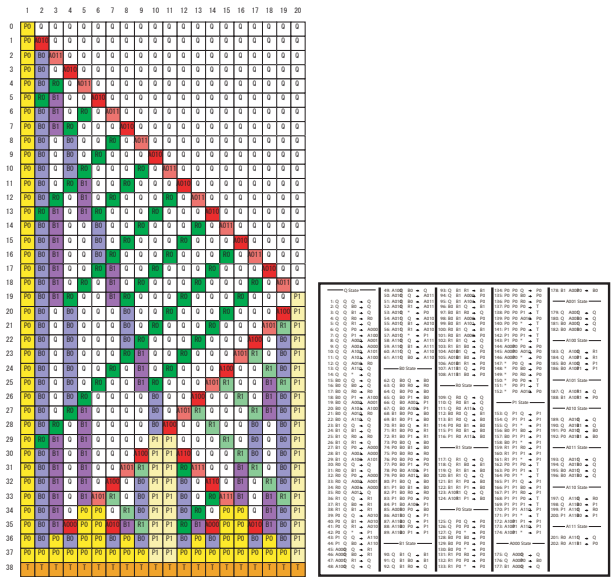


Fig. 4. Waksman's FSSP

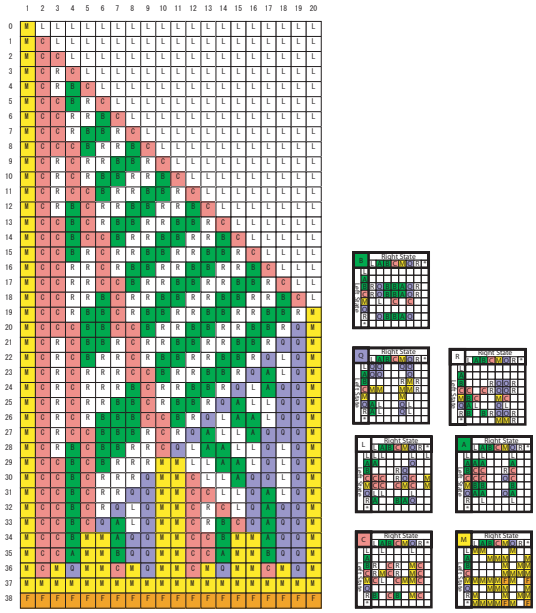


Fig. 5. Balzer's FSSP

B. Waksman's 16-state Algorithm

Waksman [20] proposed a 16-state firing squad synchronization algorithm. Umeo, Hisaoka, and Sogabe [13] corrected all errors in Waksman's original transition table. In Figure 4, we give a snapshot of the synchronization processes on 20 cell and a list of transition rules for Waksman's algorithm. The list is a revised version presented in Umeo, Hisaoka, and Sogabe [13]. The state-change complexity of the algorithm is $O(n^2)$.

C. Balzer's Eight-state Algorithm

Balzer [1] constructed an eight-state, 182-rule synchronization algorithm and the structure of which is completely identical to that of Waksman [20]. In Figure 5, we give

 ISBN: 978-1-61208-460-2

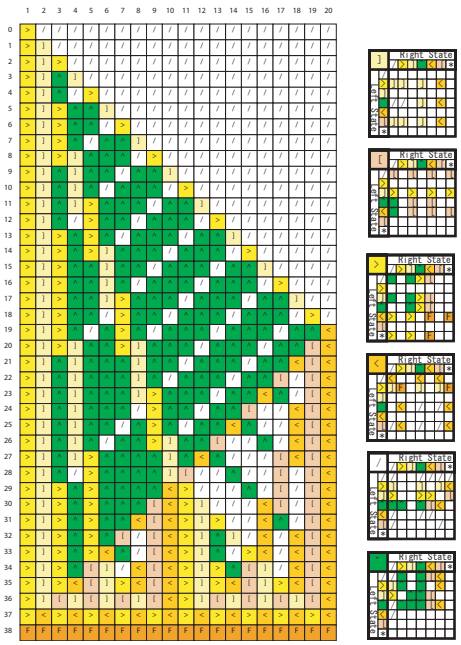


Fig. 6. Gerken's FSSP

a snapshot of the algorithm and a list of transition rules for Balzer's algorithm. The state-change complexity of the algorithm is $O(n^2)$.

D. Gerken's Seven-state Algorithm

Gerken [2] reduced the number of states realizing Balzer's algorithm and constructed a seven-state, 118-rule synchronization algorithm. In Figure 6, we give a list of the transition rules for Gerken's algorithm and its snapshots. The state-change complexity of the algorithm is $O(n^2)$.

E. An Optimum-Time 2D FSSP Algorithm $\mathcal{A}$

We assume that an initial general G is on the north-west corner cell $C_{1,1}$ of a given array of size $m \times n$. The algorithm consists of three phases: a marking start phase for 2D arrays, pre-synchronization phase and a final synchronization phase. An overview of the 2D synchronization algorithm $\mathcal{A}$ is as follows:

Step 1. Start the recursive-halving marking for cells on each row and column, **find** a *center cell(s)* of the 2D array, and **generate** a new general(s) on the center cell(s). Note that a crossing(s) of the center column(s) with the center row(s) is a center cell(s) of the array.

Step 2. Pre-synchronize the center column(s) using Lemma 6, which is initiated by the general in step 1. Every cell(s) on the center column(s) acts as a general at the next Step 3.

Step 3. Synchronize each row using Lemma 6, initiated by the general generated in Step 2. This yields the final synchronization of the array.

Thus, we have:

Theorem 10 The synchronization algorithm $\mathcal{A}$ can synchronize any $m \times n$ rectangular array in optimum $m + n + \max(m, n) - 3$ steps.

Theorem 11 There exists an optimum-time synchronization algorithm that can synchronize any three-dimensional array of size $m \times n \times \ell$ with a general at $C_{1,1,1}$ in optimum $m + n + \ell + \max(m, n, \ell) - 4$ steps.

Theorem 12 There exists an optimum-time synchronization algorithm that can synchronize any kD array of size $n_1 \times n_2 \times ... \times n_k$ with a general at $C_{1,1,...,1}$ in optimum $n_1 + n_2 +, ..., +n_k + \max(n_1, n_2, ..., n_k) - k - 1$ steps.

F. A Comparison of Quantitative Aspects of Optimum-Time Synchronization Algorithms

Here, we present a table based on a quantitative comparison of optimum-time synchronization algorithms and their transition tables discussed above with respect to the number of internal states of each finite state automaton, the number of transition rules realizing the synchronization, and the number of state-changes on the array.

TABLE I. COMPARISON OF FSSP ALGORITHMS

Algorithm	# of states	# of transition rules	State change complexity
Goto [3]	many thousands	—	$\Theta(n \log n)$
Waksman [20]	16	202	$O(n^2)$
Balzer [1]	8	165	$O(n^2)$
Gerken I [2]	7	105	$O(n^2)$
Mazoyer [5]	6	119	$O(n^2)$
Gerken II [2]	155	2371	$\Theta(n \log n)$

G. O(1)-bit vs. 1-bit Communication FSSP

In the study of cellular automata, the amount of bit-information exchanged at one step between neighboring cells has been assumed to be O(1)-bit data. An O(1)-bit CA is a conventional CA in which the number of communication bits exchanged at one step between neighboring cells is assumed to be O(1)-bit, however, such inter-cell bit-information exchange has been hidden behind the definition of conventional automata-theoretic finite state description. On the other hand, the 1-bit inter-cell communication model is a new CA in which inter-cell communication is restricted to 1-bit data, referred to as the 1-bit CA model. The number of internal states of the 1-bit CA is assumed to be finite in the usual sense. The next state of each cell is determined by the present state of that cell and two binary 1-bit inputs from its left and right neighbor cells. Thus, the 1-bit CA can be thought of as one of the most powerless and the simplest models in a variety of CA's. A precise definition of the 1-bit CA can be found in Umeo and Yanagihara [17]. Umeo and Yanagihara [17] constructed an optimum-time synchronization algorithm on a 1-bit CA model, based on Waksman's algorithm. In Figure 7, we show a configuration of the 1-bit synchronization algorithm on 15 cells. Each cell has 78 internal states and 208 transition rules. The small black triangles ▶ and ◀ indicate a 1-bit signal transfer in the right or left direction, respectively, between neighboring cells. A symbol in a cell shows internal state of the cell.

[Theorem 13] There exists a 1-bit CA that can synchronize n cells in optimum $2n - 2$ steps.

 ISBN: 978-1-61208-460-2

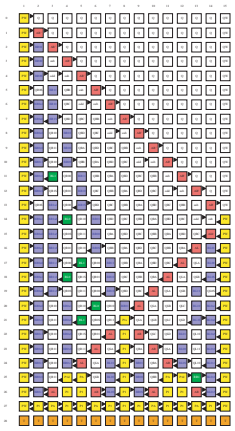


Fig. 7. FSSP on 1-bit CA

IV. Conclusion and Future Work

In the present paper, we have given a survey on recent developments in FSSP algorithms for one-dimensional cellular arrays. We focus our attention on a new class of FSSP algorithms based on recursive-halving. It is shown that the recursive-halving marking has been used in the design of many optimum-time FSSP algorithms and can be generalized and expanded to multi-dimensional arrays. Several multi-dimensional generalizations of the algorithms are also given. As a future work an FSSP for growing multi-dimensional arrays would be interesting.

References

[1] R. Balzer: An 8-state minimal time solution to the firing squad synchronization problem. *Information and Control*, vol. 10, pp. 22-42, 1967.

[2] Hans-D. Gerken: Über Synchronisations - Probleme bei Zellularautomaten. *Diplomarbeit*, Institut für Theoretische Informatik, Technische Universität Braunschweig, pp. 50, 1987.

[3] E. Goto: A minimal time solution of the firing squad problem. *Dittoed course notes for Applied Mathematics* 298, Harvard University, pp. 52-59, with an illustration in color, 1962.

[4] J. Mazoyer: An overview of the firing squad synchronization problem. *Lecture Notes on Computer Science*, Springer-Verlag, vol. 316, pp. 82-93, 1986.

[5] J. Mazoyer: A six-state minimal time solution to the firing squad synchronization problem. *Theoretical Computer Science*, vol. 50, pp. 183-238, 1987.

[6] M. Minsky: *Computation: Finite and infinite machines*. Prentice Hall, pp. 28-29, 1967.

[7] E. F. Moore: The firing squad synchronization problem. in *Sequential Machines, Selected Papers* (E. F. Moore, ed.), Addison-Wesley, Reading MA., pp. 213-214, 1964.

[8] F. R. Moore and G. G. Langdon: A generalized firing squad problem. *Information and Control*, 12, pp. 212-220, 1968.

[9] P. Sanders: Massively parallel search for transition-tables of polyautomata. In *Proc. of the VI International Workshop on Parallel Processing by Cellular Automata and Arrays*, (C. Jesshope, V. Jossifov and W. Wilhelmi (editors)), Akademie, 99-108, 1994.

[10] A. Settle and J. Simon: Smaller solutions for the firing squad. *Theoretical Computer Science*, 276, 83-109, 2002.

[11] H. Szwerinski: Time-optimum solution of the firing squad synchronization problem for n-dimensional rectangles with the general at an arbitrary position. *Theoretical Computer Science*, vol. 19, pp. 305-320, 1982.

[12] H. Umeo: Firing squad synchronization problem in cellular automata. In *Encyclopedia of Complexity and System Science*, R. A. Meyers (Ed.), Springer, Vol.4, pp.3537-3574, 2009.

[13] H. Umeo, M. Hisaoka, and T. Sogabe: A survey on optimum-time firing squad synchronization algorithms for one-dimensional cellular automata. *Int. J. of Unconventional Computing*, vol. 1, pp.403-426, 2005.

[14] H. Umeo, N. Kamikawa, and J.-B. Yunès: A family of smallest symmetrical four-state firing squad synchronization protocols for ring arrays. *Parallel Processing Letters*, Vol.19, No.2, pp.299-313, 2009.

[15] H. Umeo, N. Kamikawa, K. Nishioka, and S. Akiguchi: Generalized Firing Squad Synchronization Protocols for One-Dimensional Cellular Automata - A Survey. *Acta Physica Polonica B, Proceedings Supplement*. Vol.3, pp.267-289, 2010.

[16] H. Umeo, K. Kubo, and K. Nishide: A class of time-optimum FSSP algorithms for multi-dimensional cellular arrays. *Communications in Nonlinear Science and Numerical Simulation*, 21, pp.200-209, 2015.

[17] H. Umeo and T. Yanagihara: Smallest implementations of optimum-time firing squad synchronization algorithms for one-bit-communication cellular automata. *Proc. of the 2011 International Conference on Parallel Computing and Technology, PaCT 2011*, LNCS 6873, pp. 210-223, 2011.

[18] R. Vollmar: On Cellular Automata with a Finite Number of State Change. *Computing, Supplementum*, vol. 3, pp. 181-191, 1981.

[19] R. Vollmar: Some remarks about the "Efficiency" of polyautomata. *International Journal of Theoretical Physics*, vol. 21, no. 12, pp. 1007-1015, 1982.

[20] A. Waksman: An optimum solution to the firing squad synchronization problem. *Information and Control*, vol. 9, pp. 66-78, 1966.

[21] J. B. Yunès: A 4-states algebraic solution to linear cellular automata synchronization. *Information Processing Letters*, Vol. 19, Issue 2, pp.71-75, 2008.

 ISBN: 978-1-61208-460-2

Dynamic Power Simulator Utilizing Computational Fluid Dynamics and Machine Learning for Proposing Task Allocation in a Data Center

Kazumasa Kitada*, Yutaka Nakamura†, Kazuhiro Matsuda‡ and Morito Matsuoka‡ §
*School of Information Science and Technology, Osaka University, Osaka, Japan
Email: k-kitada@ane.cmc.osaka-u.ac.jp
†Graduate School of Engineering Science, Osaka University, Osaka, Japan
Email: nakamura@is.sys.osaka-u.ac.jp
‡Cybermedia Center, Osaka University, Osaka, Japan
Email: kazuhiro.matsuda@ane.cmc.osaka-u.ac.jp
§Advanced Telecommunications Research Institute International, Kyoto, Japan
Email: matsuoka@cmc.osaka-u.ac.jp

***Abstract*—A dynamic power simulator for a data center was demonstrated by combining computational fluid dynamics (CFD) and machine learning. The total power consumption of the data center was simulated. The sensitivity of the temperature distribution along the virtual machine (VM) allocation was analyzed using a non-parametric process for the CFD. An allocation of server tasks was proposed for reducing the power consumption of the air conditioner installed in the data center. This simulation showed that the optimum operating temperature increases with the power usage effectiveness. These results indicate that the power simulator developed in this study is a powerful tool for dynamic power simulation and for estimation of better operation parameters, including VM allocation, from the aspect of power consumption.**

Keywords–Data center; power simulator; computational fluid dynamics; virtual machine allocation; machine learning.

I. INTRODUCTION

In recent years, many data centers have been built to support the increased popularity of services such as video on demand and cloud storage. In line with this trend, the power consumption of data centers has increased sharply [1]. As a result, the power consumption of data centers has become a serious social problem, and studies on energy-efficient data centers have attracted considerable attention [2]–[6]. Various proposals have been put forward for reducing the power consumption of data center equipment such as air conditioners, power supply units, and servers [7]–[9]. Considering the total power consumption of a server, Khuller et al. proposed an energy management system in which the workload of all servers was concentrated in a portion of the servers and the other servers were turned off [10]. Iyengar et al. improved the energy efficiency of data centers by configuring the fan speed of air conditioners or shutting off air conditioners according to the temperature distribution in the data center [11]. However, reducing the power consumption of individual units without coordination among them is insufficient for reducing the total power consumption of the data center, in which various types of equipment operate in a strongly interrelated way.

ICT equipment and air conditioners account for most of the power consumption in a data center [12]. Therefore, cooperative control, such as simultaneous optimization of the server workload assignment and air conditioners, is essential for improving the energy efficiency of the data center. The power consumption of each piece of equipment depends not only on its own operational conditions but also on those of others. For example, the server power consumption depends on the inlet air temperature, which is in turn affected by the air-conditioner setting. The server workload assignment and operation set points of the air conditioners must be controlled in a coordinated manner to minimize the total energy consumption of the data center. To reduce the total power consumption, it is necessary to predict the state of the data center, such as its temperature distribution, given by the operational conditions such as virtual machine allocation. Because the heat emitted from the servers accounts for most of the heat emitted from the data center, it is essential to predict the temperature of the exhaust heat from the servers for predicting the total power consumption of the data center. However, predicting the heat emitted from the servers is difficult because many factors are interdependent. For example, the temperature of the exhaust heat from the server depends on the air volume passing through the server and the server's power consumption. Moreover, as the intake temperature and air volume vary according to the position of the server in the data center, the temperature of the exhaust from the server changes with the location even when the same workload is assigned to servers with the same power consumption.

Computational fluid dynamics (CFD) is a representative technique for simulating the temperature distribution when various parameters are interrelated [13]–[15]. However, CFD estimates only the temperature distribution and normally does not simulate the power consumption directly.

In this study, we developed a novel simulator that estimates the power consumption of the data center. This simulator was demonstrated by combining the temperature distribution simulation using CFD and machine learning techniques to predict the power consumption of the server and air conditioner. Power consumption of the servers and the air conditioner was estimated by regression models, such as a neural network. Their parameters were trained so that the estimates fit the exact power consumption of the data center in operation. The air temperature and fan rotational speed of the air conditioner were measured as learning data to build the power consumption model of the air conditioners. The procedure for estimating the total power consumption of the data center is as follows. First, the exhaust temperature for the servers is simulated using a CFD simulator with a task. Then, the total power consumption of the data center is calculated as a summation of the individual equipment power consumptions obtained from the CFD simulation results and the power consumption model

 ISBN: 978-1-61208-460-2

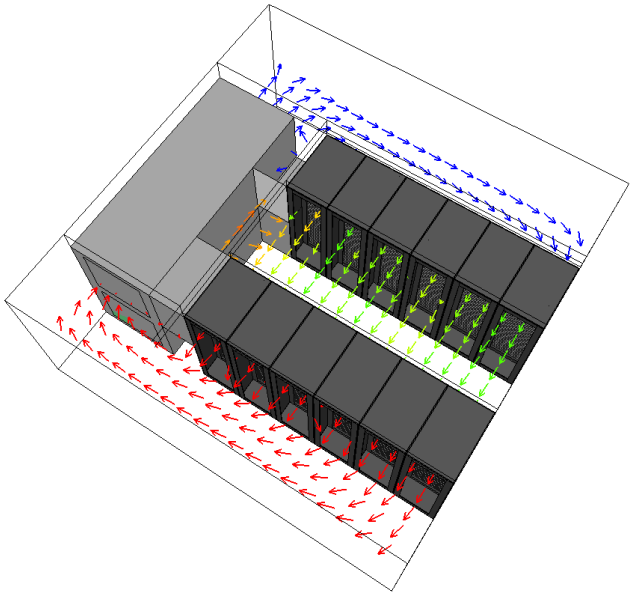
of each equipment.

By combining the proposed simulator and a sensitivity analysis of CFD, VM allocation optimization from the aspect of power consumption was demonstrated to equalize the exhaust temperature in the rack plane. Through VM allocation, we increased the blowing temperature of the air conditioner and reduced its power consumption. In addition, we evaluated the optimum temperature set point of the air conditioner to minimize the total power consumption of the data center with several power usage effectiveness (PUE) [12] values.

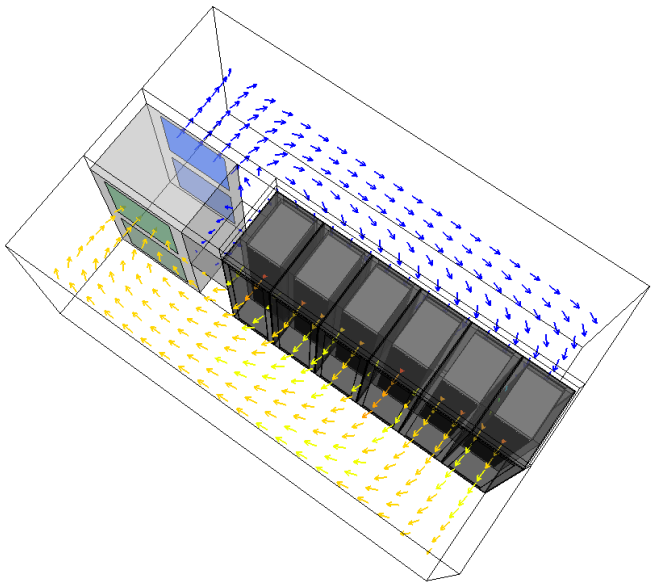
The remainder of this paper is organized as follows. Section II describes the power consumption simulator. Section III describes the procedure for VM allocation. Section IV discusses the evaluation of the proposed power consumption simulator. Finally, Section V presents our conclusions and possible areas of future work.

II. Energy Simulator

Our research group manages an experimental data center as a test bed to demonstrate energy-saving technologies for a green data center. Figure 1(a) shows a general view of the testbed data center. In this data center, a system that reuses exhaust heat from the servers is implemented to reduce power consumption efficiency [16]. Figure 1(b) shows the equipment arrangement and airflow of a conventional data center with one hot aisle (HA) and one cold aisle (CA). On the other hand, the testbed data center has a conventional arrangement with cold and hot aisles and an additional super-hot aisle (SHA) to raise the exhaust heat temperature to around 50°C. The heat reuse efficiency reaches practical levels at this temperature.



(a) Tandem arrangement with cold aisle, hot aisle, and super-hot aisle to reuse exhaust heat from servers



(b) Conventional arrangement with cold aisle and hot aisle

Figure 1. Equipment arrangement in data center

A. Summary of power simulator

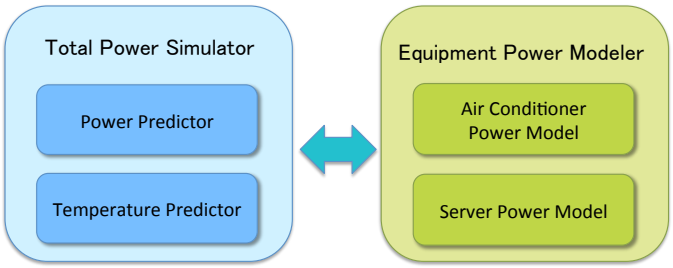
Figure 2(a) shows a conceptual diagram of the power simulator for the data center. In our previous study, the temperature prediction and power estimation were demonstrated using only the machine-learning-based power simulator [17] [18]. In that system, sensor data are required for temperature prediction as well as estimation of the power consumption of the data center.

In this study, the power consumption simulator consists of the CFD simulator and external programs to estimate the power consumption of individual pieces of equipment in the data center. Figure 2(b) shows a conceptual diagram of the power simulator, which comprises CFD and a power predictor, for the data center. It is not necessary to install sensors across the entire data center because the sensor data are not required for estimating the power consumption. As a result, this system is more practical than a power simulator based on machine learning alone. The server's power consumption is estimated by a neural network whose inputs are the CPU usage, intake temperature, and rotational speed of the fan of the air conditioner. The exhaust temperature of each server is estimated by the CFD simulator according to the predicted heating value. The power consumption of each server and that of the air conditioner are estimated using the calculation result of the CFD simulation. By using the CFD simulation result, the power consumption is estimated by the power model for the server and CFD simulation. The CFD simulation and power consumption estimation are performed in turn. The total power consumption of the data center is estimated by the power consumption of the servers and air conditioners. The power consumption of the air conditioner is also estimated by using the learning data on the total power consumption of the servers, temperature from the air conditioner, air conditioner fan rotational speed, and fresh air temperature and humidity. After estimating the power consumption of all servers, the power consumption of the air conditioner is estimated using the air conditioner model. As a result, the power consumption of the entire data center and the temperature distribution in the data center were determined.

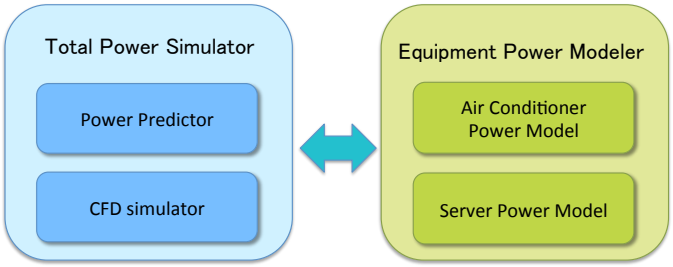
B. CFD simulation

CFD is an analytical technique for numerically solving a hydrodynamic governing equation [19]. For the CFD simulation, we follow the steps described below. First, the simulation

 ISBN: 978-1-61208-460-2



(a) Machine-learning-based power simulator



(b) CFD-based power simulator (this study)

Figure 2. Block diagrams of the power consumption simulator for a data center constructed using only machine learning and using the hybrid CFD and machine learning method

TABLE I. SERVER INFORMAITON FOR POWER MODEL

CPU	Intel(R) Xeon(TM) CPU 3.80 GHz
Memory	1 GB
Storage	500 GB

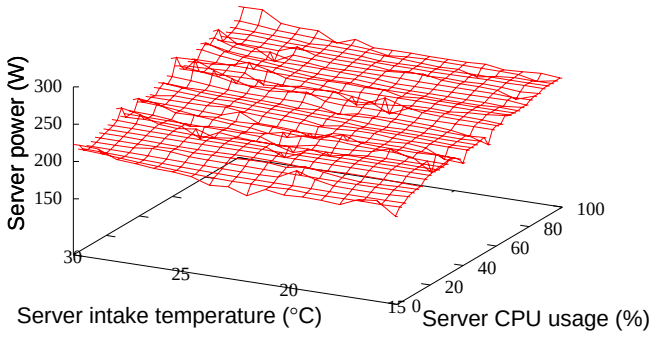


Figure 3. A typical server power model constructed by the neural network method

model is built, in which servers acting as heat sources and an air conditioner are emulated. Next, parameters including the heat value of the server, intake air temperature, and air volume are set. Then, the required estimation is done using the CFD simulator (Flow Designer Ver. 12; Advanced Knowledge Lab., Inc.). As a result of the simulation, the temperature at any point in the entire analysis domain can be evaluated. Two models are used for CFD analysis: steady-state analysis for the state in which the temperature is stable and unsteady analysis for the transition state before converging to a steady state. In this study, a simulation based on steady-state analysis is demonstrated.

C. Server power consumption model

This section describes the construction of the server power consumption model. The CPUs, Memory, hard disc drive (HDD) and internal fans account for most of the power consumption of a server. The power consumption of an HDD is almost constant, and additional power consumption occurs when the HDD is accessed. Memory also consumes a certain amount of energy only when it is accessed. As a result of our experiment to analyze the server power consumption, we found that the power consumption was around 4 W at most when 1 GB of memory is accessed. The change in the power consumption of the HDD is almost negligible even when the drive is accessed. Therefore, we assumed that the change in the server power consumption mainly depends on the CPU and internal fan in this case, because the amount of memory installed in each server was 1 GB at most.

The power consumption of the CPU changes with the usage ratio and its temperature. The CPU temperature depends on the CPU usage ratio, intake temperature of the server, and air speed passing through the server. The air speed passing through the server mainly depends on the rotational speed of the internal fans. The rotational speed of the internal fan is controlled by using the CPU temperature and intake temperature of the server. In this way, the power consumption of the server is determined by several parameters that depend on each other. As a result, a simple linear regression model is not suitable for constructing the power model in the data center.

In this study, we use a neural network to construct the server power consumption model. The parameters of the neural network were determined by using the operation data of a server in the testbed data center. Table I shows the specifications of the server 's measured operation data. The CPU usage ratio and intake temperature were adopted as the explanatory variables to estimate the server power consumption. We also adopt the blowing temperature and fan rotational speed of the air conditioner as explanatory variables. The network shape and hyperparameters of learning are optimized. As a result, the server power consumption model is constructed. The root mean square error (RMSE) between the measured and the estimated values of the test data set is 8.41 kW. Figure 3 shows a server power model constructed by the neural network method. This figure shows the typical relationships among the measured power consumption of the server, its CPU usage ratio, and its intake temperature. The estimated error of the server power consumption reaches 6.7% for the test data.

D. Air-conditioner power model

Coefficient of performance (COP) is an index that indicates the power consumption efficiency of an air conditioner [20], and it is defined as follows:

$$COP = \frac{c}{P_{consume}} \quad (1)$$

where c and $P_{consume}$ are the coolability and total power consumption of the data center, respectively. Figure 4 shows the COP against the operational parameters of outlet air temperature and fan speed of the air conditioner as estimated by a support vector regression model. This support vector

 ISBN: 978-1-61208-460-2

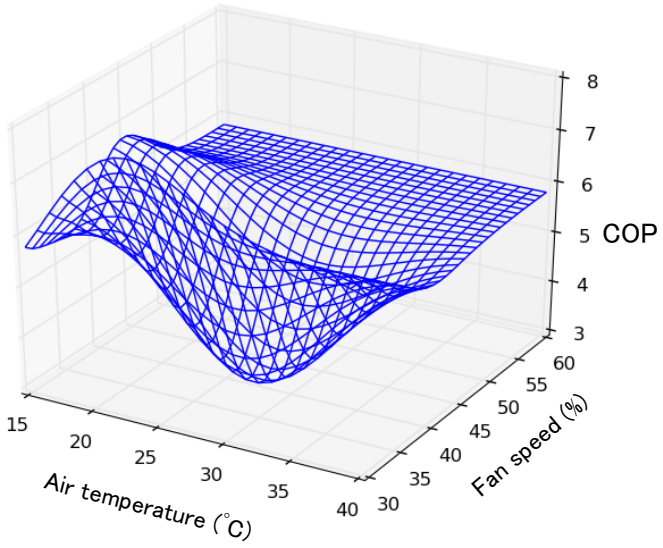


Figure 4. Air-conditioner power model constructed by support vector regression method

regression model is trained by using the recorded data from our experimental data center [21] [22]. From this figure, it is clear that the COP dynamically changes against the operational parameters of the air conditioner. Therefore, this suggests that controlling the air conditioner using the prediction method enables the power consumption to be reduced.

III. Task Allocation

A. Procedure for Task Allocation

This section describes the procedure for determining the quantity of tasks to assign to each server when some workload is assigned to the entire data center. A VM is assigned and moved to each server as a task unit. We do not distinguish between different types of VMs, and we assume that there is no difference in the load between VMs. Turning off servers with no assigned task to reduce power consumption is not done in this allocation method.

VM allocation follows the steps described below.

1) Calculate the temperature distribution subject to a given initial allocation of workloads by order analysis of CFD simulation.
2) Conduct sensitivity analysis against the given objective with respect to the generated heat at each server. In this study, the objective is to achieve a uniform exhaust temperature distribution, and thus, the objective function is defined using the variance of the exhaust temperatures.
3) Find the "local optimal" heat pattern to minimize the variance using the sensitivity analysis result.
4) Find the VM allocation pattern that produces a generated heat pattern that is closest to the local optimum.
5) Reallocate VMs to realize the allocation pattern.

The VM allocation procedure for the data center model with a conventional arrangement is described in the following subsections.

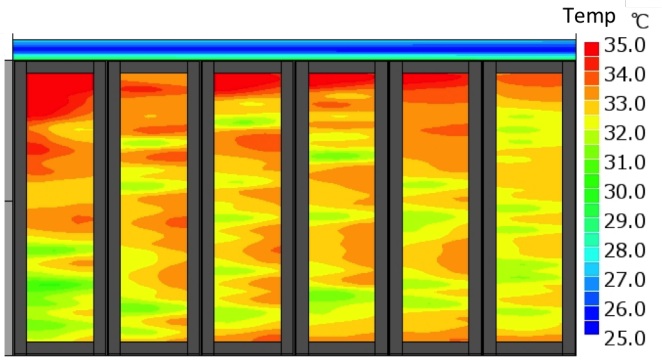


Figure 5. Temperature distribution in the exhaust-side plane of the rack when VMs are assigned to servers randomly

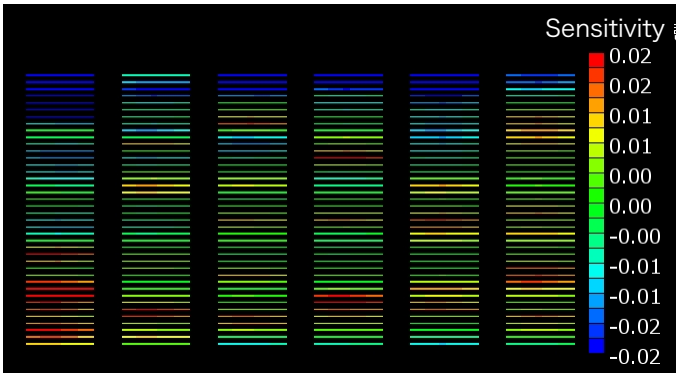


Figure 6. Sensitivity map of each server for temperature distribution in the case of the randomly assigned VM shown in Fig. 5

B. Order analysis of the initial pattern of generated heat

As preparation for sensitivity analysis, order analysis for the initial workload is demonstrated. The initial pattern of generated heat is obtained for the VM allocation randomly assigned to the servers. Figure 5 shows the temperature distribution in the exhaust-side plane of the rack. CFD simulation is applied for the initial pattern of generated heat.

C. Sensitivity analysis for exhaust temperature in rack plane

A sensitivity analysis is a non-parametric process to estimate the intensity of the influence (sensitivity) of a small change in each parameter [23]. In this process, the variance with the temperature of each server in the exhaust-side plane is used as an objective function. The sensitivity analysis for the objective function is demonstrated, and then, the sensitivities for the variance in the rack exhaust temperature of each server are obtained. Figure 6 shows the value of the sensitivity of each server in a color bar. This sensitivity shows exhaust temperature variance in the rack plane when the heating value increases to 1 W.

D. Calculate target pattern of generated heat

By using the sensitivity, we find the servers' allocation pattern of generated heat that equalizes the exhaust temperature in the rack plane. The local optimal generated heat pattern that does not require VM reallocation is the target pattern. The target amount of generated heat $target\_Power_i$ of server i

 ISBN: 978-1-61208-460-2

TABLE II. EXHAUST TEMPERATURE VARIANCE WHEN USING FIVE COEFFICIENTS FOR PATTERN OF GENERATED HEAT

Coefficient	*Variance*
0 (initial)	0.175
15000	0.479
30000	0.256
45000	0.107
60000	0.103

is calculated, and it changes in proportion to the sensitivity of the current amount of generated heat $Power_i$, as shown in (2). After finding the local optimum, we do not recalculate to find a better allocation pattern because the exhaust temperature distribution hardly improves from the viewpoint of the power consumption.

$$target\_Power_i = Power_i + Sensitivity_i \times \alpha \quad (2)$$

Here, coefficient α in (2) is a proportionality constant. The coefficient α is set to an appropriate value, and a pattern of generated heat that reduces the rack exhaust temperature variance is obtained.

The difference between the intake and the exhaust air temperature is $\Delta Temp_i$, and it depends on the heating value of a server $Power_i$, speed of air passing over a server $AirSpeed_i$, and the heat transfer coefficient between the air and the server $\eta_{heat\_transfar}$, as shown in (3) [24].

$$\Delta Temp_i = \frac{Power_i}{Air\_Speed_i} \times \eta_{heat\_transfar} \quad (3)$$

Assuming that the speed of air passing over a server is constant, $\Delta Temp_i$ is proportional to the server heating value. The exhaust temperature variance in the rack plane is almost proportional to the square of the server heating value. As with the temperature, the objective function of sensitivity analysis is proportional to the square of the coefficient α. The order analysis for the patterns of generated heat calculated using two values of α is shown, and the exhaust temperature variance in the rack plane is calculated. Table II shows the exhaust temperature variance in the rack plane as estimated by order analyses using at least three patterns of generated heat, including the initial pattern. A quadratic function of the exhaust temperature variances in the rack plane and a coefficient using at least three pairs of variances are calculated. As a result, a coefficient is estimated so that the quadratic function approaches a local minimum. Figure 7 shows the quadratic curve for the five points shown in Table II and the local minimum of the quadratic curve. Here, an appropriate value for α is 57676 in the initial pattern of generated heat. Then, the target heating value of each server using this value of α and equation (2) is obtained.

E. VM reallocation

VM reallocation is demonstrated to make each server's amount of generated heat close to the target pattern. The difference between them is shown in equation (4).

$$diff_i = Power_i - target\_Power_i \quad (4)$$

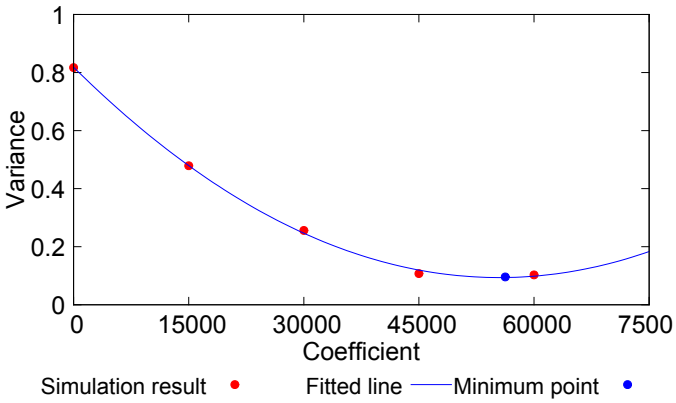


Figure 7. Quadratic function of rack exhaust temperature variances and coefficient

$$\sum |diff_i| \quad (5)$$

Here, $server_j$ means that the decrease in $diff_i$ becomes a maximum when one VM is removed from the server. $server_k$ means that the decrease in $diff_i$ becomes a maximum when one VM from the server is moved using the server power model. Then, a VM is moved from $server_j$ to $server_k$. This procedure is repeated until (5) does not decrease. As a result, this procedure greedily assigns server VMs.

IV. EVALUATION

A. Power estimation using proposed simulator

We estimated the power consumption of the data center by using the developed power simulation with the measured CPU usage ratio of each server and the air-conditioner setting in the testbed data center. We compared the result of the power simulation with the measured power consumption of the testbed data center. Our testbed data center includes many types of servers, including the servers for which a power model was constructed. Table III shows information about these servers. The power model of each server was constructed in the same manner as in the previous section. Table IV shows the power consumption of each piece of equipment, total power consumption estimated for the data center, and power consumption measured in the testbed data center. The power simulator estimated the power consumption of the data center with an average error rate of 5.34%.

B. VM allocation in data center

We describe the VM allocation procedure for the data center model with a conventional arrangement (Fig. 1(b)). This data center model comprises six racks with 40 servers per rack and air conditioners. RX300S4 servers in tableI are constructed in this data center model. The maximum number of VMs $maxVM$ that each server runs is eight. The CPU usage cpu_usage_i of server$i$ was calculated by using the assigned number of VMs VM_i and (6).

$$cpuusage_i = \frac{inVM_i}{maxVM} \times 100 \quad (6)$$

 ISBN: 978-1-61208-460-2

TABLE III. SERVER INFORMATION FOR THE TESTBED DATA CENTER

Server	CPU	Memory	HDD storage
Server A	Intel(R) Xeon(R) CPU 2.40 GHz	1 GB	200 GB
Server B	Intel(R) Xeon(TM) CPU 3.20 GHz	1 GB	200 GB
Server C	Intel(R) Xeon(TM) CPU 3.80 GHz	1 GB	500 GB
Server D	Intel(R) Xeon(R) CPU 1.60 GHz	1 GB	250 GB
Server E	Intel(R) Xeon(R) CPU 2.00 GHz	1 GB	400 GB
Server F	Intel(R) Xeon(TM) CPU 3.80 GHz	1 GB	400 GB
Server G	Intel(R) Xeon(R) CPU 3.00 GHz	1 GB	1 TB
Server H	Intel(R) Xeon(TM) CPU 3.80 GHz	1 GB	500 GB

TABLE IV. POWER CONSUMPTION OF EACH DEVICE ESTIMATED BY SIMULATION AND MEASUREMENT IN TESTBED DATA CENTER

	Server power	*Air-conditioner power*	*Data center total power*
Testbed data center	57.15 kW	5.85 kW	65.70 kW
Simulation result	56.63 kW	5.74 kW	62.37 kW

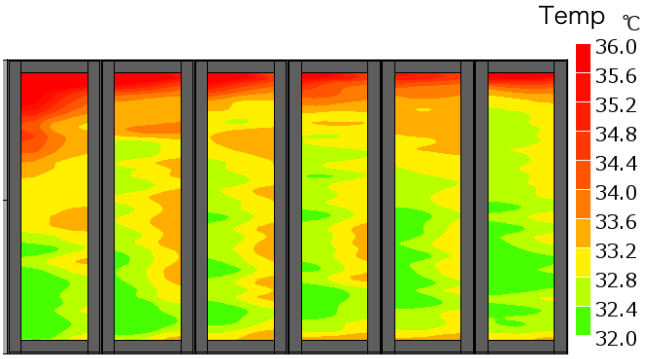


Figure 8. Air temperature distribution of the data center with random pattern of generated heat (RMSE: 0.82)

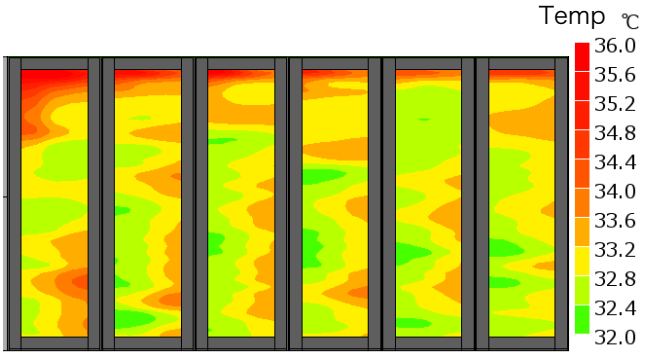


Figure 9. Air temperature distribution of the data center after VM reallocation (RMSE: 0.23)

The average utilization rate of the data center is 20%–30%; therefore, 650 VMs are assigned to 240 servers for the initial pattern of generated heat so that the mean utilization rate of the servers is 33%. Figure 8 shows the temperature distribution of the rack exhaust side as obtained from a CFD simulation using this pattern of generated heat. The rack exhaust temperature variance for this temperature distribution was 0.76. The proposed VM allocation procedure was implemented for this initial pattern of generated heat. 140 VMs were migrated, and a new pattern was obtained. Then, a CFD simulation was conducted using the newly obtained pattern of generated heat. The new temperature distribution of the rack exhaust side was obtained, and then the rack exhaust temperature variance decreased to 0.23. Figure 9 shows this temperature distribution. Table V shows the temperature distribution of the rack exhaust side from the analysis results before and after reallocation. The maximum rack exhaust temperature decreased from 38.4°C to 36.2°C by using the proposed VM allocation procedure. This indicates that the air-conditioner temperature can increase by 2.2°C relatively in comparison with that before the VM reallocation, and then the power consumption of the air conditioner might decrease by around 5%.

TABLE V. RACK EXHAUST TEMPERATURE VARIANCE AND MAXIMUM RACK EXHAUST TEMPERATURE BEFORE AND AFTER VM REALLOCATION

	Initial (random)	*After reallocation*
Exhaust temperature variance	0.82	0.23
Maximum exhaust heat temperature	38.4°C	36.2°C

C. Minimizing total power consumption of data center

1) Suitable operation point of air conditioner: The power consumption of a server depends on its CPU usage, its intake temperature, and other factors, and it increases when the intake temperature increases. The power consumption of the air conditioner depends on its blowing temperature and fan rotational speed, and it increases when the blowing temperature is reduced. Therefore, increasing the air conditioner's blowing temperature reduces its power consumption. On the other hand, as the intake temperature of the servers increases, the total power consumption of the servers increases because the leakage current at the processors or rotational speed of the internal fans increase. Therefore, it is expected that operational temperature has the most suitable operation point from the aspect of power consumption of the data center which is the sum of the power consumption of all servers and air conditioners. A reduction in the power consumption of the entire data center is expected by operating the air conditioner at the abovementioned blowing temperature. Figure 10 shows the total power consumption of the servers, power consumption of the air conditioner, and total power consumption of the data center as described in when the air temperature is varied from 15°C to 30°C and the air conditioner is operated. Figure 11

 ISBN: 978-1-61208-460-2

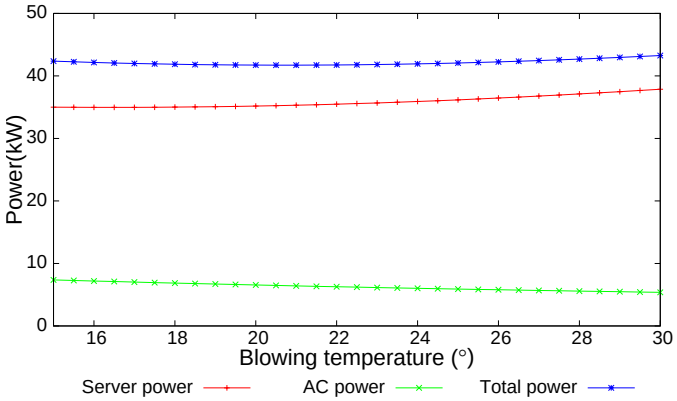


Figure 10. Power consumption of servers, power consumption of air conditioner, and total power consumption of data center versus air-conditioner blowing temperature

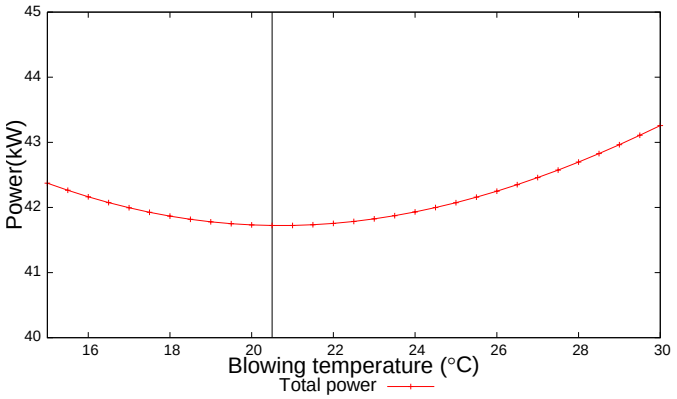


Figure 11. Total power consumption of data center versus air temperature, showing the temperature that minimizes the power consumption

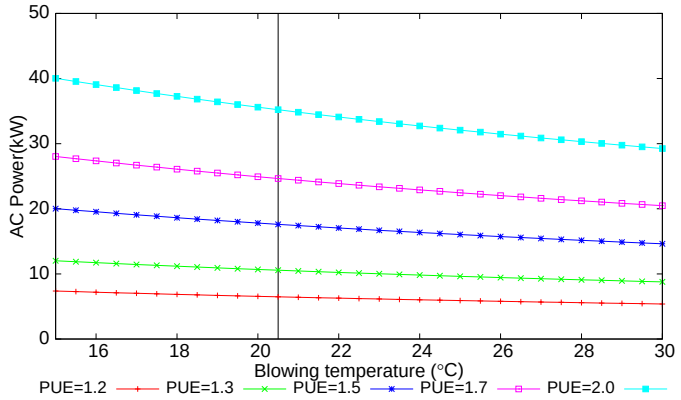


Figure 12. Power consumption of the air conditioner for PUE values of 1.2, 1.3, 1.5, 1.7, and 2.0

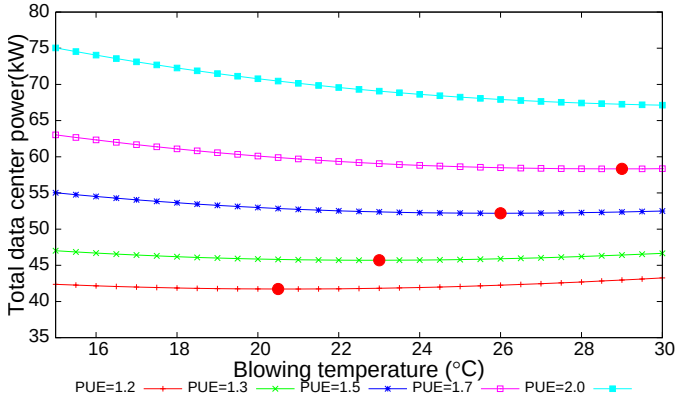


Figure 13. Total power of the data center and the air temperature that minimizes the power consumption of the data center for PUE values of 1.2, 1.3, 1.5, 1.7, and 2.0

shows an enlarged view of the total power consumption of the data center shown in Figure 10. The total power consumption of the data center is given by a concave function, and it reaches a minimum when the air blowing temperature of the air conditioner is set to 20.5°C and operated. In this manner, the air-conditioner blowing temperature that minimizes the power consumption of the data center was calculated by the power consumption simulator. A reduction in the power consumption of the data center is expected by using the abovementioned value of the air blowing temperature.

2) Change in operation point of air conditioner for PUE: We evaluate the change in the air conditioner's blowing temperature that minimizes the total power consumption of the data center for different PUE values. The PUE is an index that indicates the power consumption efficiency of the data center, and it is calculated by using the power consumption of all servers $Power_{server}$ and the total power consumption of the data center $Power_{DC}$, as given in (7).

$$PUE = \frac{Power_{DC}}{Power_{server}} \tag{7}$$

According to the power simulation result, when the air temperature was set to 20.5°C as calculated in IV-C1, the PUE was 1.18. We estimate and compare the power consumption of the data center for PUE values of 1.3, 1.5, 1.7, or 2.0 when the air conditioner blowing temperature was set to 20.5°C. Figure 12 shows the power consumption of the air conditioner for each PUE value when the air conditioner blowing temperature was varied.

The air-conditioning power that gives each PUE was calculated on the assumption that the power to give the PUE (1.18 at 20.5°) obtained in an actual data center varies in proportion to each temperature. It is assumed that the server power did not change in this case. In addition, Figure 13 shows the change in the total power consumption of the data center at each PUE versus the air-conditioner blowing temperature using the air-conditioner power shown in Fig. 12. The red points show the operating temperature point at which the data center's total power is minimized for each PUE. In this manner, an air conditioner's operation point that minimizes the total power consumption of the data center changes with the PUE. Moreover, it was found that the air temperature that minimizes the power decreases so that the PUE value is small. In other words, it is advantageous from the viewpoint of the power consumption to set a lower air temperature when the power efficiency is high. Thus, the developed power simulator is a powerful tool for VM allocation to reduce an air conditioner's

 ISBN: 978-1-61208-460-2

power consumption. Furthermore, it can be used to determine the most suitable operation point from the viewpoint of the power consumption of the data center through the dynamic simulation of the power consumption.

V. Conclusion and Future Work

A novel power simulator was developed for the dynamic and real-time estimation of the power consumption of a data center. This simulator estimates the data center's power consumption through a combination of CFD and machine learning. The prediction error was suppressed to around 5.3% for the power consumption. A VM assignment policy was proposed to reduce the power consumption of the air conditioners by narrowing the distribution of exhaust heat from the servers. In addition, the optimum operation temperature was proposed to reduce the power consumption by using the power simulator.

These results indicate that the power simulator developed in this study shows promise as a real-time and dynamic power simulation tool.

In this study, learning data were obtained from a real server located in a data center. However, several interactions occur between servers in a real environment. In future work, it is necessary for the power simulator to take the relationships among the servers.

Acknowledgment

This work was supported by a CO_2 emission reduction project of the Japanese Ministry of the Environment and NICT. A part of this work was collaborative research with NTT Network Innovation Laboratories. We thank Advanced Knowledge Laboratory, Inc., for support with the CFD techniques. We thank Takasago Thermal Engineering Inc. for support with the air conditioner and aisle arrangement. We also thank Fujitsu Ltd. for supporting server management and sharing information about the characteristics of the servers.

References

[1] J. Koomey, "Growth in data center electricity use 2005 to 2010," A report by Analytical Press, completed at the request of The New York Times, Aug. 2011.

[2] A. Banerjee, T. Mukherjee, G. Varsamopoulos, and S. Gupta, "Cooling-aware and thermal-aware workload placement for green hpc data centers," in Proceedings of International Green Computing Conference, Aug. 2010, pp. 245–256.

[3] T. D. Boucher, D. M. Auslander, C. E. Bash, C. C. Federspiel, and C. D. Patel, "Viability of dynamic cooling control in a data center environment," Journal of electronic packaging, vol. 128, no. 2, Jun. 2006, pp. 137–144.

[4] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," Future generation computer systems, vol. 28, no. 5, May 2012, pp. 755–768.

[5] B. Battles, C. Belleville, S. Grabau, and J. Maurier, "Reducing data center power consumption through efficient storage," NetApp Technical Report, Feb. 2007.

[6] L. Parolini, B. Sinopoli, and B. H. Krogh, "Reducing data center energy consumption via coordinated cooling and load management," in Proceedings of the Conference on Power Aware Computing and Systems, Dec. 2008.

[7] M. Norota, H. Hayama, M. Enai, T. Mori, and M. Kishita, "Research on efficiency of air conditioning system for data-center," in Proceedings of International Telecommunications Energy Conference 2003, Oct. 2003, pp. 147–151.

[8] C. Ge, Z. Sun, and N. Wang, "A survey of power-saving techniques on data centers and content delivery networks," IEEE Communications Surveys Tutorials, vol. 15, no. 3, Jul. 2013, pp. 1334–1354.

[9] D. Cavdar and F. Alagoz, "A survey of research on greening data centers," in Proceedings of Global Communications Conference Exhibition and Industry Forum 2012, Dec. 2012, pp. 3237–3242.

[10] S. Khuller, J. Li, and B. Saha, "Energy efficient scheduling via partial shutdown," in Proceedings of Society for Industrial and Applied Mathematics 2010, Jan. 2010, pp. 1360–1372.

[11] M. Iyengar, R. Schmidt, and J. Caricari, "Reducing energy usage in data centers through control of room air conditioning units," in Proceedings of 12th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, Jun. 2010, pp. 1–11.

[12] The Green Grid, "Guidelines for energy-efficient data centers," *available at http://www.thegreengrid.org/*.

[13] C. D. Patel, C. E. Bash, C. Belady, L. Stahl, and D. Sullivan, "Computational fluid dynamics modeling of high compute density data centers to assure system inlet air specifications," in Proceedings of The Pacific Rim/ASME International Electronic Packaging Technical Conference and Exhibition, Jul. 2001, pp. 1–9.

[14] C. Patel, R. Sharma, C. Bash, and A. Beitelmal, "Thermal considerations in cooling large scale high compute density data centers," in Proceedings of Proceedings of IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, June 2002, pp. 767–776.

[15] U. Sing, A. Singh, and A. Sivasubramaiam, "CFD-based operational thermal efficiency improvement of a production data center," in Proceedings of the First USENIX Conference on Sustainable Information Technology, ser. SustainIT'10, Feb. 2010, pp. 6–6.

[16] Y. Taniguchi and et al., "Tandem equipment arranged architecture with exhaust heat reuse system for software-defined data center infrastructure," IEEE Transactions on Cloud Computing, vol. PP, no. 99, June 2015, pp. 1–13.

[17] Y. Tarutani and et al., "Temperature distribution prediction in data centers for decreasing power consumption by machine learning," in Proceedings of IEEE 7th International Conference on Cloud Computing Technology and Science, Dec. 2015.

[18] S. Tashiro and et al., "A network model for prediction of temperature distribution in data centers," in Proceedings of IEEE 4th International Conference on Cloud Networking (CloudNet), Oct. 2015, pp. 261–266.

[19] J. D. Anderson and J. Wendt, Computational fluid dynamics. Springer, 1995, vol. 206.

[20] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling" cool": Temperature-aware workload placement in data centers." in USENIX annual technical conference, General Track, 2005, pp. 61–75.

[21] T. Deguchi and et al, "Impact of workload assignment on power consumption in software-defined data center infrastructure," in Proceedings of IEEE CloudNet 2014, Oct 2014, pp. 440–445.

[22] T. Deguchi and et al., "A workload assignment policy for reducing power consumption in software-defined data center infrastructure," IEICE Transactions on Communications, vol. E99B, no. 2, Feb. 2016, pp. 347–355.

[23] K. Momose and K. Ikejima, "Development of sensitivity-based decision support system for heat and fluid flow design : Variational approach to sensitivity analysis," in The society of Heating, Air-Conditioning and Sanitary Engineers of Japan, vol. 18, no. 1, 2006, pp. 403–406.

[24] Y. Cengel, Ed., Heat transfer: a practical approach 2nd edition. Mcgraw-Hill, Oct. 2003, ISBN: 978-0072458930.

 ISBN: 978-1-61208-460-2

Analysis of Virtual Networking Options for Securing Virtual Machines

Ramaswamy Chandramouli
Computer Security Division, Information Technology Laboratory
National Institute of Standards & Technology
100 Bureau Drive, Gaithersburg, MD, USA
E-Mail: mouli@nist.gov

***Abstract* – Virtual Machines (VMs) constitute the primary category of resources to be protected in virtualized infrastructures. Out of the two types of protection for VMs – Host-level and Network-level – it is the approaches for the Network-level protection that are different in virtualized infrastructures as compared to those for non-virtualized environments. This is due to the fact that the VMs are end nodes of a virtual network as opposed to being end nodes of a physical network. In this paper, we provide a detailed analysis (in terms of advantages and disadvantages) of some of the key approaches for two Network-level protection measures for virtualized infrastructures – Network Segmentation and Traffic Control using Firewalls. The choice of these two Network-level protection measures is due to the fact that they form the foundation for the network configuration of the entire virtualized infrastructure. We also provide the overall conclusions from the analysis in the form of recommended deployment choices based on approaches for these two network-level protection measures for securing VMs.**

Keywords - Virtual Machine; VLAN; Hypervisor; VXLAN; Virtual Firewall.

I. INTRODUCTION

Virtualized hosts (also called hypervisor hosts) are increasingly deployed in data centers because of efficiency, scalability and cost considerations. The virtualized infrastructure resulting from the deployment of virtualized hosts has three main categories of components - Hypervisor Software, Virtual Machines (VMs) and Virtual Networking components such as Virtual Network Interface Cards (vNICs), Virtual Switches and Virtual Firewalls.

Out of the three categories of components above, the VMs constitute the fundamental resource to be protected in a virtualized infrastructure, since they are the compute engines on which business/mission critical applications of the enterprise run. These VMs are virtual counterparts of physical servers and hence just like their physical counterparts, security for these VMs has to be provided through host-level and network-level measures. These measures may also vary depending upon whether the overall virtualized infrastructure (in which the VM resides) is used for in-house enterprise applications or for offering cloud services to external entities (e.g., Infrastructure as a Service Public Cloud). We provide a brief overview of the two types of protection mentioned above. (a) Host-level protections required for VMs include features for robust authentication, access using secure access protocols and secure session establishment. The mechanisms required for providing these features are no different for VMs compared to their physical counterparts (i.e., physical servers). (b) Network-level protections required for VMs are feature-wise similar to those that are required by their physical counterparts. However, the mechanisms or approaches required for providing these protections are different due to the fact that the VMs are end nodes of a virtual network as opposed to being end nodes of a physical network.

For any type of datacenter infrastructure (virtualized or non-virtualized), there is a general consensus that the following are some of the key Network-level protection measures [1]. They are: (a) Network segmentation or isolation, (b) Traffic control using firewalls, (c) Creating Redundant communication pathways, and (d) Traffic Monitoring and Prevention using IDS/IPS.

Out of the above four network-level protection measures, the first two - Network Segmentation and Traffic Control using Firewalls - form the foundation for the network configuration of the entire virtualized infrastructure. Hence, in this paper, we have chosen to focus on different approaches or mechanisms used for these two network-level protection measures, by performing a detailed analysis of the advantages and disadvantages of each of the approaches.

 ISBN: 978-1-61208-460-2

Before we describe the organization of the rest of the paper, a few observations regarding our chosen network-level protection measures in the context of virtualized infrastructure are in order. In a virtualized infrastructure, the distinguishing networking environment is the virtual network. Hence the network segmentation approaches discussed in this paper have to involve some virtual network components such as virtual switches. Similarly, a viable approach for traffic control using firewalls has to use a virtual firewall instead of a physical firewall. In Section II, we focus on two network segmentation approaches and discuss the advantages and disadvantages of each. Control of virtual network traffic using two different types of virtual firewalls and the advantages and disadvantages of each are analyzed in Section III. In Section IV, we provide the overall conclusions from the analysis, in the form of deployment choices based on approaches for the two network-level protection measures for securing VMs.

II. NETWORK SEGMENTATION IN VIRTUAL NETWORKS

The approaches to network segmentation in the context of virtualized infrastructures are the same as those used in physical network with some variations (these variations are underlined in our description below): (a) Using a combination of firewalls – the firewalls used are virtual firewalls (as opposed to a physical firewall) and are implemented as Virtual Security Appliance (VSA) and hosted on security hardened VMs with multiple Virtual Network Interface cards (vNICs). Each vNIC may be connected to a different network segment [2]. (b) Isolated network segments created as logical segments on top of a physical network segment – one is the VLAN approach that is based on tagging packets and switch ports with a unique identifier called VLAN ID, and the other is overlay-based virtual networking technology that creates an overlay network by encapsulating packets with IDs depending upon the type of overlay technology. Both approaches (VLAN and Overlay) rely on the capabilities in virtual switches of the virtualized host.

The three approaches for network segmentation in virtualized infrastructures outlined above are discussed in Sections A, B and C below. After a brief description of each approach, an analysis of each approach is provided with the relative advantages and disadvantages. Where ever applicable, the distinct advantage of a particular approach is also brought out.

A. Network Segmentation Using a Combination of Firewalls

Let us now consider a virtualized host with 4 VMs – VM1, VM2, VM3 & VM4. We can form a network segment (say a DMZ) using two virtual firewalls, one each on any two VMs - say VM1 & VM4. These firewalls are VM-based Virtual Security Appliances residing on VMs defined with multiple vNICs – each one connected to a different network segment. The firewall in VM1 then will have one vNIC connected to an external network (say the public Internet) of the enterprise and the other vNIC connected to the DMZ segment in the virtualized network within a virtualized host. Correspondingly the firewall in VM4 has to have one vNIC connected to the internal network of the enterprise and the other vNIC connected to the DMZ. The vNIC connection to the DMZ (from both firewall VMs - VM1 & VM4) is established by their pathway to an internal-only virtual switch. This internal-only virtual switch has no uplink connection to any physical NIC of the virtualized host and hence traffic from any VM connected to it cannot travel directly outside the virtual network segment (not to speak of outside the virtualized host). The internal-only switch can only forward traffic directly to VMs connected to it - say VM2 &, VM3. All incoming and outgoing traffic into the VMs connected to the internal-only virtual switch whose source/target is an internal/external network, has to go through the firewall in VM1 or in VM4. The firewalls in VM1 & VM4 thus form the gatekeepers for the virtual network segment (i.e., DMZ).

A.1 Analysis

The advantages of network segmentation within the virtualized network of a virtualized host using a combination of virtual firewalls are: (a) Simplicity of Configuration: It can be configured with commodity firewall VSAs hosted on multi-vNIC VMs. (b) Flexibility within a Virtualized host: More than one isolated network segment can be created within the virtual network of the virtualized host by simply adding another firewall VM.

The limitations of this approach to network segmentation in a virtualized network are the following: (a) The logical network segment created inside a virtualized host can neither be extended to the physical network of the data center nor to the virtual network in another virtualized host (since segmentation is obtained by virtual firewalls inside the virtualized host). This makes the approach to network segmentation non-scalable. (b) A consequence of creating a non-scalable network segment is that the migration of a VM in the network segment to any other virtualized host (due to

 ISBN: 978-1-61208-460-2

performance or availability or load balancing reasons) is out of the question, unless a network segment (with identical configuration) exists on the target virtualized host.

B. Network Segmentation Using Virtual LAN (VLAN) Technology

The second approach to network segmentation in virtualized infrastructures is broadcast-containment networking technologies, such as VLAN. The requirement for this is that the hypervisor should have capabilities to define virtual switches that are VLAN-aware [3][4]. The segmentation is obtained by assignment of an identifier called the VLAN ID to one or more ports of a switch and connecting the VMs designated for that VLAN segment to those ports. VMs on one VLAN can only communicate directly with VMs on the same VLAN and a router is needed for communication between VMs on different VLANs [5]. Assignment of a VM to a particular VLAN can be based on the application tier it is hosting (e.g., Web Server, DBMS server, etc.) or the client to which the VM belongs (in cloud data centers). These VLAN-capable virtual switches (VS) can perform tagging of all packets going out of a VM with a VLAN tag (depending upon which port it has received the packet from) and can route an incoming packet with a specific VLAN tag and MAC address to the appropriate VM by sending it through a port whose VLAN ID assignment equals the VLAN tag of the packet. An example of a VLAN-based virtual network segmentation inside a virtualized host is given in Figure 1.

B.1 Analysis

The advantages of a VLAN-based network segmentation approach are: (a) Network segments can extend beyond a single virtualized host (unlike the segment defined using virtual firewalls) since the same VLAN ID can be assigned to ports of virtual switches in different virtualized hosts. (b) The number of network segments that can be configured is reasonably large since a single virtual switch can typically support 64 ports and the 12-bit VLAN ID address space enables creation of 4000 VLAN segments.

The disadvantages of VLAN-based network segmentation approach are: (a) The configuration of the ports in the physical switch attached to a virtualized host must exactly match the VLANs defined on the virtual switches inside that virtualized hosts. This results in tight coupling between virtual network and physical network of the data center. The consequence of this tight coupling is that the port configuration of the physical switches has to be frequently updated since the VLAN configuration of the virtual network of the attached virtualized host may frequently change due to migration of VMs among VLANs as well as among virtualized hosts. (b) Another consequence of frequent migration of VMs among VLANs, as well as among virtualized hosts is that the VLAN configuration of ports in the physical switch may not match with that of the connected virtualized host. This may result in a situation where a particular hypervisor (or a virtualized host) may end up processing messages for every VLAN on the network, even when it is not hosting any active VM belonging to that VLAN [6] and (c) Segments created using broadcast-containment technologies cannot be allowed to have a large span since they will result in greater traffic in the overall data center due to multicast and broadcast traffic. But greater VM mobility (due to load balancing and availability reasons) may require VLANs with a large span resulting in an undesirable phenomena called VLAN sprawl [6].

C. Network Segmentation Using Overlay-based Virtual Networking Technology

In the VLAN-based approach, the logical network segments were created on a physical LAN using portgroups of virtual switches inside virtualized hosts. These logical network segments did span multiple virtualized hosts. The total number of these segments possible is limited to around 4000 due to the 12 bit address space of VLAN ID. Another limitation is the lack of independence between the physical and virtual networking infrastructure, since the port configuration of the physical switches attached to the virtualized hosts have to be consistent with the VLANs defined on the port groups of virtual switches inside those virtualized hosts. The overlay-based virtual networking approach to network segmentation overcomes these two limitations in the following two ways [7].

 ISBN: 978-1-61208-460-2

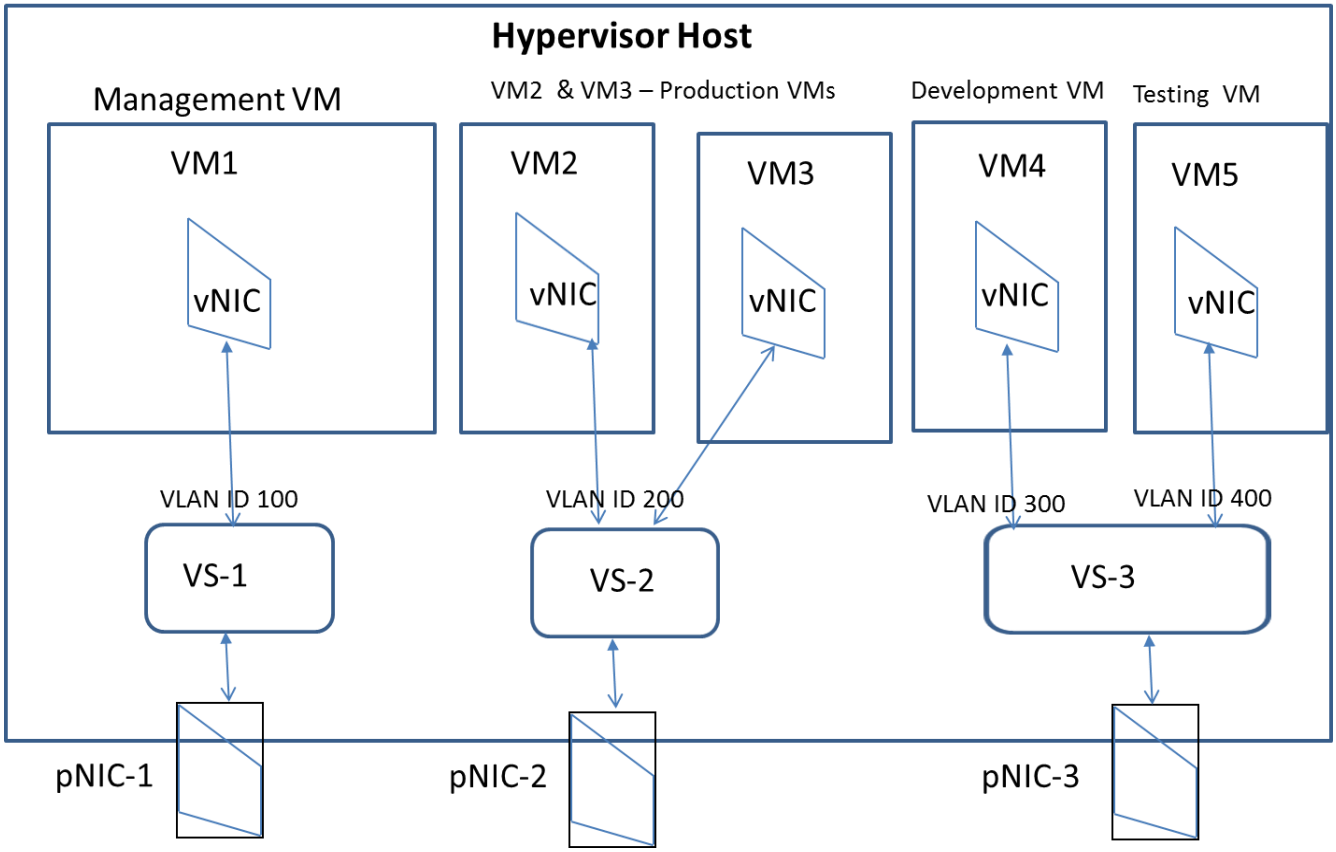


Figure 1 - VLAN-based Network Segmentation

(a) Overlay-based virtual networking technologies have a larger address space enabling larger number of virtual network segments. An example is the 24 bit VXLAN ID of the VXLAN overlay scheme that can enable 16 million virtual network segments to be defined, and (b) The overlay schemes by their definition, create a logical Layer 2 network (called the overlay network) over the physical Layer 3 backbone of the data center (called the underlay network). Since this scheme does not warrant any modifications to the physical network, it provides physical-logical network independence. As already alluded to, overlay-based virtual networking schemes achieve segmentation by creating a logical Layer 2 network over the physical Layer 3 network. The overlay network is created by encapsulating a native Layer 2 packet with another Layer 2 identifier. There are three common encapsulation schemes – VXLAN, GRE and STT [8].

Let us now look at the encapsulation process in VXLAN [9] through components shown in Figure 2. The Ethernet frame originating from a VM, that just holds the MAC address of the destination VM is encapsulated in two stages: (a) First with the 24 bit VXLAN ID (virtual Layer 2 (L2) segment) to which the sending/receiving VM belongs and (b) Second, with the source/destination IP address of the VXLAN tunnel endpoints called VTEPs. [10]. VTEPs are logical network endpoints (nodes) for the encapsulated VXLAN packets and they reside in the kernel of a hypervisor. A VXLAN-encapsulated packet originates at the VTEP in the kernel of the hypervisor where the source VM resides (carrying the VTEP's address as the source IP address) and terminates at the VTEP in the kernel of the hypervisor where the destination VM resides (carrying this VTEP's address as the destination IP address). Thus, we see that VXLAN encapsulation enables creation of a virtual Layer 2 segment that can span not only different hypervisor hosts but also different IP subnets within the data center.

C.1 Analysis

The advantages of a network segmentation approach based on Overlay-based networking technology are: (a)

 ISBN: 978-1-61208-460-2

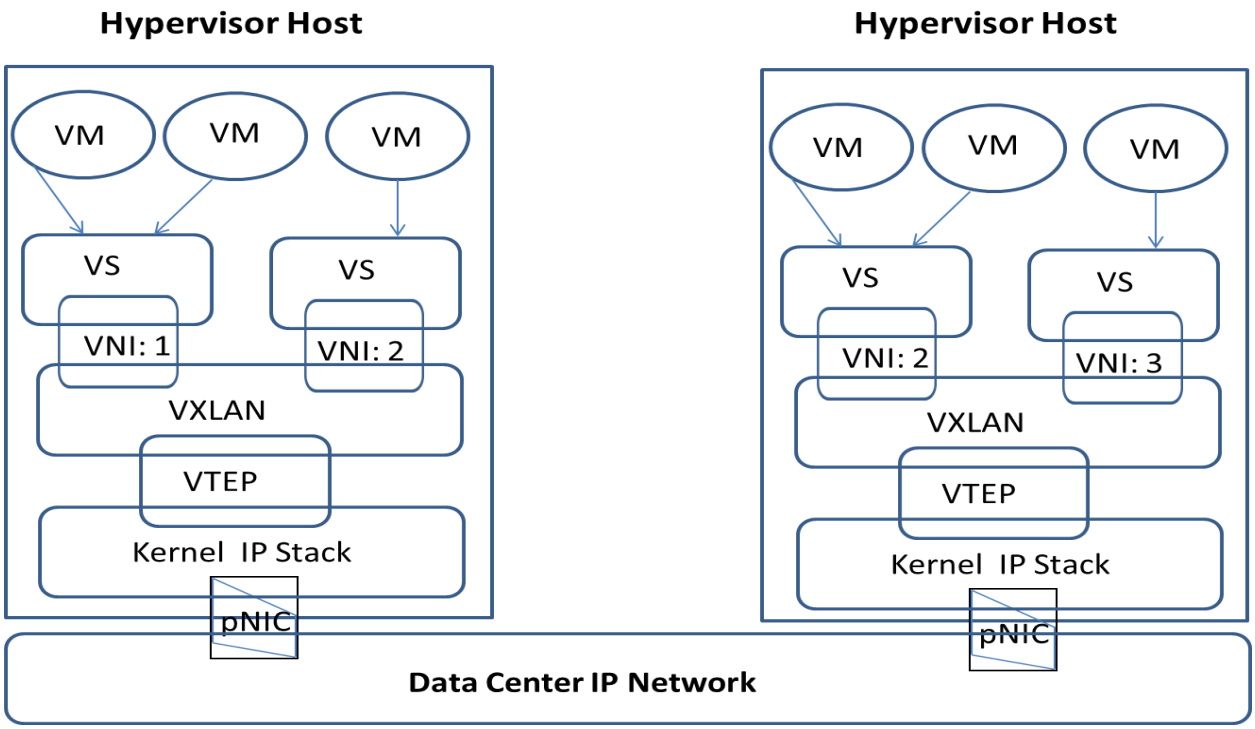


Figure 2 – Overlay-based Virtual Network Segmentation

Because of independence between the virtual network and the physical network, there is greater VM mobility compared to a VLAN- based virtual network environment, (b) The physical-logical network independence, together with the bigger overlay segment ID address space (e.g., a VXLAN ID is 24 bits as opposed to 12 bit VLAN ID allowing for 16 million segments compared to 4096 for VLAN), makes the overlay based network segmentation infinitely scalable. Another factor contributing to scalability of the overlay scheme is that the encapsulating frame is an IP/UDP packet. Hence, the number of virtual network segments is limited only by the size of IP subnets that can be defined within the data center and not by the number of ports in virtual switches as in the case of VLAN-based network segmentation. Further, by using internal, non-routable IP addresses for VMs (using DHCP and NAT capabilities) running within virtualized hosts, the number of virtual networks that can be realized is even higher and (c) The VLAN scheme uses the Spanning Tree Routing Protocol to forward packets, VXLANs can use the ECMP protocol of Layer 3 [11], thus efficiently utilizing all available network links in the network fabric of the data center.

The disadvantage of a network segmentation approach based on Overlay-based networking technology is that it requires large mapping tables in each virtual switch level in order to generate overlay packets – since the MAC address of the destination VM could be located in any IP subnet and any host in the data center. Building these mapping tables using just a flooding technique is inefficient. Hence, a control plane needs to be deployed in the virtualized infrastructure to populate the mapping tables for use by the overlay packet generation module in the hypervisor. This creates an additional layer of control and adds to the complexity of network management [11].

III. TRAFFIC CONTROL IN A VIRTUAL NETWORK

Traffic control in a virtual network can be performed using either a virtual firewall or a physical firewall. However, in a virtualized infrastructure, the computing nodes (whose incoming/outgoing traffic needs to be controlled) are VMs and are end nodes of a virtual network. Hence, the deployment of a physical firewall will require the traffic from the virtual network to be diverted into the physical network (where the physical firewall resides) and

 ISBN: 978-1-61208-460-2

then back into the virtual network. This extra route travelled by communication packets will result in latency and consequently reduced performance of applications hosted on VMs. Hence, in this paper, we consider only virtual firewall-based solutions for traffic control in virtual networks for securing VMs.

Earlier in this paper (Section 2), we saw that two or more virtual firewalls can be used to create network segments in a virtual network. Since the focus of this Section is on traffic control, we are going to analyze the use of virtual firewalls only for controlling inter-VM traffic.

Inter-VM traffic can be of two kinds: the traffic between two VMs residing on the same virtualized host (either connected to the same or different virtual switches) and traffic between two VMs hosted on different virtualized hosts. Traffic between two VMs residing on the same virtualized host can only be enabled if each VM has at least one vNIC (a VM can have multiple vNICs just like a physical server can have multiple physical network interface cards or network adapters) connected to a common virtual switch. This is due to the fact that two virtual switches in a virtualized host cannot be connected to each other. Although, theoretically, a pathway between two VMs on the same virtualized host can be establishing by routing the traffic from one VM (say VM1) to the physical network (through one physical NIC) and back into the same virtualized host (through another physical NIC) to the target VM (say VM2), this is not a viable option in most situations, due to the latency issue referred to above. Of course, for enabling traffic between two VMs residing on two different virtualized hosts, the traffic has to travel from the virtual network (in the originating virtualized host) where the originating VM resides, through the physical network of the data center and back again into the virtual network of the target VM (in the target virtualized host).

Virtual firewalls come in two flavors: (a) VM-based – this class of virtual firewalls, comes packaged as a virtual security appliance on a specially-configured VM and (b) Hypervisor Kernel-based – this class of firewalls operates as a kernel loadable module in the kernel of the hypervisor.

A. Traffic Control using VM-based Firewalls

A VM-based firewall, as the name indicates, is a firewall software that runs in a VM. It can be installed as a software module on a guest VM already running in a virtualized host or it can be packaged as a virtual security appliance on a specially prepared VM instance. Its location within the virtual network of a virtualized host is critical as its function is to monitor, drop or forward packets between sets of VMs belonging to different security zones. This is the reason that this class of firewalls is called bridge-mode firewalls as it also acts as a bridge between zones (since the only link between VMs connected to two different virtual switches is through a VM with vNICs connected to both virtual switches).

A.1 Analysis

The advantage of VM-based firewall for traffic control is that since it is available as a Virtual Security Appliance, it is easy to deploy and configure in a virtualized host. Its initial location within the virtual network of the virtualized host is relatively easy as it is dictated by the layout of the security zones based on the various virtual switches and this type of firewall only monitors and filters packet flows between one virtual switch and another.

The disadvantages of VM-based firewalls are: (a) It cannot monitor and filter traffic flowing between two VMs connected to the same virtual switch, (b) Its performance is limited by the number of virtual CPU cores allocated to the VM it is residing or packaged in, and (c) All traffic flowing into and out of all portgroups and virtual switches associated with zones pertaining to this firewall, has to be redirected to this firewall causing unnecessary traffic (a phenomena called Traffic Trambones [12]).

B. Traffic Control using Hypervisor Kernel-based Firewalls

Hypervisor kernel-based firewalls are also called hypervisor-mode firewalls and VM NIC firewalls. These firewalls install as a hypervisor module along with a VSA, the latter used purely for initial configuration (and re-configuration) for the hypervisor module. Hence, the main firewall functions of monitoring and packet filtering are done in the hypervisor kernel-module with the VM hosting the VSA portion playing the role of a Control or Service VM. Logically residing between a VM vNIC and the hypervisor virtual switch, this type of firewall provides a vNIC-level firewall enforcement point for traffic to and from VMs. Thus they can be used for selectively protecting a given subset or all the VMs in a host or a cluster. Because of the visibility at the vNIC level, these firewalls can protect traffic flowing between two VMs connected to the same virtual switch, unlike the bridge-mode firewalls. Another distinguishing feature of this type of firewalls is that the firewall does not require any changes

 ISBN: 978-1-61208-460-2

to the virtual network configuration inside a virtualized host (such as additional network pathways for redirecting traffic to the VM hosting firewall) or modification to IP addresses of VMs.

B.1 Analysis

The advantages of hypervisor kernel-based firewalls are: (a) Their performance is of an order of magnitude much higher than bridge-mode firewalls since they perform packet inspection from within the kernel at native hardware speeds rather in a VM where the performance is limited by the capacity of virtual CPU allocated to it [13], (b) These perform monitoring at the VM NIC (vNIC) level and hence all firewall rules (or ACLs) and state are logically attached to the VM interface. Hence these rules and state move with the VM when it migrates from one virtualized host to another, thus providing continuity of security protection for a VM irrespective of its location [12], and (c) Implementations that support firewall rules at a higher level of abstraction than IP addresses or ports, can be used to filter packets at data center, host cluster and port group levels.

The disadvantages of hypervisor kernel-based firewalls are: (a) This class of firewalls works as a managed kernel process and is therefore neither a VM resident program nor is part of the virtual network of the hypervisor. Hence conventional management tools having access only to VMs or virtual networks cannot be used to monitor this class of firewalls and (b) Some of the implementations of this class of firewall support only 5-tuple based rules (Source and Destination IP Address, Source and Destination Ports and Protocol). They do not support higher level abstractions such as Security Groups, Zones or Containers. However, some of the latest offerings do support firewall rules based On higher level abstractions and flow statistics as well.

IV. SUMMARY & CONCLUSIONS

In this paper, we performed a detailed analysis of two network segmentation approaches and two virtual firewall types for the protection of VMs in virtualized infrastructures. Comparing the features of the two network segmentation techniques, we find that the only distinct advantage that overlay-based network segmentation (such as VXLAN) holds over the VLAN-based approach is its infinite scalability. Hence, unless the number of VMs in the data center is in the order of thousands, the VLAN-based approach provides a satisfactory outcome in terms of performance and for meeting the goal of securing VMs. Because of this, the VLAN approach is economically justifiable in many environments. Further justification comes from the fact that overlay-based network segmentation schemes require sophisticated virtual switches, large mapping tables and the overhead of creating a control plane using SDN controllers.

Analyzing the relative advantages and disadvantages of the two firewall types – VM-based and hypervisor kernel-based – we find that the hypervisor kernel-based firewall is superior to the VM-based one in terms of three features. They are: (a) Performance (executes in the hypervisor kernel instead of in a VM), (b) Reduced network traffic (no diversion of traffic needed from various switches to the VM hosting the firewall) and, (c) Firewall rules are associated with the VM interface (since it is placed between a VM NIC and the virtual switch) and move with VM. The third feature is very critical from the point of view of the security of the VM, since it provides continued protection to it even when it migrates to different virtualized hosts or host clusters within the data center, without any additional re-configuration. It is this overwhelming security assurance feature that makes the hypervisor kernel-based firewall, the security software of choice in many virtualized infrastructures.

REFERENCES

[1] D. Shackleford, Virtualization Security – Protecting Virtualized Environments, Wiley Publishing Inc, Indianapolis, IA, USA, 2013

[2] "DMZ Virtualization with VMware Infrastructure". [Online]. http://www.vmware.com/files/pdf/dmz_virtualization_vmware_infra_wp.pdf [retrieved: Jan, 2016].

[3] "MAC Bridges and Virtual Bridged LANs". [Online]. https://www.ietf.org/meeting/86/tutorials/86-IEEE-8021-Thaler.pdf [retrieved: Dec, 2015].

[4] "IEEE 802.1Q Virtual LANs (VLANs)". [Online]. http://www.ieee802.org/1/pages/802.1Q.html [retrieved: Dec, 2015].

[5] A. Hameed, and A. N. Mian, "Finding Efficient VLAN Topology for better broadcast containment," Third International Conference on the Network of the Future (NOF), Gammarth, Nov 2012, pp.1-6.

[6] Introduction to Virtualized Networking. [Online]. http://www.ipspace.net/Introduction_to_Virtualized_Networking [retrieved: Dec, 2015].

[7] Overlay Virtual Networking. [Online]. http://www.ipspace.net/Overlay_Virtual_Networking [retrieved: Dec, 2015].

 ISBN: 978-1-61208-460-2

[8] “Overlay Virtual Networking and SDDC”. [Online]. http://my.ipspace.net/bin/list?id=xSDNOverlay [retrieved: Jan, 2016].
[9] “Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks”. [Online]. https://tools.ietf.org/html/rfc7348 [retrieved: Jan, 2016].
[10] “VXLAN Overview: Cisco Nexus 9000 Series Switches”. [Online]. http://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-729383.pdf [retrieved: Dec, 2015].
[11] “Scaling Overlay Virtual Networks”. [Online]. http://content.ipspace.net/get/Scaling%20Overlay%20Virtual%20Networks.pdf [retrieved: Jan, 2016].
[12] Virtual Firewalls. [Online]. http://www.ipspace.net/Virtual_Firewalls [retrieved: Dec, 2015].
[13] Virtual Firewall. [Online]. https://en.wikipedia.org/wiki/Virtual_firewall [retrieved: Dec, 2015].

 ISBN: 978-1-61208-460-2

Profiling and Predicting Task Execution Time Variation of Consolidated Virtual Machines

Maruf Ahmed, Albert Y. Zomaya
School of Information Technologies, The University of Sydney, Australia
Email: mahm1846@uni.sydney.edu.au, albert.zomaya@sydney.edu.au

***Abstract*—The *task execution time variation* (TETV) due to consolidation of *virtual machines* (vm), is an important issue for data centers. This work, critically examines the nature and impact of performance variation from a new angle. It introduces a simple and feasibly implementable method of using micro and syntactic benchmarks to profile vms. It is called, the *Incremental Consolidation Benchmarking Method* (ICBM). In this method, server resources are systematically incremented, in order to quantitatively profile the effects of consolidation. Resource contention due to basic resources, like CPU, memory and I/O, have been examined separately. Extended experiments have been done on the combinations of those basic resources, too. All experiments have been done and data are collected from real virtualized systems, without using any simulator. The *least square regression* (LSR) is used on the profiled data, in order to predict the TETV of vms. To profile the TETV data, the server has been consolidated with different types and levels of resource loads. The prediction process, introduced here is straightforward and has low overhead, making it suitable to be applied on a wide variety of systems. The experimental results show that, the LSR models can reasonably well predict TETV of vms under different levels of consolidation. The *root mean square error* (RMSE) of prediction for combination of resources, like CPU-Memory, CPU-I/O and Memory-I/O, are within 2.19, 2.47 and 3.08, respectively.**

Keywords–virtualization; consolidation; performance; variation; prediction.

I. INTRODUCTION

The *virtualization* is an essential part of modern data centers. It is required for running many day-to-day operations, like deploying a fault tolerance scheme, or providing *Cloud* services. A *virtual machine* (vm) is a self-contained unit of execution, usually created with the help of a hypervisor running on top of a physical *host*. The vms are immensely important for data centers, as they help to implement the pay-as-you-go model for the Cloud. Usually, a number of vms are run on a host to reduce the operational cost. All the simultaneously running vms of a physical host are collectively known, as the *co-located vms*. The Cloud users can rent vms and have complete control over the rented vms. However, they do not have access to the underlaying physical hardware.

The *consolidation* of vms, is generally done to increase the resource utilization of virtualized servers. However, it imposes a performance penalty, which manifest itself through the *task execution time variation* (TETV) of co-located vms [1]–[3]. This performance variation happens because of resource contention among the vms. It is an obstacle to efficiently scheduling parallel applications on virtualized systems, for several reasons: (i) The variation depends on the server load and resource contention among the vms. The result is that, the same task may take different amount of time to be completed on different vms. At present there is no well accepted method to predict this variation; (ii)To schedule a task of any parallel application, the execution finish time of all the parents tasks must be known. It becomes difficult due to the TETV. Thus, it is an important issue to address, if parallel applications are to be scheduled on virtualized clusters, efficiently.

Most of the previous works on this area fall into two main categories. The first one, is to explore the cost-performance models of parallel applications on public Clouds [4]–[13]. The other one, is virtualized server consolidation benchmarking [14]–[19]. Nonetheless, one can easily identify several weaknesses of those works: (i) They do not explore the resource contention and performance variation of co-located vms explicitly; (ii) Experiments have been done, mainly with parallel applications. Those have complex internal structure of their own, usually represented by a task graph. The virtualization involves so many layers of abstraction and hardware indirection. The complex internal structures of an application can make it difficult to accurately capture the relationship among the co-located vms; (iii) They do not provide the option to control usages of different computing resources, either individually or granularly during experiments. In this case such an ability is very desirable; (iv) Consolidation benchmarks are designed to provide an overall average point of some combination of tests, not details about different levels of consolidation; (v) The consolidation benchmarks are hugely dependent on vendors and their applicability for comparing different virtualization technologies are not well defined. For example, the *VMmark* benchmark is designed for VMware ESX servers; (vi) Most works use complex mathematical optimization tools, which have high overhead. The performance modeling greatly depends on system configuration, and changes to system configuration may require model retraining, which in turn becomes hugely time consuming process due to high overhead; (vii) Many works deal with theoretically derived model of the Cloud and simulation is used for verification. Those simulations often do not take virtualization into consideration, hence does not always portray the reality.

It is clear that, a new design for the consolidation benchmark is required, to address above limitations. The *Incremental Consolidation Benchmarking Method* (ICBM) overcomes above issues, using micro and syntactic benchmarks. Some of the design features of the ICBM are discussed next, in order to explain how does it overcome above issues:

1) Micro and syntactic benchmarks suites have been used instead of parallel applications. This gives the ability to manipulate basic resources, like CPU, memory and I/O, both individually and granularly during experiments. This makes it possible to analyze the effect of consolidation on each resource type more discretely than the previous works;

2) The ICBM, at its core is agnostic to both virtualization technology and system configuration. First and foremost, it is

 ISBN: 978-1-61208-460-2

a methodology that can be applied to any system in general, making it suitable to compare a wide range of systems;

3) The TETV prediction models for combinations of resources have been built from profiled vm data. Separately collected TETV data due to basic resources, like CPU, memory and I/O, have been used to predict TETV for combination of resources, like CPU-Memory, CPU-I/O and Memory-I/O. The prediction results have a good level of accuracy, demonstrating that profiling for every combination of resource load is not necessary. It can potentially save a huge amount of time while profiling a large data center;

5) All experiments have been done on actual virtualized servers rather than using simulators. All results presented here are real system data. The results show that, the ICBM can predict TETV of real virtualized systems quite accurately;

4) Prediction models have been built using the *Least Square Regression* (LSR), which has very low overhead. Use of LSR instead of a complex mathematical tool, makes the training and prediction process much faster. Often, changes in system configuration may require retraining of models, in such cases a low overhead process can save a lot of time;

5) Analysis of profiled data reveals some interesting patterns. For example, it shows that certain types of resource combinations can cause the task execution time of consolidated vms to degrade more rapidly than the others. This indicates that resource contention is one of the main factors, behind the average utilization of virtualized servers being so low.

The micro and syntactic benchmarks suites play an important part in the design of the ICBM. They are important tools for server performance analysis, and a lot of studies have been done on their design. These benchmarks are the result of long time analysis of commercially successful applications. They are inspired by an interesting phenomenon that, the applications spend both 80% of their time and resources, executing just 20% of the code [20]. The micro and syntactic benchmark suites are carefully crafted to mimic such important parts rather than running the entire application. Each benchmark suite consists of several individual applications. These applications are grouped together in a well-thought-out pattern.

The benchmarks suites are used here, to get a finer control on the individual server resource types. Without using these benchmark suites, such controlling of resources is not possible. Experimental results show that, the benchmark suites can cause significant amount of resource contention and TETV on vms. Thus, the benchmark suites can be a set of powerful tools for studying the performance variation of virtualized servers. The experiments that are done here, are very comprehensive and provide some interesting results.

Rest of the paper is structured as follows. The main objectives of experiments are discussed in the Section II, followed by the methodology described in the Section III. Sections III-B and III-A briefly discuss the benchmarks used in the experiments and experiential setup, respectively. Detail results of experiments and prediction are given in the Section IV. Finally, Section V concludes the paper, with an overview of the future work.

II. Problem Description

This section describes the objectives of the experiments. The first objective, is to quantitatively measure the effect of consolidation on the task execution time of vms. The logical view of a virtualized server is shown in the Figure 1a. Here, a typical situation has been considered. Usually, a physical server hosts vms of the same configuration, however number of simultaneously running vms may change over time. Different vms may also have different types of resource requirements. Some of them may be CPU or memory intensive, while others may be multiple resources intensive. As the number of vms increase or decrease on a physical host, it is important to know the TETV of individual vms. The objective here, is to offer a systematic way to record the TETV due to different numbers of vms, and find a way to predict the TETV.

The second objective, is to establish a relationship between TETV due to basic types of resource contention and that of combination of resources. Figure 1b depicts the situation of a server from resource usages point of view. In each graph, the x-axis represents the number of vms simultaneously running on the host. The y-axis represent the task execution time of a vm. As the number of co-located vms increase, the task execution time starts to vary.

The actual amount of variation depends on types and amount of resource loads. For three basic resources types, namely CPU, memory and I/O (Figure 1b(i-iii)), the amount of TETV are excepted to be different. Again for combination of resources, like CPU-memory, CPU-I/O and memory-I/O (Figure 1b(iv-vi)), the TETV would be different, too. Profiling a virtualized system is difficult for many reasons. A server may be running different number of vms, with different amount of loads at different points. That makes it impractical, to profile all vms, for all combination of resources. Establishing a relation, among the TETV due to basic resource types and that of combination of resources, would save a lot of time while profiling a large data center. Next, the procedure used in the ICBM is discusses in details.

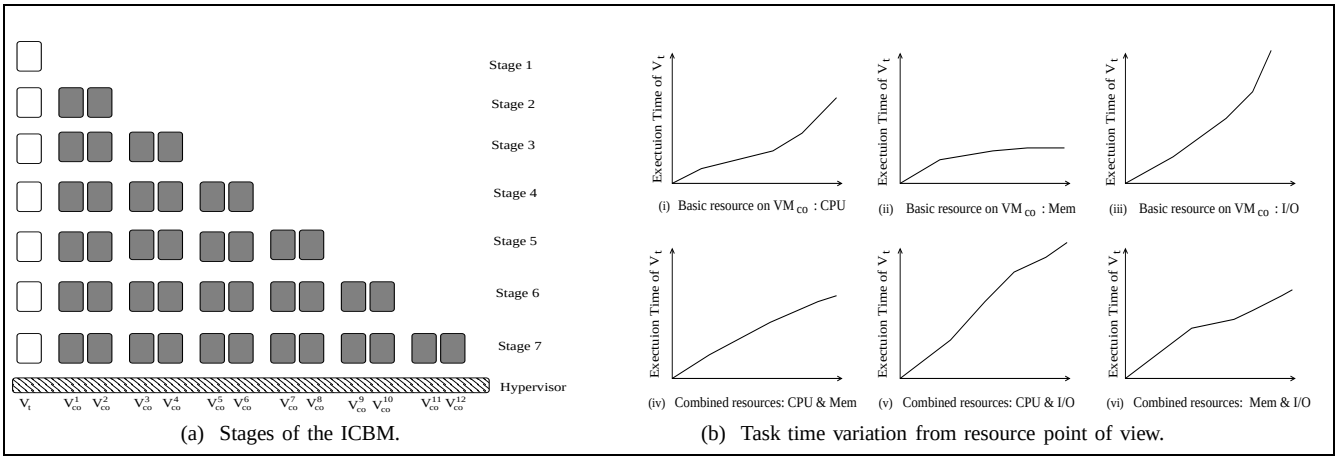
III. Methodology

This section introduces the terminologies and methodology that have been used for rest of the paper. One of the vms of host, is designated as the target vm (v_t), while rest are designated as co-located (v_{co}) vms. The v_t has been used to run different tasks to record their TETV. On the other hand, v_{co} have been used to collectively create resource contention.

In Cloud, the data is generally stored on dedicated nodes over the network. However, before processing, all data is retrieved into the local node. For example, a MapReduce worker node performs all the mapping and reducing steps on local data. In fact, MapReduce nodes rely more on local data, and try to consume as little bandwidth as possible [21]. That way, local I/O contention has much more impact compared to that of network I/O. What is more, in order to address the overall I/O contention issue, it is necessary to address the local I/O contention first [22]. This work provides a quantitative way to measure the I/O contention of vms of a node. The network I/O system consists of several such nodes. Taking this work as a basis, the issue of network I/O contention can be addressed through extended experiments.

A parallel application can have many different data flow paths. As mentioned in the introduction, such an application can be decomposed into a set of tasks. Then, those tasks need to be scheduled on different vms individually. This work lays ground, for understanding the effect of consolidation at vm

 ISBN: 978-1-61208-460-2



(a) Stages of the ICBM. (b) Task time variation from resource point of view.

Figure 1. Experimental setup and objective of ICBM.

level. Without this understanding, it is not possible to formulate the effect on consolidation at a higher level.

The micro and syntactic benchmarks used here, are actually application suites. They are combinations of several applications, patterned together to execute such a way that they put maximum pressure on a particular system resource. For example, the Filebench includes several different categories of tests, including a file-server test [23]. Similarly, the Unixbench consists of many tests, including the System Call Overhead, Pipe-based Context Switching and Execl Throughput test [24]. The Nbench includes, a task allocation algorithm, Huffman compression, IDEA encryption, Neural Net and LU Decomposition test [25]. There are many more test components on those suites, and all of them are designed to mimic the steps of commercial applications. Thus, together they can put the vms under stress like a real application. Also the experimental results from real systems show that, these benchmarks cause the task execution time to vary significantly. Thus, they are well capable of creating resource contention like a real application.

Initially, v_t run one task alone on the host (stage 1 in Figure 1a) and execution finish time is recorded. In successive stages, co-located vms are added in a particular pattern to create resource contention. Manipulating each co-located vm, independently at each step, makes it possible to increase a particular type of resource load at a desired quantity. In the experimental setup, two v_{co} are added at each stage until the server reaches the saturation point. Depending on the configuration, a host can simultaneously run only a certain number of vms, without becoming unresponsive. That is considered to be the saturation point. As co-located vms with different resource types are added at each stage the TETV of all vms are profiled.

Figure 2 provides the pseudo code of ICBM for basic resource types. The ICBM repeats the same steps for each resource type, on co-located vms. Therefore, it is not necessary to explain the steps, for each resource type. Next, these steps are explained for one type of resource load, the CPU load.

Let t be a task whose TETV due to CPU intensive co-located vms is going to be investigated. First, the t is run alone on a single vm (v_t) of the host (stage 1 of Figure 1a), in order to obtain the execution finished time, without any interference from co-located vms. Next, (stage 2) the t is run again, this time along with two simultaneously running co-located vms (v_{co}^1 and v_{co}^2). Both the new vms, run one B_{cpu} type benchmark each, thus only increasing CPU intensive load on the system. This gives the execution time of t on v_t, which is now consolidated with two co-located CPU intensive vms. Afterwards, two more CPU intensive v_{co} are added (stage 3), increasing the number of CPU intensive co-located vms to four. The execution finish time of v_t is profiled again for this setting. This way, two new v_{co} are added at each stage, until the co-located vms stop responding.

```
T ← A set of tasks.
B_cpu ← A set of CPU intensive benchmarks.
B_mem ← A set of memory intensive benchmarks.
B_i/o ← A set of I/O intensive benchmarks.
v_t ← A target vm.
v_co ← A set of co-located vms.
for Each task t ∈ T do
    Add v_t to the host.
    Run t on v_t alone and record the execution finish time.
    for Each benchmark type B ← B_cpu, B_mem, B_i/o do
        while All the vms of host are responding do
            Add two extra v_co with benchmark of type B to host.
            Run v_t simultaneously with the added v_co, and
            record the execution finish time of each.
        end while
        Remove all v_co and free related system resources.
    end for
end for
```

Figure 2. Basic steps of ICBM.

The maximum number of vms, that can be simultaneously run on a host without making it non-responsive, is dependent on the system configuration. In the experiments, the host had one *Intel i7-3770* processor, with four cores and eight hardware threads. Assigning, a logical CPU to each vm, the system could run maximum of fourteen such vms, simultaneously. Adding anymore vm, would made the whole system non-responsive. With one vm designated as v_t, and two more new v_{co} added at each stage, the final stage (stage 7) of the experiment had thirteen simultaneously running vms (one v_t along with twelve v_{co}). The vms are created with least possible subset of CPU resources, so that CPU load can be increased with fine granularity. However, vms with larger number of logical CPU

can be also created, if such is required.

The same steps are repeated for memory intensive v_{co}, too. However, in this case memory intensive benchmarks are run on co-located vms instead of CPU intensive benchmarks. All vms are configured to have 1 GB of RAM. The experiment starts (stage 1) with a single vm (v_t) running the task, whose TETV due to memory intensive co-located vms is going to be investigated. Next, (stage 2) v_t is run again, along with two new co-located vms (v_{co}^1 and v_{co}^2), each running one memory intensive benchmark. Similarly, two more vms (v_{co}^i and v_{co}^{i+1}) are added at each stage until the system reaches a predetermined point of memory load. In this case, adding a new v_{co} makes the host memory load to be increased by 1 GB. It was done so that, the host memory load can be changed granularly. The host has 16 GB of RAM. By restricting the number of vms to thirteen at the final stage, maximum of 13 GB RAM is allocated to the co-located vms, leaving rest for the hypervisor. As with the CPU load, this predetermined load is also not a fixed number. Afterwards, for I/O load the same steps are repeated by adding two I/O intensive benchmarks on two v_{co}, at each stage. Thus, the TETV for three basic resource types (CPU, memory and I/O) are collected.

Next, the procedure is repeated for **resource combinations**. The combinations are made by choosing two resource types at a time, from previously mentioned basic three. Those are CPU-Memory, Memory-I/O and I/O-CPU. Experiments for combination of loads are done exactly the same way. That is, start with one vm (v_t), and at each stage add two co-located vms (v_{co}^i and v_{co}^{i+1}) to increase the load. Difference is that, now two new vms run two different types of benchmarks. Both of them, together create the effect of load combinations. For example, to increase CPU-Memory load, two v_{co} are added at each stage. One (v_{co}^i) runs a CPU intensive benchmark, while the other one (v_{co}^{i+1}) runs a memory intensive benchmark.

Above experiments demonstrate, how the execution time of v_t is varied due to co-located vms (v_{co}^i). However, there is another angle to this problem, that is how the v_{co} are collectively effect the execution times of each other. To examine this, the execution finish times of all the v_{co}^i, are also recorded at each stage. Finally, the whole procedure is repeated without the v_t altogether. That is, all the experimental steps are repeated, by adding only load (on v_{co}^i) at each stage.

Advantage of the ICBM is that, the server load can be changed granularly, for each basic resource type or their combinations. Furthermore, there is no need to know the internal structure of a complex application, which is running on the vm. Establishing a relationship between task execution time and the server load would be helpful for a virtualized data-center in many ways, including designing heuristic algorithms for scheduling, consolidation and live-migration of vms [26][27].

A. Experimental setup

A Dell XPS-8500 system has been used in the experiments. It has one Intel i7-3770 processor and 16GB of RAM. The i7-3770, has four cores and eight hardware threads, each clocked at 3.4 GHz. The host is deployed with Xen 4.2 over Fedora 17 core. Fourteen vms have been setup, one to acts as v_t, while the rest are v_{co}^i. Each vm run Fedora 16, have one logical CPU, 1 GB of RAM and 30 GB virtual disk. The vms are not pinned to the logical cores. Total of 13 GB RAM is allocated to the vms, rest are available for the hypervisor. All the benchmarks are installed on vms beforehand, and a concurrent java application manages all benchmarks and vms from a remote node.

B. Benchmarks used

This section gives an overview of the benchmark suites, used in the experiments. A data center may be running thousands of servers. While designing an algorithm for consolidation or live migration scheme, the internal structures of all the parallel applications may not be known. It is more realistic to characterize the servers by their loads. Micro and syntactic benchmark suites are well suited for such purposes.

Three different categories of benchmarks have been used for three categories of resources, namely CPU, memory and I/O. The *Nbench* [25] and *Unixbench* [24], are the two CPU intensive benchmark suites used here. Two memory benchmark suites used here, are the *Cachebench* [28] and *Stream* [29][30]. Finally, three I/O benchmark suites have been used, they are the *Dbench* [31], *Filebench* [23] and *Iozone* [32]. For each benchmark, several different parameters are need to be set. Owing to the space limitation, it is not possible to list all the parameters here. A fuller discussion about the benchmarks, is out of scope for this paper. Interested readers are encouraged to refer to the citations of respective benchmark suites for details.

IV. RESULTS

The results for prediction are given in the Section IV-C. However, to interpret the prediction results, it is necessary to discuss some observations of the experimental results. It will also help to clarify the training and testing data format, which has been used during the training and prediction phrases.

A. Execution time variations of the target vm (v_t)

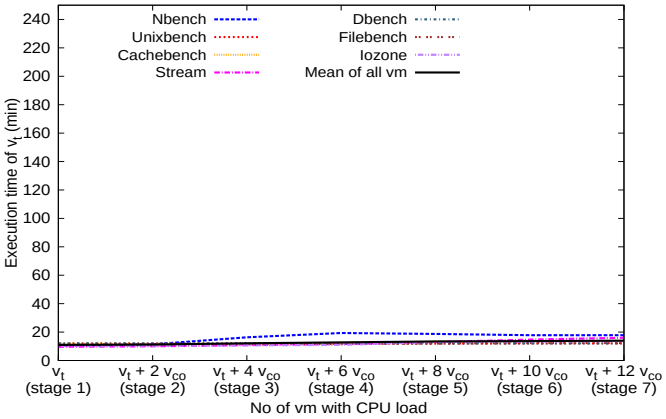
Three graphs of Figure 3, show the TETV of v_t due to three basic types of resource loads. They are being, CPU (Figure 3a), memory (Figure 3b) and I/O (Figure 3c). In each graph, the x-axis shows the total number of vms running on the system, while y-axis presents the task completion time on v_t. First point of the graph, shows the execution time of v_t, when it is run alone on the server. At this stage v_t is free from any interference from co-located vms. As explained in the Section III, two co-located vms (v_{co}) are added to the server at successive stages. In the final stage, there were twelve v_{co}, simultaneously running besides the v_t. In other words, from left to right along the x-axis, the interference from co-located vms increases. The first point of each graph, gives execution time of the task without any interference from co-located vms. On the other hand, the last point gives the execution time with maximum interference from co-located vms. Results clearly show that different types and number of v_{co}, make the task execution time to vary at a different rate.

Figure 3a shows that, the TETV of v_t with the increase of CPU intensive v_{co}^i, is the minimum among all types of workloads. On the other hand, for memory intensive v_{co} (Figure 3b), two CPU intensive tasks (Nbench and Unixbench) on v_t show less variation compared two memory intensive tasks (Cachebench and Stream) under the same situation. As an example, in isolation the Nbench takes 10.1 minute to execute on v_t. With other 12 memory intensive co-located vms (v_{co}) it extends to 12.83 minutes, which is only 27.02% longer. In contrast, the Cachebench under the exact same setup takes 587.58% longer to execute (execution time goes from 11.76

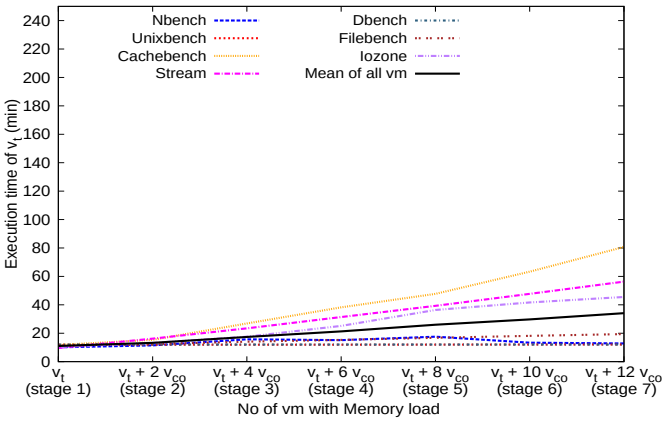
 ISBN: 978-1-61208-460-2

min to 80.68 min). Figure 3c shows the TETV due to I/O intensive v_{co}. Here, CPU intensive tasks do not show much performance degradation, while both the memory and I/O intensive tasks do. For example, the Cachebench (a memory intensive task) have 1057.311% increase in execution time, while Iozone (an I/O intensive task) have 1482.93%.

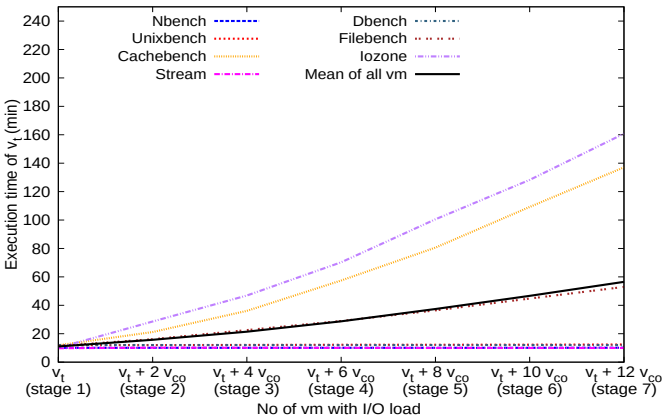
Next, Figure 4 shows the TETV on v_t, when combination of loads have been used on v_{co}. The combinations are being CPU-Memory (Figure 4a), CPU-I/O (Figure 4b) and Memory-I/O (Figure 4c). Figure 4a shows the TETV on v_t due to a



(a) Basic resource type: CPU.

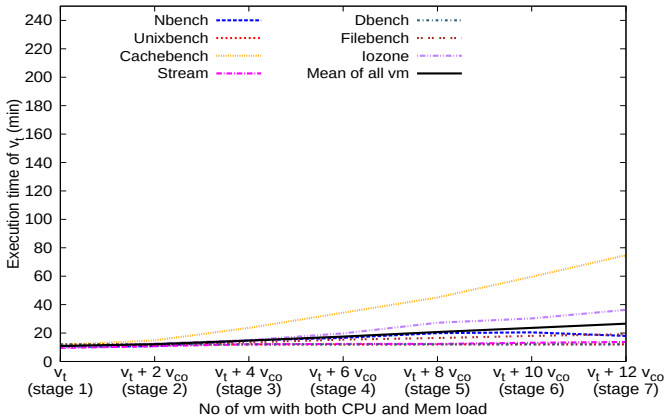


(b) Basic resource type: Memory.

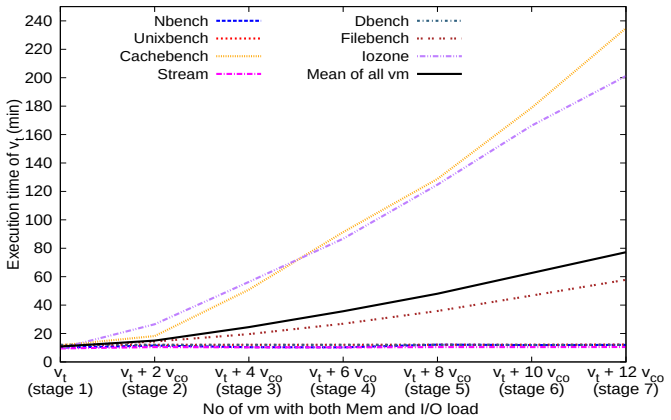


(c) Basic resource type: I/O.

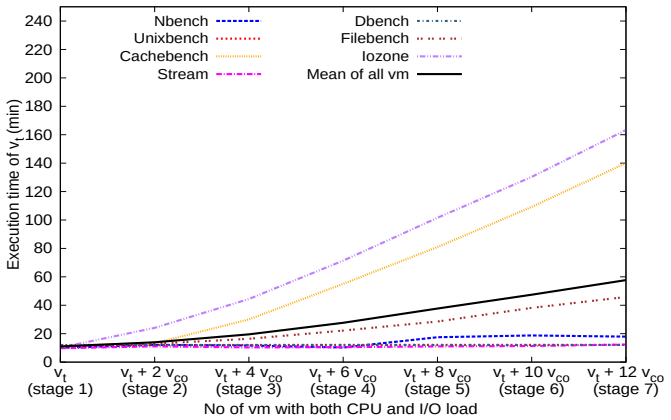
Figure 3. The TETV of v_t due to various basic types of resource load on v_{co}.



(a) Combination of resources: CPU-Memory.



(b) Combination of resources: Memory-I/O.



(c) Combination of resources: CPU-I/O.

Figure 4. The TETV of v_t due to different combinations of resource load on v_{co}.

mix of CPU and memory intensive v_{co}. Here, CPU intensive tasks on v_t show least amount of degradation just as observed previously. Among the memory intensive benchmarks, the Cachebench shows highest rate of performance degradation (542.74%). However, for I/O intensive tasks the effect of CPU-Memory v_{co} combination, is less adverse compared to memory intensive tasks. Figures 4c and 4b, show the performance degradation of v_t, when I/O intensive v_{co} is coupled with CPU and memory intensive v_{co}, respectively.

In both cases, memory and I/O intensive tasks on v_t show comparatively more performance degradation. In the case

 ISBN: 978-1-61208-460-2

of CPU-I/O load (Figure 4c), the Cachebench and Iozone show increase of execution time by 1907.21% and 1991.67%, respectively. Again for Memory-I/O load (Figure 4b), the same two benchmarks show worse performance degradation (1100.09% and 1502.56%, respectively). Table I shows the *standard deviation* (SD) of execution times of all seven tasks on v_t through stage 1 to 7.

TABLE I. STANDARD DEVIATION OF TASK EXECUTION TIMES (MIN).

	Task type	CPU intensive		Mem. intensive		I/O intensive		
	Load type	Nbench	Unixbench	Cachebench	Stream	Dbench	Filebench	Iozone
Basic	CPU	3.66	0.05	0.99	2.42	0.00	0.41	1.51
	Memory	2.58	0.05	25.28	16.89	0.01	2.97	14.18
	I/O	0.01	0.15	46.59	0.12	0.02	15.16	54.70
Combined	CPU-Memory	4.12	0.05	23.51	1.43	0.00	3.06	9.89
	CPU-I/O	0.88	0.08	83.85	0.35	0.01	17.33	71.62
	Memory-I/O	3.82	0.06	49.40	0.82	0.01	13.17	56.62

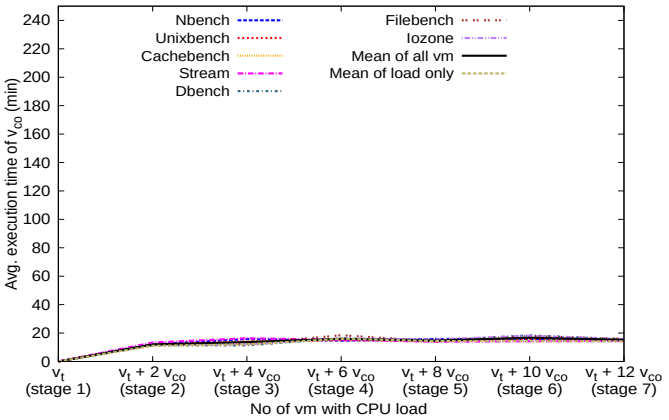
The results of this section show, how the degradation of task execution time for each resource can be presented in a quantitative way. At present, there is no standard way to compare the effect of consolidation, for different types of resources or different classes of servers. The straightforward method presented here, can be used to compare the effect of consolidation on wide varieties of systems.

B. Execution time variations of the co-located vms (v^i_{co})

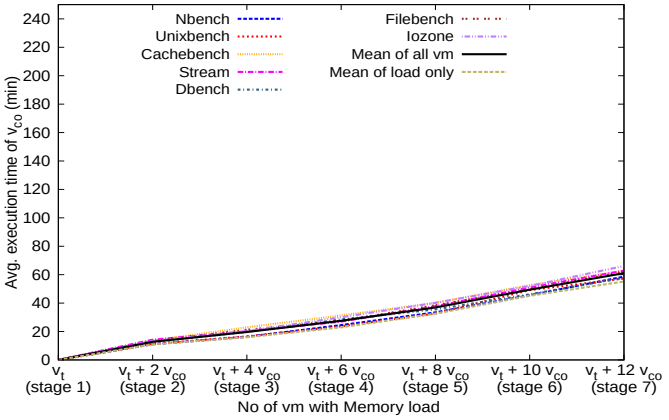
Execution finish time of all the co-located vms, at each stage are also profiled. This data is used for predicting the TETV of v_t. Figures 5 and 6 show the arithmetic mean of execution times of all v^i_{co}, at each stage, separately. During experiments, it was observed that, the arithmetic mean of execution time of all the v^i_{co} follow a pattern, for each resource type. Even though, individual v^i_{co} execution time might not have such characteristic. This, clearly indicates that overall resource usages of all vms, is what ultimately shapes the performance degradation curve during consolidation.

Both in Figures 5 and 6, the first point of each graph is always zero, as there is no v^i_{co} running on the host at stage 1 (see Figure 1a). At stage 2, two v^i_{co} are running, therefore second point is the arithmetic mean of task execution times of two vms (v^1_{co}, v^2_{co}). Similarly, third point is the arithmetic mean of values of four vms (v^1_{co}, v^2_{co}, v^3_{co} and v^4_{co}) and so on.

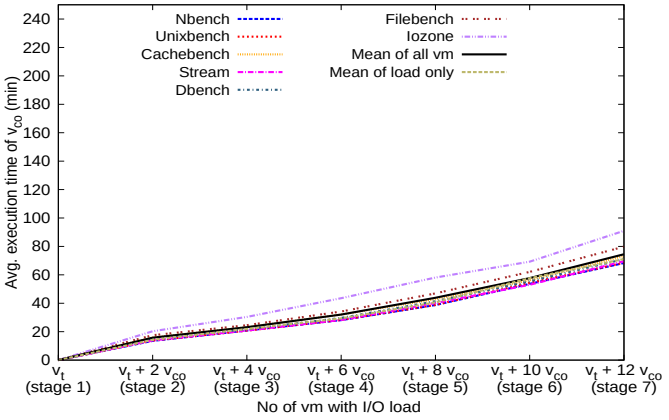
Using above procedure, each subgraph has seven arithmetic mean variation plotting of v_{co}, for seven different tasks running on v_t. Furthermore, arithmetic mean of those seven values are also shown (in black). Increase of different types of workloads causes the arithmetic mean of execution times to change differently. It can be seen that, the variation is the minimum for CPU intensive load (Figure 5a), among all three basic types of resources. Among the three combination of resources, the CPU-Memory (Figure 6a) intensive v_{co} combination shows least amount of performance degradation. On the other hand, the combination of Memory-I/O intensive v_{co} (Figure 6b) have



(a) Basic resource type: CPU.



(b) Basic resource type: Memory.



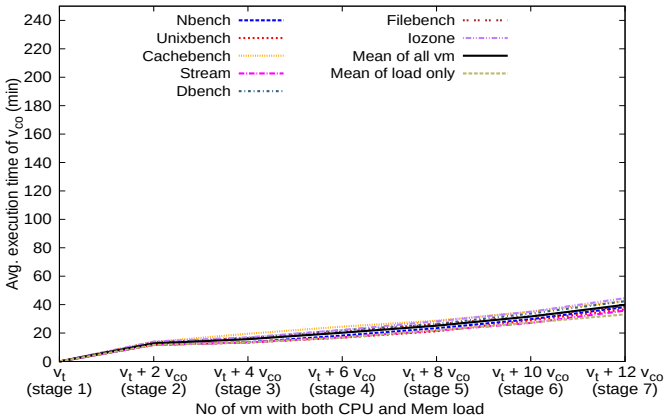
(c) Basic resource type: I/O.

Figure 5. Arithmetic mean of TETV of v_{co} with respect to three basic resource types load changes.

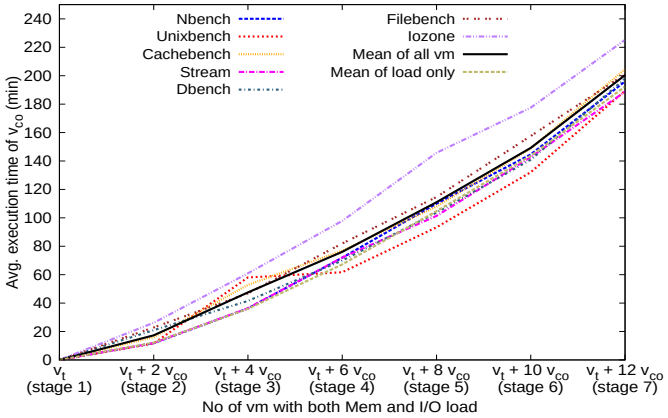
debilitating effect on the system as the arithmetic mean of execution time rises rather quickly, compared to other cases.

In order to obtain above execution time values, each experiment has been repeated at least three times. The order of adding vms has been shuffled, in order to observe their effect. However, no significant difference between the results have been observed. Two vms are added at each stage, only to conduct the experiments in a uniform way. In our observation, the order of adding vms do not change the results ultimately, rather it depends on the cumulative resource usages of vms.

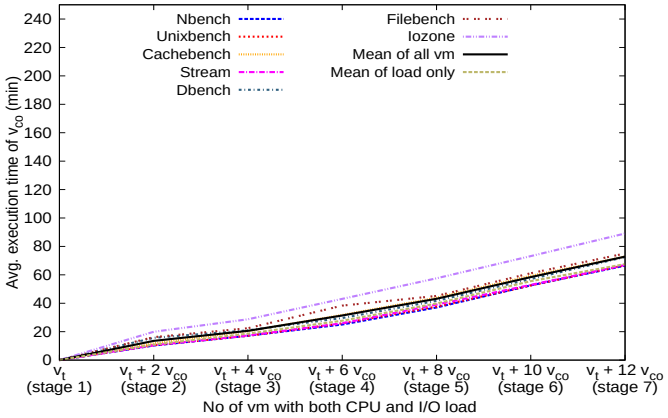
 ISBN: 978-1-61208-460-2



(a) Combination of resources: CPU-Memory.



(b) Combination of resources: Memory-I/O.



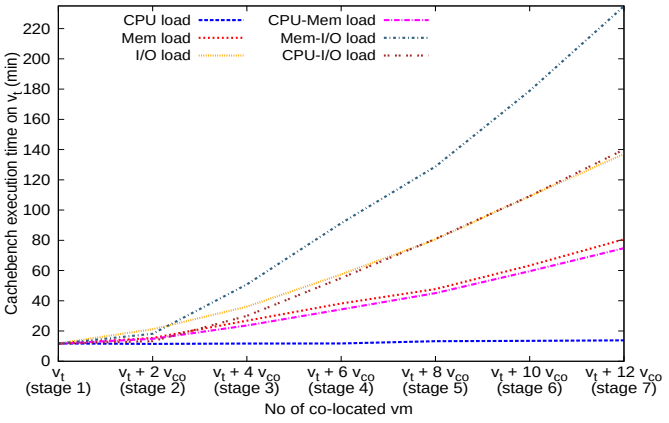
(c) Combination of resources: CPU-I/O.

Figure 6. Arithmetic mean of TETV of v_{co} with respect to three combinations of resource load changes.

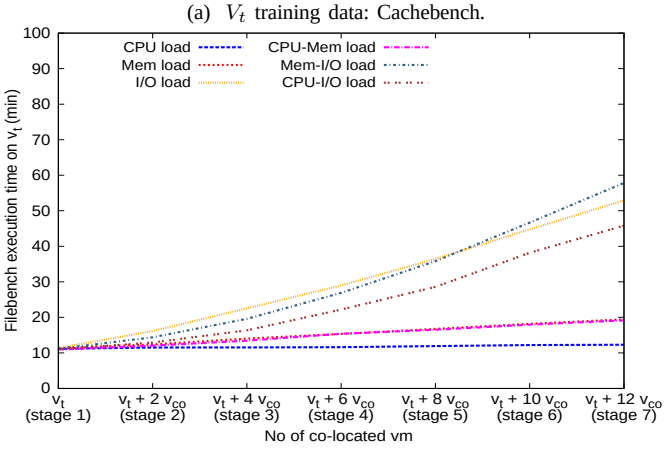
C. Task execution time variation prediction

Four benchmarks have been used as tasks for training. They are the Nbench, Unixbench, Cachebench and Stream. Three other benchmarks have been used for testing. They are the Dbench, Filebench and Iozone. All of the test benchmarks have different levels of performance degradation. Therefore, they can help to evaluate the prediction process better. Training and testing have been done on different sets of data. No training data have been used for testing, and vice versa.

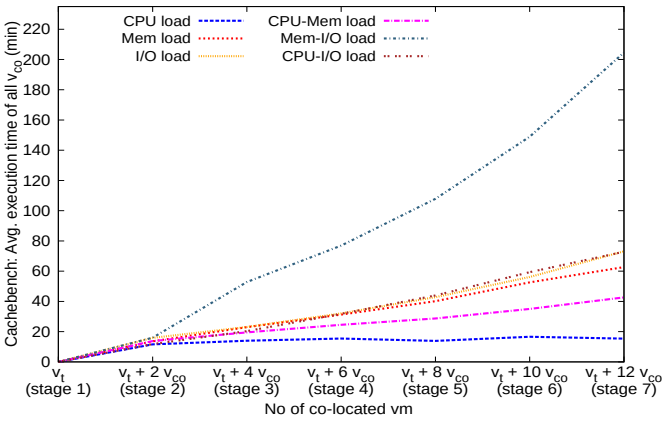
The nine subgraphs of Figure 8, separately show prediction results for three resources of three test tasks on v_t. Each subgraph, contains two sets of predictions, obtained from two separate data sets, which are described next. First set of predictions, are shown in blue on Figure 8. In this case, the TETV data of v_t for basic resource types, has been used to predict TETV of v_t for combination of resources. First, the TETV data of v_t for CPU (Figure 3a), Memory (Figure 3b) and I/O (Figure 3c) intensive v_{co} are recorded, separately. Those are used as inputs. Then, three resource combinations have been used as three separate targets, which are CPU-Memory (Figure 4a), CPU-I/O (Figure 4c) and Memory-I/O (Figure 4b).



(a) V_t training data: Cachebench.

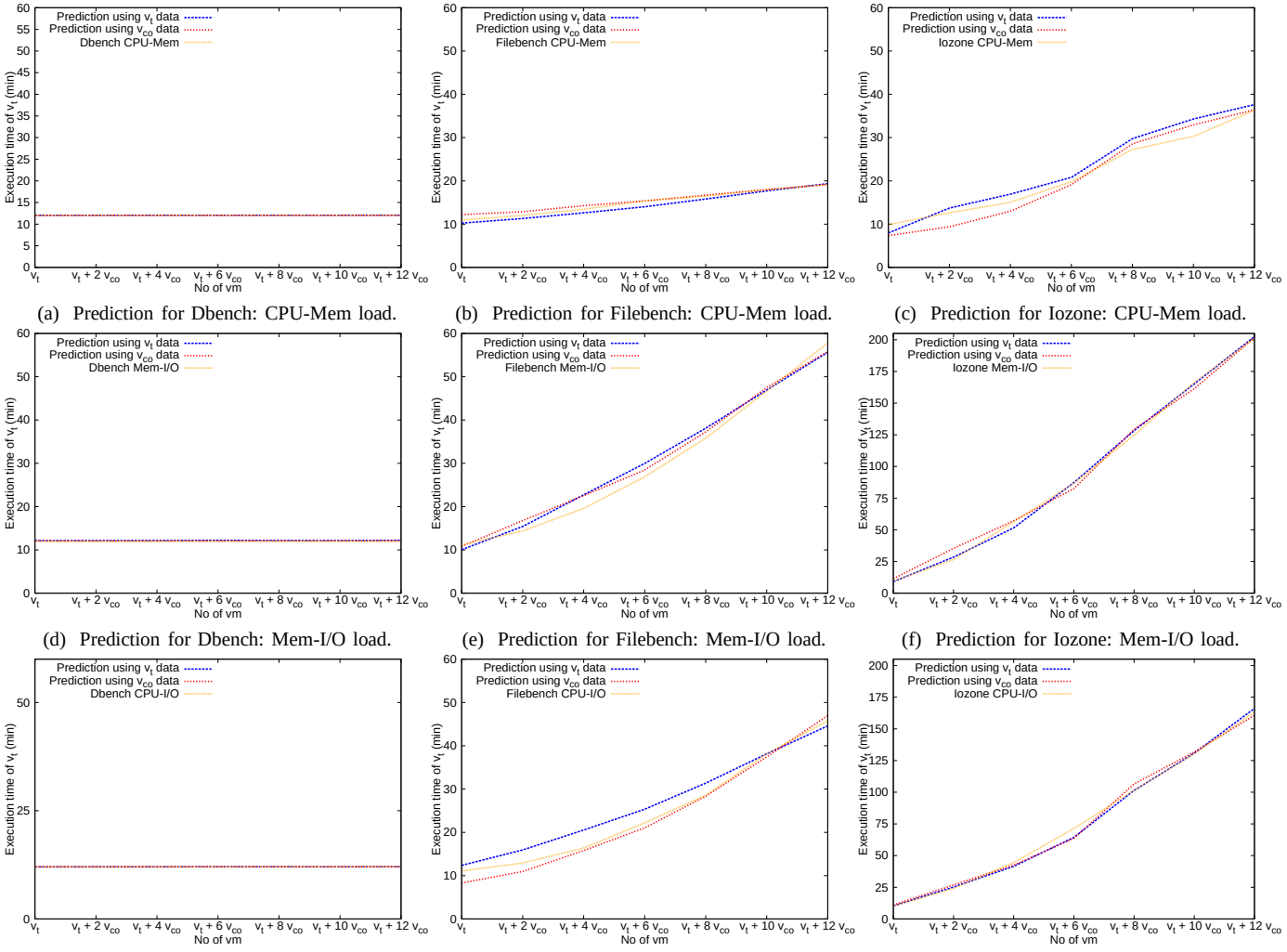


(b) V_t & V_{co} testing data: Filebench.



(c) V_{co} training data: Cachebench.

Figure 7. Examples of input and target data used for both training and testing.



(a) Prediction for Dbench: CPU-Mem load. (b) Prediction for Filebench: CPU-Mem load. (c) Prediction for Iozone: CPU-Mem load.

(d) Prediction for Dbench: Mem-I/O load. (e) Prediction for Filebench: Mem-I/O load. (f) Prediction for Iozone: Mem-I/O load.

(g) Prediction for Dbench: CPU-I/O load. (h) Prediction for Filebench: CPU-I/O load. (i) Prediction for Iozone: CPU-I/O load.

Figure 8. Task execution time prediction from v_t and v_{co} data.

Input & target data example. In above Figures 3 and 4, data is grouped according to resource loads. It is done to demonstrate that, different types of resources influence the task execution time differently. Thus, it is important to study the effect of each resource type separately. However, for training and prediction, input data needs to be grouped benchmark wise. It is required, because during training, the TETV data due to basic resources of a task, needs to be linked to the TETV data due to combination of resources of the same task. Otherwise, training would not be done with a consistent data.

Figure 7a shows an example, of this rearranged input and target data set that is used for training. It combines the TETV data of the Cachebench on v_t, for three basic and three combined types of v_{co} loads from Figures 3 and 4, respectively. All benchmarks data are rearranged similarly. During training, three basic resources data (CPU, Memory and I/O) are used as inputs and three combination of resources data (CPU-Memory, CPU-I/O and Memory-I/O) as targets. From the training data set, three sperate models have been generated, for three sets of target data (CPU-Memory, CPU-I/O and Memory-I/O). All three targets use the same three basic resources data as inputs.

An example of test data set is shown in Figure 7b. It combines six TETV data of the Filebench, from Figures 3 and 4. Prediction results for three resources (CPU, Memory and I/O) of three test benchmarks (Dbench, Filebench and Iozone) are shown in Figure 8. The *root mean square error (RMSE)* for this set of predictions are shown in Table II.

Training with c_{co} data. The second set of predictions (shown in red on Figure 8), are obtained by training the models only with v_{co} data. This demonstrates an interesting phenomenon, that models generated through use of co-located vm (v_{co}) data only, can predict the execution time variations of target vm (v_t), too. During training the CPU (Figure 5a), memory (Figure 5b) and I/O (Figure 5c) data of v_{co} are used as inputs, while CPU-Mem (Figure 6a), Mem-I/O (Figure 6b) and CPU-I/O (Figure 6c) data of v_{co} used as targets. On the other hand, during testing the v_t data are used as both input and target. All testing have been done with the same three benchmarks (Dbench, Filebench and Iozone), that have been used in the previous section for testing. That is, in this case models are generated using only co-located vms data, while testing is done with target vm data.

 ISBN: 978-1-61208-460-2

$$T^t_{cpu-mem,i} = -20.415 + 4.795 * (T^t_{cpu,i})^{0.7} + 0.752 * (T^t_{mem,i})^{0.9} + 0.579 * (T^t_{io,i}) \\ -0.119 * (T^t_{cpu,i})^{0.7} * (T^t_{mem,i})^{0.9} - 0.035 * (T^t_{cpu,i})^{0.7} * (T^t_{io,i}) + 0.002 * (T^t_{mem,i})^{0.9} * (T^t_{io,i}) \quad (1)$$

$$T^t_{cpu-io,i} = -16.393527 + 0.802 * (T^t_{cpu,i}) + 0.282 * (T^t_{mem,i}) + 1.769 * (T^t_{io,i}) \\ -0.020 * (T^t_{cpu,i}) * (T^t_{mem,i}) - 0.023 * (T^t_{cpu,i}) * (T^t_{io,i}) + 0.003 * (T^t_{mem,i}) * (T^t_{io,i}) \quad (2)$$

$$T^t_{mem-io,i} = -16.549 + 2.104 * (T^t_{cpu,i}) + 7.920 * (T^t_{mem,i})^{0.2} - 0.169 * (T^t_{io,i})^{0.96} \\ -0.957 * (T^t_{cpu,i}) * (T^t_{mem,i})^{0.2} + 0.031 * (T^t_{cpu,i}) * (T^t_{io,i})^{0.96} + 0.455 * (T^t_{mem,i})^{0.2} * (T^t_{io,i})^{0.96} \quad (3)$$

Figure 9. TETV models for resource combinations (CPU-Memory, CPU-I/O & Memory-I/O) built from v_t data.

$$T^t_{cpu-mem,i} = -0.009 + 0.190 * (T^{co}_{cpu,i})^{1.2} + 0.220 * (T^{co}_{mem,i})^{1.2} + 0.250 * (T^{co}_{io,i}) \quad (4)$$

$$T^t_{cpu-io,i} = -0.102 + 4.217 * (T^{co}_{cpu,i})^{3.25} - 0.034 * (T^{co}_{mem,i}) + 1.383 * (T^{co}_{io,i})^{0.7} \\ +0.544 * (T^{co}_{cpu,i})^{3.25} * (T^{co}_{mem,i}) - 5.394 * (T^{co}_{cpu,i})^{3.25} * (T^{co}_{io,i})^{0.7} + 0.163 * (T^{co}_{mem,i}) * IO^{0.7} \quad (5)$$

$$T^t_{mem-io,i} = 3.720 * (T^{co}_{cpu,i})^{0.325} + 2.376 * (T^{co}_{mem,i}) - 0.020 * (T^{co}_{io,i})^{0.05} \\ +1.079 * (T^{co}_{cpu,i})^{0.325} * (T^{co}_{mem,i}) - 4.394 * (T^{co}_{cpu,i})^{0.325} * (T^{co}_{io,i})^{0.05} - 3.144 * (T^{co}_{mem,i}) * (T^{co}_{io,i})^{0.05} \quad (6)$$

Figure 10. TETV models for resource combinations (CPU-Memory, CPU-I/O & Memory-I/O) built from v_{co} data.

An **example** of the training data, used here is shown on Figure 7c. It is a collection of six different v_{co} workload TETV data of the Cachebench, from six graphs of Figures 5 and 6. During training, the variation of arithmetic mean of v_{co} TETV due to CPU, memory and I/O are used as inputs. The CPU-Memory, Memory-I/O and CPU-I/O data are used as targets. Three models have been generated in this way, with three v_{co} target data set. However, testing is done with v_t test data like, the example shown in the Figure 7b. Recall that, it was also used for testing in the previous section. Table III shows the RMSE for this set of prediction. It shows that, the prediction results have a good level of accuracy.

Root mean square error (RMSE) for prediction. The RMSE of predictions for above two sets of data, are shown separately on Table II and Table III. To the best knowledge of the authors, there exist no approximation algorithm to estimate the TETV of co-located vms. Therefore, there is no acceptable theoretical bound for the RMSE values [33]. Generally, a lower value of RMSE means better prediction accuracy.

In each case, the prediction is better, when all three basic resources have been used to generate the model rather than two. For example, while predicting the TETV of CPU-Memory load combination, the model build using CPU, memory and I/O (all 3 basic resources) data produces better results, than that generated by using only CPU and memory (2 resources) data.

Parametric model. Lastly, the Figures 9 and 10 show model parameters and coefficients, as they are trained from profiled v_t and v_{co} data, respectively. They demonstrate the relation between input and prediction data formats. Three equations of the Figure 9, are for three resource combination targets. The terms are self-explanatory. For example, $T^t_{cpu-mem,i}$ denotes predicted task execution time of a task on v_t, when it is consolidated with total of i number of CPU and memory intensive co-located vms. Similarly, $T^t_{cpu-io,i}$ and $T^t_{mem-io,i}$ represent the predicted task execution time for CPU-I/O and Memory-I/O load combinations, respectively. The same input parameters have been used for all equations. $T^t_{cpu,i}$ represents the task execution time on v_t when server is consolidated with i number of CPU intensive co-located vms. Similarly, $T^t_{mem,i}$ and $T^t_{io,i}$ represent the task execution time when the server is consolidated with the same number of memory and I/O intensive co-located vms, respectively.

The equations of Figure 10 have the same targets, however inputs are different. Here, arithmetic mean of execution times of co-located vms have been used as inputs. For example, $T^{co}_{cpu,i}$ represents the arithmetic mean of execution times of i number of CPU intensive co-located vms. In this case, arithmetic mean of execution times of co-located vms have been used to predict the TETV of target vm.

TABLE II. ROOT MEAN SQUARE ERROR (RMSE) FOR PREDICTION USING TARGET VM (v_t) DATA.

	CPU-Memory Prediction		CPU-I/O Prediction		Mem-I/O Prediction	
Benchmarks	With all 3 resources: CPU, Mem, I/O	Only 2 resources: CPU, Mem	With all 3 resources: CPU, Mem, I/O	Only 2 resources: CPU, I/O	With all 3 resources: CPU, Mem, I/O	Only 2 resources: Mem, I/O
Dbench	0.036	2.516	0.146	0.506	0.011	0.468
Filebench	0.771	2.628	2.131	11.933	2.605	3.022
Iozone	**2.193**	10.505	**2.475**	21.566	**3.083**	3.588

TABLE III. ROOT MEAN SQUARE ERROR (RMSE) FOR PREDICTION USING CO-LOCATED VM (v_{co}) DATA.

	CPU-Memory Prediction		CPU-I/O Prediction		Mem-I/O Prediction	
Benchmarks	With all 3 resources: CPU, Mem, I/O	Only 2 resources: CPU, Mem	With all 3 resources: CPU, Mem, I/O	Only 2 resources: CPU, I/O	With all 3 resources: CPU, Mem, I/O	Only 2 resources: Mem, I/O
Dbench	0.006	0.317	0.020	17.621	0.014	20.182
Filebench	0.657	4.515	1.841	10.607	1.475	47.797
Iozone	2.083	43.546	4.572	87.725	3.898	59.863

Changes in basic configuration of vms, may require some modification of model parameters. The public Cloud offers vms for renting in certain number of predefined configurations. Therefore, number of such models required in reality, are limited. What is more, since LSR has low overhead, building or rebuilding of a limited number of such models would not be time consuming.

V. Conclusion and Future work

This work addresses an important issue of virtualization, the performance variations due to consolidation. A straightforward methodology has been introduced here, that is practically feasible to implement. It is different from most of the complementary works on virtualization, that employ complex mathematical tools and rely on simulation. In contrast, this work introduces a low overhead prediction process, and results are collected from real virtualized systems.

Micro and syntactic benchmark suites have been used here in a step by step process, to manipulate various vm resources individually. The methodology introduced here is unique, and results prove the effectiveness of this method. Experimental results from real virtualized systems show that, in this way it is possible to predict the TETV of vms quite accurately. It provides a new and quantitative way to explore the mutual performance interference of co-located vms. The results also provide some valuable inside into the nature of resource contention due to consolidation of vms.

The experimental results encourages one to continue working along this direction. For future work, experiments would need to be extended to a wider range of virtualization techniques and server configurations, to derive a more generalized model. More questions related to this method, can be addressed in details in future works.

References

[1] T. Janpan, V. Visoottiviseth, and R. Takano, “A virtual machine consolidation framework for CloudStack platforms,” in ICOIN, 2014, pp. 28–33.

[2] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen, “Utilization Prediction Aware VM Consolidation Approach for Green Cloud Computing,” in CLOUD, 2015, pp. 381–388.

[3] A. Tchana, N. D. Palma, I. Safieddine, D. Hagimont, B. Diot, and N. Vuillerme, “Software Consolidation as an Efficient Energy and Cost Saving Solution for a SaaS/PaaS Cloud Model,” in Euro-Par, 2015, pp. 305–316.

[4] M. Dobber, R. D. van der Mei, and G. Koole, “Effective Prediction of Job Processing Times in a Large-Scale Grid Environment,” in HPDC, 2006, pp. 359–360.

[5] V. E. Taylor, X. Wu, J. Geisler, and R. L. Stevens, “Using Kernel Couplings to Predict Parallel Application Performance.” in HPDC, 2002, pp. 125–134.

[6] W. Gao, Y. Li, H. Lu, T. Wang, and C. Liu, “On Exploiting Dynamic Execution Patterns for Workload Offloading in Mobile Cloud Applications,” in ICNP, 2014, pp. 1–12.

[7] A. Gupta, L. V. Kalé, F. Gioachin, V. March, C. H. Suen, B. Lee, P. Faraboschi, R. Kaufmann, and D. S. Milojicic, “The who, what, why, and how of high performance computing in the cloud,” in CloudCom, Volume 1, 2013, pp. 306–314.

[8] J. Simão and L. Veiga, “Flexible SLAs in the Cloud with a Partial Utility-Driven Scheduling Architecture,” in CloudCom, Volume 1, 2013, pp. 274–281.

[9] S. Xi, M. Xu, C. Lu, L. T. X. Phan, C. D. Gill, O. Sokolsky, and I. Lee, “Real-time multi-core virtual machine scheduling in Xen,” in EMSOFT, 2014, pp. 27:1–27:10.

[10] J. Zhao, J. Tao, L. Wang, and A. Wirooks, “A Toolchain For Profiling Virtual Machines,” in ECMS, 2013, pp. 497–503.

[11] A. Iosup, “IaaS Cloud Benchmarking: Approaches, Challenges, and Experience,” in HotTopiCS, 2013, pp. 1–2.

[12] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking Cloud Serving Systems with YCSB,” in SoCC, 2010, pp. 143–154.

[13] A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Y. Zomaya, and B. B. Zhou, “Profiling Applications for Virtual Machine Placement in Clouds,” in CLOUD, 2011, pp. 660–667.

[14] R. McDougall and J. Anderson, “Virtualization performance: Perspectives and challenges ahead,” SIGOPS Oper. Syst. Rev., vol. 44, no. 4, Dec. 2010, pp. 40–56.

[15] V. Makhija, B. Herndon, P. Smith, E. Zamost, and J. Anderson, “VMmark: A Scalable Benchmark for Virtualized Systems,” VMware, Tech. Rep. TR-2006-002, 2006.

[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. Apparao, R. Iyer, and D. Newell, “Towards Modeling & Analysis of Consolidated CMP Servers,” SIGARCH Comput. Archit. News, vol. 36, no. 2, May 2008, pp. 38–45.

[18] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell, “Modeling Virtual Machine Performance: Challenges and Approaches,” SIGMETRICS Perform. Eval. Rev., vol. 37, no. 3, Jan. 2010, pp. 55–60.

[19] H. Jin, W. Cao, P. Yuan, and X. Xie, “VSCBenchmark: Benchmark for Dynamic Server Performance of Virtualization Technology,” in IFMT ’08, 2008, pp. 5:1–5:8.

[20] G. G. Shulmeyer and T. J. McCabe, “Handbook of Software Quality Assurance (3rd Ed.),” G. G. Schulmeyer and J. I. McManus, Eds. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999, ch. The Pareto Principle Applied to Software Quality Assurance, pp. 291–328.

[21] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” Commun. ACM, vol. 51, no. 1, 2008, pp. 107–113. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[22] S. Ma, X.-H. Sun, and I. Raicu, “I/O throttling and coordination for MapReduce,” Illinois Institute of Technology, Tech. Rep., 2012.

[23] OpenSolaris Project, “Filebench,” URL: `http://filebench.sourceforge.net/wiki/index.php/Main_Page`, Retrieved: February, 2016.

[24] “Unixbench: BYTE UNIX benchmark suite,” URL: `http://github.com/kdlucas/byte-unixbench`, Retrieved: February, 2016.

[25] C. C. Eglantine, NBench. TypPRESS, 2012, iSBN: 9786136257211.

[26] W. Hu, A. Hicks, L. Zhang, E. M. Dow, V. Soni, H. Jiang, R. Bull, and J. N. Matthews, “A Quantitative Study of Virtual Machine Live Migration,” in CAC, 2013, pp. 11:1–11:10.

[27] S. Nathan, P. Kulkarni, and U. Bellur, “Resource Availability Based Performance Benchmarking of Virtual Machine Migrations,” in ICPE, 2013, pp. 387–398.

[28] P. J. Mucci, K. London, and P. J. Mucci, “The CacheBench Report,” URL: `www.earth.lsa.umich.edu/ keken/benchmarks/cachebench.pdf`, Retrieved: February, 2016.

[29] J. D. McCalpin, “Memory Bandwidth and Machine Balance in Current High Performance Computers,” IEEE CS TCCA Newsletter, Dec. 1995, pp. 19–25.

[30] ——, “STREAM: Sustainable Memory Bandwidth in High Performance Computers,” University of Virginia, Charlottesville, Virginia, Tech. Rep., 2007.

[31] A. Tridgell, “The dbench benchmark,” URL: `http://samba.org/ftp/tridge/dbench/README`, 2007, Retrieved: February, 2016.

[32] W. Norcott and D. Capps, “IOzone Filesystem Benchmark,” URL: `www.iozone.org`, Retrieved: February, 2016.

[33] Z. A. Mann, “Allocation of Virtual Machines in Cloud Data Centers&Mdash;A Survey of Problem Models and Optimization Algorithms,” ACM Comput. Surv., vol. 48, no. 1, Aug. 2015, pp. 11:1–11:34.

 ISBN: 978-1-61208-460-2

Data Locality via Coordinated Caching for Distributed Processing

Max Fischer and Eileen Kuehn
Karlsruhe Institute of Technology
76344 Karlsruhe, Germany
Email: {max.fischer, eileen.kuehn}@kit.edu

***Abstract*—Modern data analysis methods often rely on data locality. Processing applications are executed directly where data is stored. Frameworks enabling this require their own specialised environment and modifications. We propose an alternative approach using coordinated caching integrated into classic batch systems. A custom middleware layer provides relevant data locally on worker nodes. Most importantly no modifications are needed to add data locality to existing workflows. However, considerably more factors must be addressed by distributed caches compared to isolated ones. We investigated our approach with theoretic modelling, simulations, and a prototype implementation. Our evaluations show promising results for both applicability and performance.**

Keywords–Cooperative caching; Coordinated caching; Distributed caching; Batch production systems; Distributed processing.

I. INTRODUCTION

Caching as an enabler for data locality is an important topic for distributed data processing. As workflows usually process only a fraction of data frequently. It is thus inefficient to provide all data locally. In addition, changing workflow requirements in batch systems require a dynamic but coordinated caching approach.

Motivated by this, our approach enables data locality for batch systems with minimal requirements. The basis for our efforts are batch clusters which read data via network from dedicated fileservers. To make remote data available locally, a series of caches on worker nodes can be used. We propose a coordination layer that combines individual caches into a single pool.

Our approach stands out from existing caches by its scope and subject: for one, individual caches work on the scale of single machines. We target the entire batch system as a single entity. Also, distributed caches commonly target resource providers, e.g. web caches. In contrast, our approach targets the batch system as a resource consumer.

In this paper, we exemplarily consider the High Energy Physics (HEP) data analysis workflows of groups at the Karlsruhe Institute of Technology [1]. In general, HEP experiments of the Large Hadron Collider [2] are amongst the largest producers of scientific data in terms of data volume. Handling this data is performed in iterative workflows at different scopes. This ranges from reconstruction of raw data at a global scope [3] to the analysis of subsets of data at the scope of university groups.

In theory, HEP analyses conform to principles of data locality based processing. Analysis workflows execute multiple instances of an analysis application in a distributed environment. Each instance extracts the same set of variables from its share of an input data set. All sets of extracted variables are then merged. This corresponds to the map reduce method employed by frameworks such as Hadoop.

However, the wide range of HEP workflows and number of collaborating scientists dictate the use of established applications and frameworks. This also implies constraints, which are not satisfied by modern analysis frameworks. For example, HEP data is commonly stored in the ROOT binary file format. This format cannot be easily split or read from a stream, as is common in the Hadoop framework. The largest volume of data is used infrequently for validation and crosschecking.

We have performed modelling (Section II) and simulations of the situation (Section III). The estimates on architecture and scale are motivated by our own working experiences with the HEP analysis groups. From this, we conclude that classic batch processing can be considerably improved with cache based data locality. Our approach for data locality is based on a concept for coordinated caching (Section IV). To test our conclusions, we implemented a prototype of a caching middleware for batch systems (Section V). First experiences show promising improvements regarding performance and throughput (Section VI).

A. Related Work

Many individual sub-topics of our work have been focus of past research. In general, the approaches show considerable advantages over naive caching. However, no existing work matches the scope and applicability that is needed for HEP workflows.

Distributed caching has been studied extensively. For example, simulations on distributed caching with centralised control [4] show considerable improvements in hit ratio and throughput compared to independent caches. However, research usually focuses on the perspective of data providers, not consumers. Usage metadata is thus not taken into account, limiting the granularity required for data locality.

The CernVM File System (CVMFS) uses independent caches for software provisioning [5]. It is used to make shared software frameworks available in grid and cloud environments. The prototypical CacheD service of the HTCondor batch system caches binaries executed by jobs [6]. This minimises traffic, considerably speeding up deployment of multiple jobs on low-throughput networks. Both of these approaches show that caching is beneficial for batch processing. It improves throughput and allows deployment on resources without high

 ISBN: 978-1-61208-460-2

throughput network. However, these approaches target only items which are the same for all jobs.

User communities have made attempts to introduce data locality and/or caching to their workflows. On the one hand, the ATLAS collaboration demonstrated analysis speedup by using a central SSD cache for network attached HDD storage [7]. In this setup, data locality is improved but the cache is still accessed remotely.

On the other hand, attempts have been made to deploy HEP workflows on Hadoop [8], [9]. This provides advanced performance and scalability. However, extensive compromises have to be made. For example, applications have to be adapted to the I/O model and jobs may access only single files. The automatic distribution of data by the infrastructure is severely limited to ensure data integrity.

II. Modelling Distributed Caching

The modelling of distributed caches involves more aspects than using local caches on a single host. It therefore increases the complexity. Though the data to cache remain the same, factors such as location, replication, or even splitting, and grouping need to be considered.

We model caching statistically to estimate the probability of cache hits, following *cache hit rate*. Using a factorised approach, we describe issues individually and combine them to a single probability. This allows to assess a detailed cost of naively applying caching to a distributed system.

As shown in Figure 1, a distributed cache can be fundamentally viewed as a single entity. This is an ideal case, where a fraction of the global data volume is replicated on the cache volume. However, each worker node (WN) is actually limited to its local scope. In addition, jobs in a batch system are scheduled without knowledge about cache content. In the worst case, data and job placement may be perpendicular.

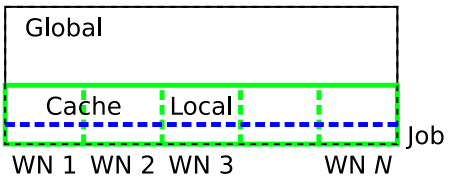


Figure 1. Scopes of distributed caching

To simplify the model, we make two assumptions: First, the volume of individual caches is large compared to items that are cached. Second, the volume of all items is large compared to both the individual and total cache volume. This allows us to treat volumes as continuous, avoiding quantisation and fringe effects. These assumptions mainly simplify the mathematical description. The general conclusions are not changed by this.

A. A Priori Hit Rate

As with regular caches, we employ a general probability that an item is cached. The choice of this is arbitrary and mainly serves for expressiveness. Without loss of generality, we assume a naive caching approach: the cache is filled with an equal fraction of all data. Thus the probability of an item being cached $P^{\text{base}}_{\text{cache}}$ is the fraction of cache volume V_{cache} and data volume V_{data}.

$$P^{\text{base}}_{\text{cache}} \propto \frac{V_{\text{cache}}}{V_{\text{data}}} \tag{1}$$

B. Item Locality

A distributed cache has no single cache volume but actually several separate ones. Cached items can only be accessed efficiently on the host they are located on. We therefore introduce a second probability for item locality, namely that an item is cached on the host it is accessed from, the *local hit rate* $P^{\text{item}}_{\text{local}}$.

If the batch system is not aware of the contents of individual caches, job scheduling is random in relation to item placement. The probability of executing on the correct host is inversely proportional to the number of hosts N_{hosts}. In contrast, creating a number of replicas N_{replica} on multiple hosts automatically increases the local hit rate. However, replicas reduce the effective total cache volume and thus the overall cache hit rate. Both hit rates combine to the expected hit rate $P^{\text{item}}_{\text{expect}}$.

$$P^{\text{item}}_{\text{local}} \propto \frac{N_{\text{replica}}}{N_{\text{hosts}}} \qquad N_{\text{replica}} \leq N_{\text{hosts}} \tag{2a}$$

$$P^{\text{item}}_{\text{cache}} \propto \frac{1}{N_{\text{replica}}} \tag{2b}$$

$$P^{\text{item}}_{\text{expect}} = P^{\text{item}}_{\text{cache}} \cdot P^{\text{item}}_{\text{local}} \propto \frac{1}{N_{\text{hosts}}} \tag{2c}$$

For naive caching of individual items and no aligned scheduling, we thus expect neither positive nor negative effect from replication. There is no naive approach to mitigate the penalty from distribution over multiple hosts.

In contrast, if we actively align job scheduling and cache locality, the local hit rate becomes a constant – the effectiveness of scheduling. In this case, the positive effect of replication is reduced and ideally eliminated with perfect scheduling. The negative effect of reducing the effective cache volume remains, however. Therefore, the penalty from distributed execution can be mitigated by active scheduling and avoiding replication.

C. Item Grouping

For efficiency, it is common for any single batch job to process groups of files. In classic batch systems, this grouping is done externally on job submission.

The benefit for each job is proportional to the number of processed items cached locally. We can express the expected efficiency $E^{\text{job}}_{\text{local}}$ as the fraction of expected local files $\left\langle N^{\text{local}}_{\text{items}} \right\rangle$ to total processed files N_{items}. This resolves directly to the local hit rate without aligned scheduling.

$$E^{\text{job}}_{\text{local}} = \left\langle P^{\text{job}}_{\text{local}} \right\rangle \propto \frac{\left\langle N^{\text{local}}_{\text{items}} \right\rangle}{N_{\text{items}}} = \frac{1}{N_{\text{items}}} \sum_{\text{items}} P^{\text{item}}_{\text{local}} \tag{3a}$$

$$= P^{\text{item}}_{\text{local}} \tag{3b}$$

For naive caching with unrelated items, this result is trivial. There are implications if aligned job scheduling is attempted,

however. Even when directing jobs to cached items, the fraction of locally available items limits the achievable efficiency. In this context, aligned job scheduling and replication are only useful if cache content is aligned as well.

D. Access Concurrency

For the scope of our work, a common caching assumption does not hold true: cache access is *not* always superior to remote access. For example, modern SSDs provide throughput at the same order as a $10\,\mathrm{Gbit/s}$ network. The difference is concurrency of accesses: caches are accessed only by local processes, whereas remote data sources can be accessed by all processes.

Modelling accesses for a distributed system in general is beyond the scope of this paper. However, the fact that both local and shared data sources have limited throughput allow for some basic statements. Ideally, both local and shared resources are used to their limit. This automatically implies that perfect cache hit rates are not desirable: a fraction of accesses should always make use of the shared resources available.

III. Estimates and Simulation

To estimate the benefit of coordinated caching, we have simulated multiple circumstances. The scope and estimates correspond to that of our associated HEP analysis groups.

We model the workflow as a set of application instances, each reading the same amount of data. We have benchmarked common HEP analysis applications to assess realistic limitations. The measurements show a maximum throughput of $20\,\mathrm{MB/s}$ for each instance. This is a technical limit induced by the software frameworks and data formats in use. The total input volume is estimated as $640\,\mathrm{GB}$. This corresponds to the volume of a 2 months data taking period.

For the processing environment, we model our current analysis infrastructure: a set of worker nodes connected to a set of fileservers via a dedicated network, as shown in Figure 2. We assume infinite bandwidth for the fileservers, but model the shared network with its limited throughput of $10\,\mathrm{Gbit/s}$. For processing, we model a number of worker nodes of equal configuration: 32 execution slots, a $4\,\mathrm{Gbit/s}$ SSD cache and a $10\,\mathrm{Gbit/s}$ network connection to the fileservers. For simplicity, we assume that as many instances as execution slots are deployed.

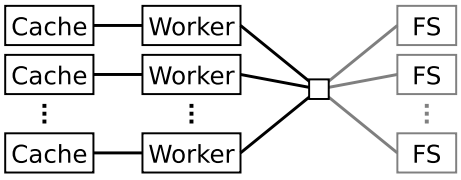


Figure 2. Simulated infrastructure

The results of the simulation can be seen in Figure 3. To rate performance, we have chosen the total processing time. For this observable, lower values are desirable. The cache hit rate to the local cache is used as a free parameter in the simulation. This is motivated by assuming adequate selection of cache content by existing algorithms. Thus, the hit rate is only determined by the scheduling of data and

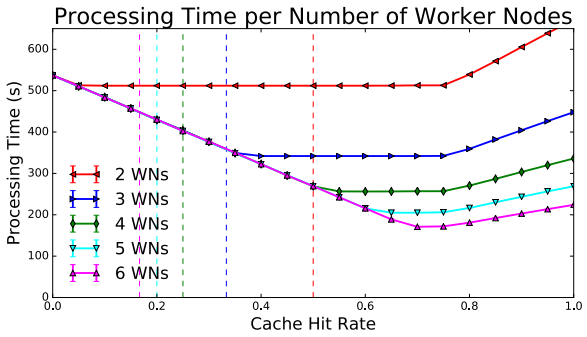


Figure 3. Simulation of workflow runtime

jobs. Dashed, vertical lines indicate the expected local hit rate without coordination.

There are two major results from our simulations (see Figure 3): on the one hand, caching enables to scale the infrastructure horizontally. This is especially important since remote access is a bottleneck even with few worker nodes. On the other hand, perfect cache hit rate is actually not desirable. Instead, there is a range of cache hit rates that give best performance. This range is defined by the points where either local or remote throughput is maximized.

Notably, even for perfect caching the expected local hit rate $P^{\mathrm{item}}_{\mathrm{expect}}$ is in this range only for the smallest setup. Therefore, we must coordinate caches to benefit from data locality.

IV. Concept for Coordinated Caching

Our approach for coordinated caching distinguishes between local and global view as known from batch systems. The local view refers to the individual worker nodes where batch jobs are running. The global view is given by the managing layer where queuing and scheduling of batch jobs is performed.

Based on this scheme, the functionality of a cache can be split into three layers, illustrated in Figure 4:

- the data provisioning on local scale,
- the data access on global as well as local scale, and
- the caching algorithm and distribution logic on global scale.

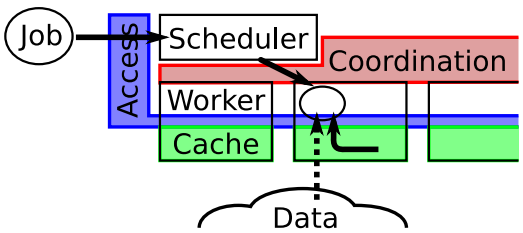


Figure 4. Layers of the distributed cache concept

A. Provisioning Layer

The provisioning layer handles the actual data on worker nodes. The worker nodes are the only elements with guaranteed

access to remote source data. They are therefore the logical place to retrieve and store both actual data and its metadata.

The provisioning layer implements the data provisioning associated with caches: remote items are copied to local cache devices, periodically validated, and potentially removed as needed. In addition, metadata of items, such as size and creation time, can only be collected on the worker node.

Since cache content is by design volatile, the layer also handles content metadata. This ensures that content metadata is as volatile as the actual content: the provisioning layer is the first to notice changes and invalidation of items. In the context of opportunistic resources, worker nodes shutting down neatly remove their allocation information as well.

Every worker node of the provisioning layer is limited to its local scope. Thus, major decisions are outsourced to the global scale. Every worker node exposes information required for cache decisions. The information are composed of its own metadata, such as size of caches, and metadata of items. In return, worker nodes rely on being notified of new items to cache and their relevance.

B. Coordination Layer

The coordination layer is responsible for decision making. Since it is not co-located with any active component, there are no restrictions on resource usage. It can thus fetch, aggregate, and process metadata without impacting other components.

The most important task of the layer is the selection of items to cache. This is fundamentally the same as in other caches. For example, scores can be calculated from access times. Given that arbitrary processing power can be provided, algorithms may be more complex than common. Both metadata and processing resources can be extended as needed.

The additional responsibility of the coordination layer is item placement. This means the deliberate assignment of items to elements of the provision layer. Common features of distributed systems, such as load balancing, must be taken into account. In addition, item relations must be considered for this. For example, common input groups should be assigned together.

C. Access Layer

The benefit of caches depends on low overhead for accessing items. Obviously, executed applications require local access to cached items. On the global scale, the batch system must receive information on item location.

Implementing access to cached items for executed applications on worker nodes depends on the actual setup of batch jobs. Primarily, the access protocol is the deciding factor. In principle, local redirection to cached items is sufficient.

Similarly, details of integrating with a batch system are setup specific. In any case, information on item location must be provided to the batch system. This is the responsibility of thin front ends, which forward information from coordination and cache layer. Functionally, this is similar to lookup services in distributed file systems, e.g. the NameNode of Hadoop FS. However, since all information originates in coordination and cache layer, our front ends are expendable.

V. HTDA Middleware Prototype

We have implemented a prototype of the coordinated caching concept: the High Throughput Data Analysis (HTDA) middleware [10]. The test environment and community are the HEP analysis groups of our university. We have deliberately kept dependencies on HEP software small, however. The prototype supports the HTCondor batch system [11] and applications performing POSIX I/O.

Architecturally, we build on a general purpose framework for a pool of worker nodes. Every instance of our application represents an individual node. The nodes are loosely coupled. They form a pool using stateless communication. Every node hosts component plugins, which collaborate in the abstracted pool. The component plugins implement the actual layers described in the previous chapter.

We currently use three node types that directly correspond to each layer:

- Provisioning nodes represent the provisioning layer. They stage, maintain, and validate cached files on worker nodes.
- Coordinator nodes represent the coordination layer. They aggregate metadata, and select and assign files for caching.
- Locator nodes provide locality information for the access layer. They act as proxies to the provisioning nodes.

In addition, we enforce access with hooks in the batch system and a redirection layer in each worker node's virtual file system.

A. Coordination of Provisioning Nodes

Coordination of provisioning nodes is based on the scoring mechanism of files. We use a 1-dimensional score to express the relevance for caching. The score is simple to transmit, unambiguous to interpret, and simplifies many algorithms with clear break conditions.

The file score caries an implicit command: files rated higher are assigned with higher priority to provisioning nodes. If space is required for new files, existing files with low scores are discarded first. However, relations between files are not apparent from the score. They are purely handled at the coordination level.

The score of files is calculated inside the coordinator node. A factorised approach is used to express multiple perspectives. Usage prediction is performed using a Least Recently/Frequently Used (LRFU) algorithm [12]. The algorithm is staged as a plugin. Other scoring algorithms can be used if needed. Additionally, we exemplarily model adequacy for caching: exceptionally small and large files are penalised to ease allocation and overhead.

Assignment of files to provisioning nodes is performed separately. Files accessed by the same job are grouped together. These groups are spread evenly amongst all provisioning nodes. Since files may belong to multiple groups, approaches such as distributed hash tables are not adequate. Instead, we exploit that grouping is deterministic and certain groupings are more likely to occur. This allows us to use a heuristic approach, which has proven itself adequate for our target community.

 ISBN: 978-1-61208-460-2

Groups scored highest are allocated first. If any files of a group are already allocated, the node with the highest file count is chosen. Otherwise, a node is chosen at random. This is performed until all files are allocated or no free space is left. Finally, the allocated files and their scores are pushed to each provisioning node individually.

Provisioning nodes only operate on the limited set of assigned files. Since they perform the actual data access, they are the final authority for staging files. This requires limited autonomy for internal allocation and rearranging of files. Thus provisioning nodes expose their allocation to the coordination layer. This allows to iteratively adjust local allocation and global distribution.

B. Application Access

A major motivation for our development is to preserve existing workflows. Therefore, all accesses must be implemented in a transparent way. To be generally applicable, we must also avoid dependencies on HEP specific software.

The HTCondor batch system natively allows to insert hooks into batch job handling. On batch job submission and finalisation, various job metadata is extracted: this includes input files, resource usage, owner and workflow identifiers. During handling of the batch job, job requirements are updated to prefer hosts caching required input files. This entire process is completely transparent to users.

We exploit HTCondor's rank based scheduling. Users can provide rating functions for worker nodes, e.g. to prefer faster machines. We add our own rating, based on file locality. Thus, locality is preferred in general but can still be overruled by users if hosts are not interchangeable. In addition, we use dynamic requirements to avoid jobs waiting indefinitely for a perfect host. We require worker nodes to satisfy a specific locality rating, which decreases over time.

Application file access on worker nodes is performed using POSIX system calls. Therefore, accesses must be redirected in the virtual file system layer of the operating system. We use a union file system, specifically AUFS [13], to squash cache file systems on top of network file systems. This redirects read access preferably to caches, with write-through to the network storage.

VI. Benchmarking and Experiences

We made overall positive experiences with our concept and prototype. Our test deployment is in operation since 6 months. The integration into infrastructure and workflows is fully transparent. Applications subject to caching show notable improvements in runtime.

Our test cluster features four worker nodes (see Table I). Each has 32 logical cores, $512\,\mathrm{GB}$ SSD cache and a $10\,\mathrm{Gbit/s}$ network interface. A total of 7 fileservers is available on each worker node. The global services for the middleware and batch system are on separate machines.

TABLE I. TEST CLUSTER WORKER NODE

OS		Scientific Linux 6 (Kernel 2.6.32)
CPU	2x	Intel Xeon E5-2650v2 @ 2.66 GHz (8 cores, 16 threads)
Memory	8x	8GB RAM
SSD	1x	Samsung SSD 840 PRO 512 GB or
	2x	Samsung SSD 840 EVO 256 GB
HDD	4x	WDC WD4000 4 TB
Network	1x	Intel X540-T1 (10GigE/RJ45)

A. Middleware and Infrastructure

The overhead from our middleware is well acceptable. On worker nodes, the software consumes $(20 \pm 5)\,\%$ of a single CPU. We consider this to be a reliable worst case estimate: first, the frequency of file validation and allocation is very high for testing purposes. This is the majority of actions performed on worker nodes. A factor 2 to 10 less is reasonable for production. Second, our prototype is written in python 2.7 and interpreted with CPython. An optimised implementation in a compiled language is guaranteed to be faster in production. We also investigate other interpreters, e.g. pypy, which provide 3 to 5 times better efficiency in our tests.

The metadata is negligible compared to the actual data. A $500\,\mathrm{GB}$ cache with 7000 cached files has $(3.0 \pm 0.5)\,\mathrm{MB}$ of persistent metadata. The memory footprint of the application is of the same magnitude. The metadata aggregated in the coordination layer is on the order of $250\,\mathrm{MB}$. Communication overhead between nodes is unnoticeable compared to data transfers.

The experience with the access layer using AUFS is mixed. There is virtually no overhead on reading performance compared to direct access. Even during concurrent accesses the full cache device performance is available. However, we have repeatedly observed spontaneous performance degradation and crashes of the union mount service.

We assume the deficiencies to be caused by the old kernel of the Scientific Linux 6 [14] operating system required by our target community. To validate this assumption, we performed tests on the same hardware using CentOS 7 [15] with kernel 4.4.1. The tests revealed none of the deficiencies experienced with Scientific Linux 6. Still, a redirection layer tailored to our access pattern may be more suitable for production deployment.

B. Integration and Performance

Our prototype operates transparently to users. Applications executed on our worker nodes require no changes. Workflows need to be adapted in one single point: required input files have to be reported explicitly to the batch system to benefit from caching. Since users delegate job submission to tools, we are able to automatise this.

The number of metrics which can be used to assess performance are numerous. A simple and illustrative method is to track individual reference workflows. These are regular end user analyses, extended to collect lightweight performance statistics. Figure 5 shows the distribution of execution times of individual jobs for one workflow. Lower execution times are better. The blue area represents jobs run with the cache disabled. The green area represents jobs run with the cache enabled.

Considerable improvements in execution time indicate appropriate data provisioning and batch job scheduling. Indeed we observe consistently improved execution times when our HTDA middleware is enabled. Highly data driven workflows have been sped up by a factor of 4. Workflows without cached

 ISBN: 978-1-61208-460-2

data also benefit from this, as cluster and network utilisation is lower.

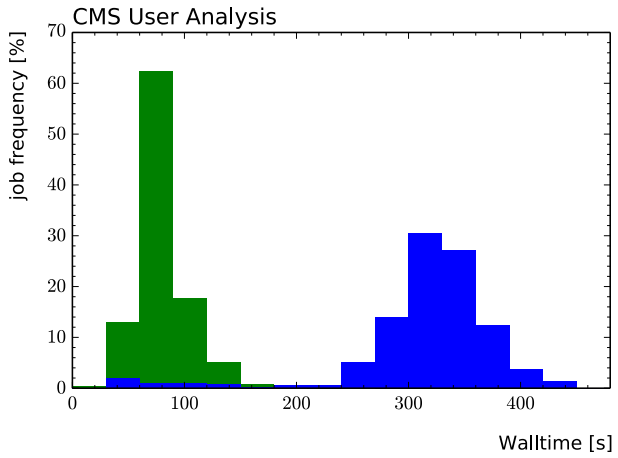


Figure 5. Performance of reference analysis

VII. Conclusion and Outlook

Modern science is able to collect vast amounts of data. Analysing the increasing volumes is a challenge in itself, however. Solutions exist but are not necessarily applicable. We have therefore investigated an alternative to transparently provide data locality, specifically in batch systems.

We propose a dedicated cache on the scope of an entire batch system. Modelling shows that using individual caches on worker nodes is not efficient. Simulations reveal that high cache hit rates do allow for improved throughput. This is only feasible when actively coordinating individual caches.

Our approach to coordinated caches uses three layers. The provisioning layer is composed of agents on worker nodes, handling data directly. The coordination layer acts on a global scale, coordinating caches based on available metadata. The access layer wraps around workflows on both scales to provide an interface to our system.

To test our estimations and concept, we implemented a prototype of the proposed system – the HTDA middleware. The current implementation handles several aspects of our considerations. This system has already proven to speed up data analysis by a notable factor.

The concept allows room for further research. As our simulations show, perfect hit rate is not desirable. This may be considered in the selection or distribution algorithm to increase effective cache volume. Compared to other cache solutions, our scope is at magnitudes more of processing resources and magnitudes less of turnaround time. Data selection algorithms may therefore be drastically expanded. Furthermore, arbitrary sources for local and external metadata can be considered.

Acknowledgment

The authors would like to thank all people and institutions involved in the project Large Scale Data Management and Analysis (LSDMA), as well as the German Helmholtz Association, and the Karlsruhe School of Elementary Particle and Astroparticle Physics (KSETA) for supporting and funding the work.

References

[1] J. Berger, F. Colombo, R. Friese, D. Haitz, T. Hauth, T. Müller, G. Quast, and G. Sieber, “ARTUS - A Framework for Event-based Data Analysis in High Energy Physics,” ArXiv e-prints, Nov. 2015.

[2] The Large Hadron Collider. [Online]. Available: http://home.cern/topics/large-hadron-collider [retrieved: Nov 12, 2015]

[3] The Worldwide LHC Computing Grid. [Online]. Available: http://wlcg-public.web.cern.ch [retrieved: Nov 13, 2015]

[4] S. Paul and Z. Fei, “Distributed caching with centralized control,” Computer Communications, vol. 24, no. 2, 2001, pp. 256 – 268. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366400003224

[5] J. Blomer and T. Fuhrmann, “A fully decentralized file system cache for the cernvm-fs,” in Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on, Aug 2010, pp. 1–6.

[6] D. Weitzel, B. Bockelman, and D. Swanson, “Distributed caching using the htcondor cached,” in Proceedings for Conference on Parallel and Distributed Processing Techniques and Applications, 2015. [Online]. Available: http://stacks.iop.org/1742-6596/513/i=3/a=032054

[7] W. Yang, A. B. Hanushevsky, R. P. Mount, and the Atlas Collaboration, “Using solid state disk array as a cache for lhc atlas data analysis,” Journal of Physics: Conference Series, vol. 513, no. 4, 2014, p. 042035. [Online]. Available: http://stacks.iop.org/1742-6596/513/i=4/a=042035

[8] S. A. Russo, M. Pinamonti, and M. Cobal, “Running a typical root hep analysis on hadoop mapreduce,” Journal of Physics: Conference Series, vol. 513, no. 3, 2014, p. 032080. [Online]. Available: http://stacks.iop.org/1742-6596/513/i=3/a=032080

[9] S. Lehrack, G. Duckeck, and J. Ebke, “Evaluation of apache hadoop for parallel data analysis with root,” Journal of Physics: Conference Series, vol. 513, no. 3, 2014, p. 032054. [Online]. Available: http://stacks.iop.org/1742-6596/513/i=3/a=032054

[10] M. Fischer, C. Metzlaff, and M. Giffels. HPDA middleware repository. [Online]. Available: https://bitbucket.org/kitcmscomputing/hpda [retrieved: Nov 12, 2015]

[11] D. Thain, T. Tannenbaum, and M. Livny, “Distributed computing in practice: The condor experience,” Concurrency and Computation: Practice and Experience, vol. 17, 2005, pp. 2–4.

[12] D. Lee, J. Choi, J.-H. Kim, S. Noh, S. L. Min, Y. Cho, and C. S. Kim, “LRFU: a spectrum of policies that subsumes the least recently used and least frequently used policies,” Computers, IEEE Transactions on, vol. 50, no. 12, Dec 2001, pp. 1352–1361.

[13] J. R. Okajima. AUFS project homepage. [Online]. Available: http://aufs.sourceforge.net [retrieved: Nov 12, 2015]

[14] S. Linux. Scientific linux. [Online]. Available: {https://www.scientificlinux.org} [retrieved: Feb 10, 2016]

[15] C. Project. Centos project. [Online]. Available: {https://www.centos.org} [retrieved: Feb 10, 2016]

 ISBN: 978-1-61208-460-2

Enhancing Cloud Security and Privacy: The Cloud Audit Problem

Bob Duncan
Computer Science
University of Aberdeen
Aberdeen, UK
Email: bobduncan@abdn.ac.uk

Mark Whittington
Accounting and Finance
University of Aberdeen
Aberdeen, UK
Email: mark.whittington@abdn.ac.uk

***Abstract*—Many people assume that cloud audit is no more difficult than IT audit in general. We provide an outline of the evolution of cloud, providing an explanation of how it differs from conventional IT. We then discuss some of the benefits and drawbacks of cloud, particularly in connection to audit challenges, highlighting the dangers and shortcomings of many approaches.**

***Keywords*—security; privacy; standards; compliance; audit.**

I. INTRODUCTION

Cloud computing offers the possibility of a substantial economic benefit to firms and governments, yet at the same time, increases complexity and risk. This results in an interesting dilemma. On the one hand, potential cost savings of 50 - 90% [1] are possible, which is highly attractive, but on the other hand, complexity can increase exponentially, placing significant increasing risk on business and government alike.

In previous work [2] on enhancing cloud security and privacy, we addressed issues of the cloud service provider's (CSP) lack of accountability in the standard service level agreement (SLA). We mentioned the importance of the role assurance plays, and the two main mechanisms used to achieve this, namely compliance and audit. In this paper, we will address some of the issues relating to audit. In order to understand how the use of cloud impacts on the audit process, and how it differs from conventional IT audit, we need to first understand what audit is, why we need to do it, who should be doing it and how it should be done. We must also understand what special difficulties the use of cloud brings to audit. We therefore revisit our definition of audit.

Audit (OED [3]: "To make an official systematic examination of (accounts), so as to ascertain their accuracy") requires outsiders who are deemed to be both objective and expert to form their own opinion of what is being audited and then to publicly state their confidence (or otherwise) in the reliability of what they have investigated. Auditing is not straightforward or easy. Just as with accounting auditors, objectivity is difficult when companies pay auditors directly and auditors would also like to be retained for the following year. Audit is also potentially very expensive if done well by the best experts in the field and there is a temptation to reduce the experts' role to one of advising, often writing checklists to be administered by qualified technicians.

We start by considering the purpose of audit, who should be carrying it out, and how it should be done, which we address in Section II. The remainder of the paper is organised as follows: Looking at past corporate computing models, Section III provides us with an understanding of how corporate computing has evolved over recent years, how these stages of evolution have developed, and how they compare and impact on cloud computing. In Section IV, we look at how audit is currently performed. In Section V, we question whether there are any weaknesses in this approach and; in Section VI, we touch on some of the cloud security compliance issues. Section VII, considers how to tackle these weaknesses; and finally, in Section VIII, we discuss our conclusions.

II. THE PURPOSE OF AUDIT

We consider three main purposes of audit, the most widely understood of which is the statutory requirement for financial statements to be audited by an independent external auditor, which has been a cornerstone of confidence in global financial systems since auditing was introduced. It provides assurance that company managers have presented a "true and fair" view of a company's financial performance and position, underpinning the trust and obligation of stewardship between company management and the owners of the company, the shareholders.

A second purpose of audit is IT systems audit. Traditional audit approaches often involved treating IT systems as "black box" systems, meaning trust was placed in the IT systems, and looking at the functioning of the IT system was not considered part of the statutory audit. The obvious shortcoming of this approach was addressed by conducting a specific IT based audit of the IT systems, to ensure these systems performed exactly as expected. These audits are usually conducted by IT specialists, often in conjunction with accounting audit professionals to ensure the functioning of these systems are properly understood. However, these are not mandated under statute, which presents a weakness. In addition, there is no requirement for an annual audit to be undertaken.

A third purpose of audit is compliance, either with regulations, or more often with standards. This is often undertaken to assure shareholders and other stakeholders that the company is using best practice in its operations. This is particularly the case in cloud computing, where systems are operated by third parties beyond the control of the cloud user. Currently, the difficulties associated with performing an adequate cloud audit present one of the key barriers to cloud adoption [33]. These audits are not mandated under statute, which presents a weakness, and there is no requirement for an annual audit to be undertaken.

Statutory audit is an area which is well understood and which benefits from over a century of research and experience. Despite this, there remain differences of opinion and a number of problems are yet to be resolved. Duncan and Whittington provide some useful background in [5]. One of the main issues concerns the independence of the auditor. The auditor is meant to be independent, yet is paid by the firm they are auditing.

 ISBN: 978-1-61208-460-2

There may also be additional links between the auditor and the firm, such as other non-audit consulting work undertaken by the auditor. An audit firm is keen to remain auditor of the firm for a long period of time to ensure continuity of income and enhancement of profit. The firm is often keen not to change auditor too frequently, lest their reputation suffer damage by being unable to retain an auditor, as well as trying to keep costs to a reasonable level. Audit firms are keen to undertake non-audit consultancy work in order to further maximise revenue and profits. The firm is generally keen for this practice to take place, due to perceived cost savings to the firm. These arrangements can potentially create tensions, which in some cases might affect the impartiality of the auditor, hence some jurisdictions seek to limit consultancy by auditors. Despite issues, financial auditors are heavily regulated, audits are mandatory and must be carried out every year.

Some industries are regulated and often the regulator will assure themselves of compliance with regulations through the use of audit. In the UK, organisations such as The Office of Gas and Electricity Markets (Ofgem); The Office of Water Services (Ofwat); The Office of Telecommunications (Oftel); The Postal Services Commission (Postcomm); The Civil Aviation Authority (CAA); The Office of the Rail Regulator (ORR); The Office for the Regulation of Electricity and Gas in Northern Ireland (Ofreg); and The Office of Communications Ofcom) will often use reports given by the company's auditor. These will generally be carried out based on the requirements of the licence granted by the regulator. The Financial Conduct Authority (FCA) and the Prudential Regulation Authority (PRA) usually take this view, although requirements here are often more onerous. Some regulators, such as those responsible for the regulation of professional services, may conduct the audit using their own auditors, and frequency of audit is usually less regular than with financial accounting.

Cloud audit difficulties have long been seen as a potential barrier to cloud adoption [33] [6], and there is certainly a belief that trust and privacy issues [7] [8] [9] [10] also need to be borne in mind. A common theme is the recognition that cloud audit is far harder to perform than for non-cloud systems.

Audit may be required to test internal control systems, particularly where they involve financial reporting. This can extend to IT audit, where rather than treat the IT systems as black box components of the company systems, the IT systems themselves are audited to provide assurance that they are capable of delivering what is needed by the company. Audit may also be required where a company is involved in a joint venture project with another company or companies. The audit requirements will usually be built in to the terms of the joint venture agreement, specifying who will have what rights to conduct the audit. Audit will not necessarily be mandatory, nor will the auditors and audit process necessarily be regulated.

Another facet of audit is internal audit, where a company seeks to assure itself of how well its internal processes are running. Often this is continuous in nature, rather than sporadic. It is not mandatory and there is no regulation of the auditors or the audit process. In previous work with Pym [11], we developed a cloud assurance model which uses continuous internal audit to help achieve the required security goals. Audit can be used to test for fraud. Forensic audit is used if fraud is discovered, to find and collect suitable evidence for presentation in a court case, whether criminal or civil.

While auditing in the accountancy world has enjoyed the benefit of over a century of practice and experience, cloud computing audit can not be considered a mature field, and there will be some way to go before it can catch up with the reflection and rigour of work done in the accounting profession. An obvious area of weakness arises when taking audit professionals from the accounting world out of their comfort zone, and placing them in a more technical field. Whilst the use of people with a computing background can overcome some of these issues, their lack of audit background presents another weakness. Clearly further research will be needed in this area.

Thus we see that there is more than one purpose for conducting audit. We can have: statutory audit, which might extend to audit of internal control over financial reporting and fraud audit; IT audit, which covers the audit of IT systems; and compliance audit, which will include regulatory compliance, standards compliance, joint venture compliance, internal audit and forensic audit. This list is not exhaustive. Of all the purposes of audit, statutory audit is the most rigorous and highly regulated, and for cloud, we could do well to learn from this wealth of experience and rigour. In today's world, information can be just as valuable as money. The impact of compromise, leakage, or theft of information can have a catastrophic impact on cloud users, thus it makes sense to consider applying equal rigour to the protection of information.

III. How did Cloud Computing Evolve?

Corporates have long understood the potential benefits to be gained from embracing information technology ever since the early days of computing, when expensive mainframes were the only option — open only to the largest corporates. Since those days, modern information systems have evolved considerably, leading to the development of complex, highly distributed information systems and the need to police them properly. The need to address traditional security issues of confidentiality, integrity and availability (CIA) has increased this complexity further, due to the need for scalability and redundancy. We have seen a relentless explosion in performance, cost reductions and wider accessibility for more and more corporates. Massive capital and operating costs no longer present the barrier they once did. Technology has brought about major change in operational efficiency. The invention of the internet has provided new opportunities and increased exposure to new markets, yet at the same time, threats to security and privacy have increased at a frightening rate. The following list highlights nine evolutions of corporate computing, with a brief explanation on each:

- Distributed Systems
- Business Process Management
- Service Oriented Architecture
- Grid Computing
- Utility Computing
- Virtualization
- Corporate Outsourcing
- Cloud Computing
- Economics of Cloud Computing

Distributed systems can be described as a software system in which components located on networked computers communicate and coordinate their actions by passing messages, in order to achieve a common goal. Early interest from military

 ISBN: 978-1-61208-460-2

and defence agencies has contributed greatly to the benefits to industry [12]. Early research effort from industry [13] [14] has also been evident. This is still an active research area today, with Lenk and Tai [15] addressing disaster recovery, Orgerie et al. [16] seeking to reduce energy costs of distributed systems, and Gottinger [17] who addresses the management issues of improving economic mechanism design of distributed systems.

Business process management (BPM) is a subset of operations management, which focuses on improving corporate performance by managing and optimising a company's business processes. Information technology (IT) can play an important role in helping with this continual process of improvement, in which three basic elements are involved — people, process and technology. This interaction between the three elements perfectly describes the business architecture of a company. Instead of adapting business processes to fit rigid and intractable software, software could now be developed to align with business practices. This allowed a better fit to the way a firm did business, ensuring greater efficiencies. A rich area of research for over three decades, from the early work of Zachman [18], Norman et al. [19] through to later work by Zhu et al. [20] and Herzberg et al. [21], it has attracted great interest from a wide range of disciplines. The opportunities offered by the development of business process architecture would lead to issues in trying to communicate with different computing systems. This lead to the development of the term "Service Oriented Architecture (SOA)" [22]. SOA was defined as *"a software architecture that starts with an interface definition and builds the entire application topology as a topology of interfaces, interface implementations and interface calls"*. SOA didn't get much traction until 2001 when web services technology became widely adopted. In many respects, web services gave SOA the foundation it needed to become widely accepted. Again a healthy research area, still very much active today, with Girbea et al. [23] addressing optimisation of industrial manufacturing processes and Picard et al. [24] presenting several alternative systems for enhancing collaboration at inter-organisational level.

Grid computing is a varied collection of computer resources spread over multiple locations designed to achieve a common goal. Grid computing differs from conventional high performance computing systems such as cluster computing. Grid computers have each node loosely coupled and highly distributed over a wide geographical area, whereas high performance cluster computing generally has all the nodes physically connected in the one location. Grid computers tend to be highly heterogeneous, whereas a cluster will usually comprise a set of identical hardware. Grid computing nodes can be owned by a diverse range of organisations who share access to these resources, whereas a cluster tends to be owned by a single organisation. Grid computing started to gain traction in the mid to late 1990s. Utility computing is a service provisioning model in which a service provider makes computing resources and infrastructure management available to the customer as needed, and charges them for specific usage rather than a flat rate. This model differs from grid computing as the service provision comes from a single service provider, rather than from a network of service providers. The model is not new, evolving during the 1960s and 1970s. To facilitate this business model, mainframe operating systems evolved to include process control facilities, security, and user metering. The model re-surfaced in the late 1990s with a number of large players offering their own flavour. The development of virtualisation software helped move the model towards cloud computing.

Virtualisation is the creation of a virtual (rather than actual) version of something, such as an operating system, a server, a storage device or network resources. It began in 1960s mainframe computers as a means of logically dividing system resources provided by mainframes between different applications, although its meaning has considerably broadened in scope since then. Virtualisation allows spare capacity in large server systems to be partitioned into self standing virtual servers, which can provide "Chinese walls" between instances to improve security where different customers use each of the virtual instances. By 2003, the process of virtualisation had become much more developed, yet few had offered resource isolation or performance guarantees; most provided only best-effort provisioning, risking denial of service. Large corporates have traditionally had a high focus on efficiency and maximising profits. Consequently, they have long explored the potential for cost savings to be had from outsourcing non core activities. Indeed, they have also used extended outsourcing techniques such as off-shoring which allow them to potentially reap far greater cost savings than with normal outsourcing methods. In the early days of mainframe computing, with no internet to worry about, security was much less of an issue. However, this would radically change with the arrival of the internet.

Cloud computing offers the possibility of a substantial economic benefit to firms, yet at the same time, increases complexity and risk further. This results in an interesting dilemma. On the one hand, potential cost savings of 50 - 90% [1] are possible, which is highly attractive, but on the other hand, complexity can increase exponentially, placing significant increasing risk on business and government alike.

It will be useful to understand how the economics of cloud computing has helped it to achieve such rapid market penetration and deployment. While the incentives to use cloud are very attractive, it brings with it other issues, such as accountability, assurance, audit, availability, confidentiality, compliance, integrity, privacy, responsibility and security, all of which need to be properly addressed. Cloud computing is the most agile of these systems, yet the most technically challenging to secure, due to the multiplicity of relationships within the cloud ecosystem.

IV. How is Audit Currently Performed?

We provide a brief outline here of how the approach to financial audit evolved over the past century. Then, vouching was the common mechanism utilised to conduct an audit. Here, the auditor checked every single transaction in the company books to vouch its authenticity. This was extremely cumbersome, and expensive, to conduct, and in ignoring management control systems, proved to be very inefficient. As companies grew larger, this technique could no longer be supported.

A move to statistical sampling of transactions, together with consideration of the effectiveness of internal controls, allowed for a more efficient approach to the audit of larger companies. Sometimes, fraud audit would also be carried out to detect the possibility of fraud having occurred, whether from external or internal sources. The use of checklists became popular. Eventually companies started to use IT for their financial systems. At first, the IT systems were treated as black

 ISBN: 978-1-61208-460-2

box systems, where auditors merely considered how the input was transformed into the expected output. Ultimately, this led to the need to check the integrity of the systems themselves and these too, were audited, creating a need for auditors to broaden the scope of their learning. Over time, there was a move towards a more risk based approach, with a greater emphasis on performing due diligence, and less emphasis on the use of the checklist. Discussing financial audit in more detail is impossible within the constraints of this paper, but provides a brief outline of how things have changed.

IT audit is not the same as financial audit. Financial audit's purpose is to evaluate whether an organisation is adhering to standard accounting practices, and to ensure the financial statements present a true and fair view of the information contained in the financial reports. The purpose of an IT audit is to evaluate the system's internal control design and effectiveness, through studying and evaluating controls and testing their effectiveness. IT audit can be carried out by the company's internal audit department, an external agency, or by the company's own auditors. IT auditors are not regulated to the same extent as financial auditors. Many of the large auditing firms have set up specialised departments to handle IT audits, with the benefit that they have access to the financial audit expertise of the firm. Many IT audit firms do not have financial audit experience. In general, there is a greater move towards a checklist based approach in performing IT audits, as with compliance audit, including security standards audit, which traditionally use the checklist approach.

Joint venture audit is conducted in accordance with the terms of the joint venture agreement, often by the internal audit department of the partners, although some will use their external auditors for this. Internal audit is sometimes performed by employees with limited experience of how external financial audit is conducted, resulting in a less risk based approach. This can be useful in that their work can better inform the external auditors when they arrive to carry out their audit. Forensic audit will usually be carried out in response to the discovery of a systems breach, usually by forensic IT specialists.

V. Are there any weaknesses in this approach?

We address some of the cloud security standards issues in Section VI. The frequency of compliance auditing is generally quite relaxed, in that reassessment need take place only when system changes take place, or every few years, otherwise. This completely fails to grasp the rapidly evolving nature of security threats. There exists a clear need to employ some method of continuous monitoring when it comes to security management. Reports from global security companies, which do not differentiate between cloud and non-cloud using companies [25]–[27], suggest that over 85% of security breaches involve a low level of technical competence, facilitated instead by lack of understanding, lack of competence, or poor configuration of systems on the part of victims. It would be very useful to the research community if security breach reporting companies were to publish cloud specific data. While CSPs are very reluctant to publicise security breaches, last year's cloud breaches on iCloud, Target, Home Depot, Sony and the US Internal Revenue Service (IRS) may have more to do with slack security culture and poor internal control processes than cloud security weaknesses. Nonetheless, this clearly illustrates the importance of the link between people, process and technology.

Vouk [28] suggests a key element of SOA is an ability to audit processes, data and results, i.e. the ability to collect and use provenance information. Since SOA is not always included in what is run on the cloud, there is a possibility that this may present a weakness if steps are not taken to address this shortcoming. Both Leavitt [29] and Wang et al. [30] suggest the use of third party auditors, yet many CSPs to this day are reluctant to offer this service. Armbrust et al. [33] suggest lack of cloud audit-ability presents the number three barrier to cloud take-up, and that more needs to be done to ensure compliance with new legislation such as Sarbanes-Oxley Act (SOX) and the Health and Human Services Health Insurance Portability and Accountability Act (HIPAA) regulations.

Chen and Yoon [31] suggest that an exceptionally robust approach to cloud audit is required in order to ensure compliance with all necessary legislation, regulation and standards. Ramgovind et al. [32] suggest the question of whether the CSP is willing to undergo audit represents a key security issue for cloud use. Wang et al. [34] propose a publicly audited cloud environment to ensure proper privacy is maintained. Zhou et al. [35] in conducting a survey of CSPs and SLAs suggest availability, audit, confidentiality, control, and data integrity should be added to standard SLAs.

Grobauer et al. [36] suggest that without proper audit provisions in SLAs, security and privacy will be compromised. Doelitzscher et al. [37] propose a technical solution to this issue using Security Audit as a Service (SAaaS) in conjunction with software agents and a neural network to detect anomalies and misuse of systems. Early results are promising, although the system has yet to run live. Ruebsamen and Reich [38] propose the use of audit agents to patrol a cloud environment to ensure proper accountability. Lopez et al. [39] propose the use of Somewhat Homomorphic Encryption (SHE) and Public-Key Searchable Encryption (PEKS) in conjunction with audit agents to ensure proper accountability in the cloud.

The approaches to financial audit and IT audit are well understood, subject to our earlier comments, and are generally perceived as fit for purpose. But, when using cloud computing, everything changes. Instead of working on systems under the control of the company being audited, these systems belong to others, such as the CSP and any one of a number of other actors involved in the cloud ecosystem. In previous work [2] on enhancing cloud security and privacy, we drew attention to the shortcomings in the standard SLA offerings of many CSPs. Most lack any serious level of accountability, assurance, audit, confidentiality, compliance, integrity, privacy, responsibility or security, merely concentrating on availability as the only measure of performance provided. Many are reluctant to allow third party auditors into their premises, making effective audit difficult, if not impossible.

One of the fundamental benefits of cloud computing, agility, presents auditors with a very difficult challenge. Maintaining an effective audit trail presents a serious challenge when cloud instances are spooled up or down, sometimes by the thousand, as needed. We address this specific issue in [40].

VI. Cloud Security Standards Compliance Issues

Cloud security standards compliance presents a number of interesting issues. First, in today's global economy, an effective

 ISBN: 978-1-61208-460-2

standard needs to be internationally accepted, which introduces jurisdictional problems to the mix, coupled with the fact that the pace of evolution of new technology far outstrips the capability of international standards organisations to keep up with the changes [41]. Second, standards compliance is not mandatory, but merely a voluntary practice. While many large organizations are keen to be compliant in order to provide assurance to their clients, there is no legal obligation to do so. Third, compliance with standards is also generally not required by regulators, although there are signs that this attitude may be changing. Fourth, compliance procedures will usually allow a degree of latitude to the compliant company in respect of the level of compliance they wish to achieve. Fifth, the audit mechanisms used by compliance auditors can be flawed [42]. Sixth, compliance auditors are not heavily regulated, as they are for financial audit. Seventh, there are a great many cloud security standards organisations in existence, often with differing agendas, or merely concerning themselves with an area of narrow focus. Eighth, no complete cloud security standard yet exists. Nine, very few early standards took a risk based approach, relying instead on the checklist approach to compliance. Ten, knowing that a prospective company is compliant is not enough. It is necessary to understand exactly what level of compliance has been achieved, and this detailed level of information is seldom disclosed.

Will compliance with a standard ensure security? In [5] we argued that compliance with a cloud security standard is more likely to ensure compliance with a security standard, rather than achieve a meaningful level of security. We take a brief look at some of the larger standards organisations.

The International Standards Organization (ISO) have done excellent work on global standards, yet the benefit of this approach is also a weakness. Seeking agreement across the globe, prevents them from keeping standards fully up to date, taking up to eight years to publish a fully agreed new standard. It has taken some time, but it is encouraging that the ISO has changed approach on ISO 27000 series standards to a risk based approach, which started to filter through in 2014, and this is very welcome. A few cloud standards are also starting to filter through, but a full range of cloud standards is still some way off. It is also encouraging to note that as new standards are published, they are adopting a risk based approach.

The National Institute of Standards and Technology (NIST), who produced one of the earliest clear definitions of what cloud computing is, have long been of the view that a risk based approach to cloud security would be more effective. The US government finally accepted last year that this was a sensible approach [43], and NIST have developed an excellent risk based security standard. NIST produce excellent work, but compliance with NIST standards often only extends to US companies and those doing business with the US.

The Cloud Security Alliance (CSA) have been very active in promoting cloud security standards. Their work is good, but the weakness lies in the approach used for the method of achieving compliance.

AICPA have produced a number of standards, including for cloud. Their SOC2 standard for cloud has seen many CSPs attain compliance, including across the globe. However, this does not cover the case where the cloud service will potentially affect the statement of financial information, for which a SOC 1 will be required. Where assurance on trust is required, a SOC 3 report should be sought. While these standards apply the same criteria for both cloud and non-cloud situations, it would be naïve to believe that non-cloud security measures would be suitable for a cloud deployment.

Considerable work has been done on addressing legal issues with cloud deployment [44] [45] [46] [47] [48]. With the global reach of both cloud users and CSPs, this will help to tackle outstanding issues of sovereignty and jurisdiction.

VII. How do we tackle these weaknesses?

There needs to be a proper understanding of precisely why a cloud audit is being performed. Different types of audit require different approaches and it is important not to forget the fundamental rationale for an audit. No matter what type of audit is being carried out for whatever purpose, the auditor needs to keep the fundamental requirements of the audit firmly in mind throughout the audit.

The cloud security standards issue is a particularly difficult challenge. We address this challenge in future work and make some helpful suggestions here to address this problem.

The CSP problem with the standard SLA needs to be addressed as an area of added risk. It is important to realise just what this additional risk will mean to the company under review. While it can be said that the standard SLA can offer a better level of security than is available to the average small to medium sized enterprise (SME) [49], it cannot be considered foolproof. Companies, and their auditors, should recognise the weaknesses inherent in the standard SLA and address these specifically as an added risk to the company. We hope that, in time, the changes we seek in [2] will come to pass.

The audit trail issue requires companies and their auditors to recognise that the problem exists. In [40], we address this issue and make some useful suggestions on how to tackle these weaknesses.

VIII. Conclusion

Even centuries of experience of financial audit have not solved all the problems of conflict of interest, or the clear understanding and interpretation of the role of audit, hence the need to consider some of the more fundamental issues facing companies today. We must bear in mind that information is now as valuable to companies as money, and deserves serious thought and action to safeguard it.

We have looked at some of the challenges facing companies who seek to obtain good cloud security assurance through audit. We have seen how weaknesses in standard CSP SLAs can impact on cloud security. We have identified issues with cloud security standards, and how that might impact on cloud security. Achieving compliance with cloud security standards is a worthwhile goal, but success will only guarantee compliance with the standard, not necessarily a useful level of security. We have considered how the lack of accountability can impact on security. We have also considered how misconceptions in the purpose and scope of audit can also impact on security.

We have touched on how these difficult areas of security might be approached as part of a comprehensive security solution based on our proposed framework. Clearly, companies could benefit from further research in several of these areas. In particular, cloud audit could benefit from applying the rigour

 ISBN: 978-1-61208-460-2

used by auditors in the accounting world when carrying out statutory audit. However, we would caution that action is needed now, not several years down the line when research reaches a more complete level of success in these areas. The threat environment is too dangerous. Companies have to act now to try to close the door, otherwise it may be too late.

REFERENCES

[1] P. Mell and T. Grance, "Effectively and Securely Using the Cloud Computing Paradigm," Tech. Rep., 2009.

[2] B. Duncan and M. Whittington, "Enhancing Cloud Security and Privacy: Broadening the Service Level Agreement," in Trustcom/BigDataSE/ISPA, 2015 IEEE. Vol. 1. IEEE, 2015, pp. 1–6.

[3] OED, "Oxford English Dictionary," 1989. [Online]. Available: www.oed.com [Retrieved: Feb 2016].

[4] M. Armbrust et al., "A view of cloud computing,"Commun. ACM, vol. 53, no. 4, 2010, pp. 50–58.

[5] B. Duncan and M. Whittington, "Compliance with Standards, Assurance and Audit: Does this Equal Security?" in Proc. 7th Int. Conf. Secur. Inf. Networks. Glasgow: ACM, 2014, pp. 77–84.

[6] H. S. Herath and T. C. Herath, "IT security auditing: A performance evaluation decision model," Dec. Supp. Sys., vol. 57, 2014, pp. 54–63.

[7] R. K. L. Ko, P. Jagadpramana, and B. S. Lee, "Flogger: A File-Centric Logger for Monitoring File Access and Transfers Within Cloud Computing Environments," Proc. 10th IEEE Int. Conf. Trst. Sec. Priv. Comp. Com. Trst., 8th IEEE Int. Conf. Emb. Soft. Sys. ICESS, 6th Int. Conf. FCST 2011, pp. 765–771.

[8] R. K. L. Ko, B. S. Lee, and S. Pearson, "Towards achieving accountability, auditability and trust in cloud computing," Commun. Comput. Inf. Sci., vol. 193 CCIS, no. Part 4, 2011, pp. 432–444.

[9] S. Pearson, "Taking Account of Privacy when Designing Cloud Computing Services 2 . Why is it important to take privacy into," Chall. Cloud Comp., 2009, pp. 44–52.

[10] S. Pearson and A. Benameur, "Privacy, Security and Trust Issues Arising from Cloud Computing," 2010 IEEE Second Int. Conf. Cloud Comput. Technol. Sci., 2010, pp. 693–702.

[11] B. Duncan, D. J. Pym, and M. Whittington, "Developing a Conceptual Framework for Cloud Security Assurance," in Cloud Comput. Technol. Sci. (CloudCom), 5th Int. Conf. (Vol. 2). IEEE, 2013, pp. 120–125.

[12] J. Parker et al., "Detection of Mutual Inconsistency in Distributed Systems," in IEEE Tra. Soft. Eng., vol. SE-9, no. 3, 1983, pp. 240–248.

[13] J. Kramer and J. Magee, "Dynamic Configuration for Distributed Systems," IEEE Trans. Softw. Eng., vol. SE-11, no. 4, 1985, pp. 424–436.

[14] F. Mattern, "Virtual Time and Global States of Distributed Systems," Event London, vol. pages, no. 23, 1989, pp. 215–226.

[15] A. Lenk and S. Tai, "Cloud Standby: Disaster Recovery of Distributed Systems in the Cloud," Serv. Cloud Comput., 2014, pp. 32–46.

[16] A.-C. Orgerie, M. Dias de Assuncao, and L. Lefevre, "A Survey on Techniques for Improving the Energy Efficiency of Large Scale Distributed Systems," ACM Comput. Surv., vol. 46, no. 4-47, 2013, pp. 1–35.

[17] H. W. Gottinger, "Internet Economics of Distributed Systems," in Soc. Sci. Educ., vol. 2, no. 6, 2015, pp. 55–70.

[18] J. A. Zachman, "A Framework for Information Systems Architecture," IBM Syst. J., vol. 26, no. 3, 1987, pp. 454–470.

[19] T. J. Norman, N. R. Jennings, P. Faratin, and E. H. Mamdani, "Designing and Implementing a Multi-Agent Architecture for Business Process Management," in PreProceedings ECAI96 Work. Agent Theor. Archit. Lang. ATAL96, N. R. Müller, Jörg P and Wooldridge, Michael J and Jennings, Ed., vol. 1193. New York: Springer, 1997, pp. 261–275.

[20] W. Zhu, L. Vizenor, and A. Srinivasan, "Towards a Reference Architecture for Service-Oriented Cross Domain Security Infrastructures," Internet Distrib. Comput. Syst., 2014, pp. 275–284.

[21] S. Bülow, M. Backmann, and N. Herzberg, "Monitoring of Business Processes with Complex Event Processing," Bus. Process Manag. Work., 2014, pp. 277–290.

[22] W. R. Schulte and Y. V. Natis, "Service Oriented Architectures, Part 1," Gartner, SSA Res. Note SPA-401-068, 1996.

[23] A. Girbea, C. Suciu, S. Nechifor, and F. Sisak, "Design and Implementation of a Service-Oriented Architecture for the Optimization of Industrial Applications," IEEE Trans. Ind. Informatics, vol. 10, no. 1, 2014, pp. 185–196.

[24] W. Picard, Z. Paszkiewicz, S. Strykowski, R. Wojciechowski, and W. Cellary, "Application of the service-oriented architecture at the inter-organizational level," Stud. Comput. Intell., vol. 499, 2014, pp. 125–201.

[25] PWC, "UK Information Security Breaches Survey - Technical Report 2012," PWC2012, Tech. Rep. April, 2012.

[26] Trend, "2012 Annual Security Roundup: Evolved Threats in a 'Post-PC' World," Trend Micro, Tech. Rep., 2012.

[27] Verizon, "2013 Data Breach Investigation Report: A study conducted by the Verizon business risk team." Tech. Rep., 2013.

[28] M. Vouk, "Cloud computing - Issues, research and implementations," ITI 2008 - 30th Int. Conf. Inf. Technol. Interfaces, vol. 16, no. 4, 2008, pp. 235–246.

[29] N. Leavitt, "Is Cloud Computing Really Ready for Prime Time?" Computer (Long. Beach. Calif)., vol. 42, no. January, 2009, pp. 15–20.

[30] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," in IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, 2011, pp. 847–859.

[31] Z. Chen and J. Yoon, "IT Auditing to Assure a Secure Cloud Computing," in 2010 6th World Congr. Serv., 2010, pp. 253–259.

[32] S. Ramgovind, M. M. Eloff, and E. Smith, "The management

[33] M. Armbrust et al., "A view of cloud computing,"Commun. ACM, vol. 53, no. 4, 2010, pp. 50–58.of security in cloud computing," in Proc. 2010 Inf. Secur. South Africa Conf. ISSA 2010, 2010, pp. 1–7.

[34] C. Wang, S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage in Cloud Computing," in IEEE Trans. Comput., vol. PP, no. 99, 2012, pp. 1–14.

[35] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, "Security and Privacy in Cloud Computing: A Survey," in 2010 Sixth Int. Conf. Semant. Knowl. Grids, 2010, pp. 105–112.

[36] B. Grobauer, T. Walloschek, and E. Stöcker, "Understanding cloud computing vulnerabilities," IEEE Secur. Priv., vol. 9, no. 2, 2011, pp. 50–57.

[37] F. Doelitzscher, M. Knahl, C. Reich, and N. Clarke, "Anomaly Detection In IaaS Clouds," in CloudCom, 2013, pp. 387–394.

[38] T. Ruebsamen and C. Reich, "Supporting cloud accountability by collecting evidence using audit agents," in Proc. Int. Conf. Cloud Comput. Technol. Sci. CloudCom, vol. 1, 2013, pp. 185–190.

[39] J. M. López, T. Ruebsamen, and D. Westhoff, "Privacy-Friendly Cloud Audits with Somewhat Homomorphic and Searchable Encryption," in Innov. Com. Serv. (I4CS), 14th Int. Conf., 2014, pp. 95–103.

[40] B. Duncan and M. Whittington, "Enhancing Cloud Security and Privacy: The Power and the Weakness of the Audit Trail," in press.

[41] G. T. Willingmyre, "Standards at the Crossroads," StandardView, vol. 5, no. 4, 1997, pp. 190–194.

[42] B. Duncan and M. Whittington, "Reflecting on Whether Checklists Can Tick the Box for Cloud Security." in Cloud Comp. Tech. Sci. (CloudCom), IEEE 6th Int. Conf. Singapore: IEEE, 2014, pp. 805–810.

[43] R. Holland et al., "Quick Take : 12 Lessons For Security & Risk Pros From The US OPM Breach," 2015, pp. 1–10.

[44] C. Millard, K. Hon, and I. Walden, "The Problem of ' Personal Data ' in Cloud Computing - What Information is Regulated ?" 2011.

[45] W. K. Hon, C. Millard, and I. Walden, "Who is Responsible for 'Personal Data' in Cloud Computing? The Cloud of Unknowing, Part 2,"Leg. Stud., no. 77, 2011, pp. 1–31.

[46] W. K. Hon, J. Hörnle, and C. Millard, "Data Protection Jurisdiction and Cloud Computing When are Cloud Users and Providers Subject to EU Data Protection Law?" Leg. Stud., 2011, pp. 1–40.

[47] J. Prüfer, "How to govern the cloud? Characterizing the optimal enforcement institution that supports accountability in cloud computing," Proc. Int. Conf. Cloud Comp. Tech. Sci., vol. 2, pp. 33–38, 2013.

[48] J. Prufer, "Trusting Privacy in the Cloud," 2014.

[49] M. Quinn, E. Strauss, and G. Kristandl, "The effects of cloud technology on management accounting and business decision-making," Fin. Man., vol. 10, 2014, pp. 54–55.

 ISBN: 978-1-61208-460-2

Enhancing Cloud Security and Privacy: The Power and the Weakness of the Audit Trail

Bob Duncan
Computer Science
University of Aberdeen
Aberdeen, UK
Email: bobduncan@abdn.ac.uk

Mark Whittington
Accounting and Finance
University of Aberdeen
Aberdeen, UK
Email: mark.whittington@abdn.ac.uk

***Abstract*—Information security in the cloud presents a serious challenge. We have identified fundamental weaknesses when undertaking cloud audit, namely the misconceptions surrounding the purpose of audit, what comprises a proper audit trail, what should be included, and how it should be achieved and maintained. A properly specified audit trail can provide a powerful tool in the armoury against cyber-crime, yet it is all too easy to throw away the benefits offered by this simple tool through lack of understanding, incompetence, mis-configuration or sheer laziness. Of course, merely having an effective audit trail is not enough — we actually have to examine it regularly to realise the potential benefits it offers.**

***Keywords*—security; privacy; audit; audit trail.**

I. INTRODUCTION

Achieving information security is not a trivial process. When this involves a cloud setting, the problem intensifies exponentially. Let us first consider how we go about achieving security. Usually, it is achieved by means of compliance with standards, assurance or audit. We provide some useful background on this in [1]. In a non-cloud setting, we have a range of established standards which are well understood by industry. However, when we move to cloud, everything changes. There are an extensive range of cloud standard setting bodies, yet there remains no definitive cloud security standard.

Assurance in non-cloud settings is well understood, but assurance in a cloud setting is less well understood. There are many challenges to overcome and, with Pym, we addressed some of those in earlier work [2] developing a conceptual framework for cloud security assurance, where we addressed: standards, proposed management method and complexity.

One of the fundamental tools that can be used to help ensure cloud security is the simple audit trail. There are, of course, many other challenges, and we revisit these in Section II, where we look at the definition of security goals, compliance with cloud security standards, audit issues, the impact of management approaches on security, and how complexity, the lack of responsibility and accountability affect cloud security. The remainder of the paper is organized as follows: in Section III, we discuss cloud audit, state of the art; in Section IV, we consider misconceptions prevalent across different disciplines of what exactly the audit trail is; in Section V, we discuss how we might improve audit trails in a cloud setting. In Section VI, we discuss our conclusions.

II. CLOUD SECURITY CHALLENGES

A number of challenges need to be addressed in order to achieve the goal of good security. The fundamental concepts of information security are confidentiality, integrity, and availability (CIA), a concept developed in an environment using agency theory to manage director self-interest and inter-corporate transactions. Agency theory recognizes the need to align the objective of agent with principal, though this has been shown to be difficult to achieve in practice. Cloud security is no different, which suggests a different approach is needed.

Ten key security issues have been identified, namely:

- The definition of security goals;
- Compliance with standards;
- Audit issues;
- Management approach;
- Technical complexity of cloud;
- Lack of responsibility and accountability;
- Measurement and monitoring;
- Management attitude to security;
- Security culture in the company;
- The threat environment.

Looking at the definition of security goals, we recognise that the business environment is constantly changing, as are corporate governance rules and this would clearly imply changing security measures are required to keep up to date. More emphasis is now placed on responsibility and accountability [3], social conscience [4], sustainability [5] [6], resilience [7] and ethics [8]. Responsibility and accountability are, in effect, mechanisms we can use to help achieve all the other security goals. As social conscience and ethics are very closely related, we can expand the traditional CIA triad to include sustainability, resilience and ethics. This expansion of CIA can help address some of the shortcomings of agency theory, but also provides a perfect fit to stewardship theory. Stewardship carries a broader acceptance of responsibility than the self-interest embedded in agency, extending to acting in the interests of company owners, society and the environment as a whole. Broadening the definition of security goals gives a more effective way to achieve successful cloud audit, but the added complexity cloud brings may complicate the audit trail.

In earlier work with Pym [2], we developed a conceptual framework to address cloud security. We identified three barriers to good cloud security: standards compliance, management method and complexity. We addressed compliance with standards in [1]. The lack of coherent cloud standards undermines the effectiveness of cloud audit and highlights a fundamental weakness in that process [9] — the use of checklists.

We also considered management method [10], where we addressed the cloud security issue with management method, arguing that historic reliance on agency theory to run companies can undermine effective security. We addressed complexity along with the difficulties in addressing measurement [11],

 ISBN: 978-1-61208-460-2

which can complicate effective audit. In [12], we addressed the lack of responsibility and accountability in standard service level agreements (SLAs). This area has been much neglected, but there are signs that it is being taken more seriously.

On the matter of achieving compliance with standards in practice, we have identified the use of assurance to achieve security through compliance and audit. Turning first to compliance, there are a number of challenges to address. Since the evolution of cloud computing, a number of cloud security standards have evolved, but the problem is that there is still no standard which offers complete security — there is no "one size covers all", which is a limitation. Even compliance with all standards will not guarantee complete security, which, presents another disadvantage [1]. The pace of evolution of new technology far outstrips the capability of international standards organisations to keep up with the changes [13], adding to the problem and meaning it may not be resolved any time soon. We have argued that companies need to take account of these gaps in the standards when addressing compliance. Reliance on compliance alone will undermine effective security. Some key areas above we will not address here. Management attitude to security, the importance of developing a strong security culture in the company, and looking at the threat environment are all areas that merit more extensive and specific research.

We have also considered weaknesses in the approach to cloud audit [14], and we expand on that work here. It is certainly the case that cloud audit is not a mature field, and much early work on cloud audit has focussed on addressing technical issues. We have long held the view that focussing on technical issues alone can never solve cloud security. The business architecture of a company comprises people, process and technology [15], not technology alone, thus focussing on a technical solution alone is likely to undermine security.

III. Cloud Audit, the State of the Art

Vouk [16], in an early description of cloud computing issues, suggests there must be an ability to audit processes, data and processing results, but does not propose a solution. Wang et al. [17] address how the cloud paradigm brings about many new security challenges, which have not been well understood. The authors study the problem of ensuring the integrity of data storage in cloud computing, in particular, the task of allowing a third party auditor (TPA), working on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The authors identify the difficulties, potential security problems and show how to construct an elegant verification scheme for seamless integration of these features into protocol design, but this relies on the willingness of the cloud service provider (CSP) to permit the TPA entry to their systems.

Leavitt [18] suggests CSPs will not be able to pass customer audits if they cannot demonstrate who has access to their data and how they prevent unauthorised personnel from retrieving information. Again, there is no detail given on how to address this. Some CSPs address this by appointing TPAs to audit their systems in advance, and by documenting procedures designed to address customers' data security needs. Bernstein et al. [19] are excited by the prospect of a "cloud of clouds", but are worried about the security processes used to ensure connectivity to the correct server on the other clouds, suggesting some kind of audit-ability would be needed. The authors stress the need for cloud systems to provide strong and secure audit trails, but do not suggest how this might be done.

Pearson and Benameur [20] recognise that achieving proper audit trails in the cloud is an unresolved issue. Wang et al. [21] address privacy preserving public auditing for data storage security in cloud, and are keen to prevent TPA introduced weaknesses to the system, presenting a mechanism to enable a more secure approach to public audit by TPAs. Development of the algorithms is at an early stage and the authors note further improvement needs to take place. Zhou et al. [22] carry out a survey on security and privacy in cloud computing, investigating several CSPs about their concerns on security and privacy issues, and find those concerns are inadequate. The authors suggest more should be added in terms of five aspects (i.e., availability, confidentiality, data integrity, control and audit) for security, but do not provide any detail. Chen and Yoon [23] present a framework for secure cloud computing through IT auditing by establishing a general framework using checklists by following data flow and its life-cycle. The checklists are made based on the cloud deployment models and cloud services models, see [9] for our views on checklists.

Armbrust et al. [24] present a detailed description of what cloud computing is, and note that the possible lack of audit-ability presents the number three barrier to implementation, without proposing any solution. Ramgovind et al. [25] provide an overall security perspective of cloud computing with the aim of highlighting the security concerns that should properly be addressed and managed to realise the full potential of cloud computing. The authors note that possible unwillingness of CSPs to undergo audit presents a real barrier to acceptance and take up. Grobauer et al. [26] note that discussions about cloud computing security often fail to distinguish general issues from cloud-specific issues. The authors express concern that many CSPs do not do enough to ensure the provision of good cloud audit practice and hence evidence proper security.

Doelitzscher et al. [27] present a prototype demonstration of Security Audit as a Service (SAaaS) architecture, a cloud audit system which aims to increase trust in cloud infrastructures by introducing more transparency to both user and cloud provider on what is happening in the cloud. This system aims to keep track of changes to the infrastructure as VMs are deployed, moved or shut down. Hale and Gamble [28] note that current SLAs focus on quality of service metrics and lack the semantics needed to express security constraints that could be used to measure risk. The authors present a framework, called SecAgreement (SecAg), that extends the current SLA negotiation standard to allow security metrics to be expressed on service description terms and service level objectives. The framework seeks to provide a lightweight approach, which can leave some weaknesses not fully addressed.

Pappas et al. [29] present CloudFence, a framework that allows users to independently audit the treatment of their private data by third-party online services, through the intervention of the cloud provider that hosts these services. The authors demonstrate that CloudFence requires just a few changes to existing application code, while it can detect and prevent a wide range of security breaches, ranging from data leakage attacks using SQL injection, to personal data disclosure due to missing or erroneously implemented access control checks. It addresses data held by a CSP, but does not claim to provide a complete audit trail. Xie and Gamble [30] outline a tiered

 ISBN: 978-1-61208-460-2

approach to auditing information in the cloud, but with little detail provided. The approach provides perspectives on auditable events that may include compositions of independently formed audit trails. Zhu et al. [30] propose the use of provable data possession (PDP), a cryptographic technique for verifying the integrity of data, without retrieving it, as part of a means of carrying out audit on the data. This tool can prove the integrity of data, but does not consider who has accessed the data.

Ruebsamen and Reich [31] propose the use of software agents to carry out continuous audit processing and reporting. The authors propose continuous audit to address the dynamically changing nature of cloud use, so as to ensure evidence concerning vital periods of use are not missed. The use of a separate evidence store is a major plus, and the tools developed look very interesting. The authors note that an audit of the cloud layer alone will not be enough. Doelitzscher et al. [32] propose the use of neural networks to analyse and learn the normal usage behaviour of cloud customers, so that anomalies which originate from a cloud security incident caused by a compromised virtual machine can be detected. While retrospective tests on collected data have proved very effective, the system has yet to reach a sufficient level of maturity to be deployed in a live environment.

Doelitzscher et al. [33] present a cloud audit policy language for their SAaaS architecture. The authors describe the design and implementation of the automated audit system of virtual machine images, which ensures legal and company policies are complied with. They also discuss how on-demand software audit agents that maintain and validate the security compliance of running cloud services are deployed. Thorpe et al. [34] present a framework for forensic based auditing of cloud logs. The authors explore the requirements of a cloud log forensics service oriented architecture (SOA) framework for performing effective digital investigation examinations in these abstract web services environments. Of course, these services need to be activated and protected before these tools can be deployed. Wang et al. [35] propose a secure cloud storage system supporting privacy-preserving public auditing. The authors further extend their proposal to enable the TPA to perform audits for multiple users simultaneously and efficiently, an improvement on earlier work.

Lopez et al. [36] propose privacy-friendly cloud audits by applying Somewhat Homomorphic Encryption (SHE) and Public-Key Searchable Encryption (PEKS) to the collection of digital evidence. The authors show that their solution can provide client privacy preserving audit data to cloud auditors. Whilst good for privacy, this does not cover all angles. Shameli-Sendi and Cheriet [37] propose a framework for assessing the security risks associated with cloud computing platforms, but propose no solution on how to achieve a high standard of audit. Xiong and Chen [38] consider how to allocate sufficient computing resources, but not to over-provision them, to process and analyse audit logs for ensuring the guarantee of security of an SLA, referred to as the SLA-based resource allocation problem, for high-performance cloud auditing. This is interesting because of the tools developed. However, it is geared toward enforcement of SLAs in high performance computing, rather than for security auditing.

The common theme running through much of this work is that there is a recognition of the need for proper audit, but little idea of how to go about it. Where tools are developed, many are excellent for what they are designed for, but do not offer a complete solution to the problem. It is clear that the consistent lack of input from the accounting profession is not helping advance the state of the art, and we would call for more input from the accounting profession. Cloud computing is such a radical change from traditional computing approaches, that we now need to involve a wider range of disciplines, working together to try to address what represents a major challenge.

IV. The Audit Trail

Auditing in the accountancy world has enjoyed the benefit of over a century of practice and experience, yet there remain differences of opinion with a number of problems yet to be resolved. Duncan and Whittington [1] provide some background on this issue. Cloud computing audit can not be considered a mature field, and there will be some way to go before it can catch up with the reflection and rigour of the accounting profession. An obvious area of weakness arises when taking audit professionals from the accounting world out of their comfort zone, and placing them in a more technical field. Equally, the use of people with a computing background can overcome some of these issues, but their lack of audit background presents another weakness.

A fundamental element of the audit process is the audit trail, and having two disciplines involved in providing cloud audit services means we have two different disciplines to contend with, namely accounting professionals and security professionals. An obvious concern is what is meant by the term "audit trail". It is easy to assume that everyone is talking about the same thing, but is that actually the case? To an accounting professional, the meaning of an audit trail is very clear.

The Oxford English Dictionary (OED) [39] has two useful definitions of an audit trail: "(a) Accounting: a means of verifying the detailed transactions underlying any item in an accounting record; (b) Computing: a record of the computing processes which have been applied to a particular set of source data, showing each stage of processing and allowing the original data to be reconstituted; a record of the transactions to which a database or a file has been subjected". This suggests common understanding, but often this is not evident in computing research.

Some 20 years ago, the National Institute of Standards and Technology (NIST) [40] provided, in the context of computing security, a very detailed description of what an audit trail is, and this is wholly consistent with the OED definition. When we look at the definitions in use in some cloud audit research papers, we start to see a less rigorous understanding of what an audit trail is. For example, Bernstein [19] suggests the audit trail comprises: events, logs, and analysis thereof, Chaula [41] suggests: raw data, analysis notes, preliminary development and analysis information, processes notes, etc.

Pearson et al. [20] recognise that achieving proper audit trails in the cloud is an unresolved issue. Ko et al. [42] explicitly note that steps need to be taken to prevent audit trails disappearing after a cloud instance is shut down. Ko [43] recognises the need to collect a multiplicity of layers of log data, including transactional audit trails in order to ensure accountability in the cloud. The EU Article 29 Working Party [44] raises several cloud-specific security risks, such as loss of governance, insecure or incomplete data deletion, insufficient

 ISBN: 978-1-61208-460-2

audit trails or isolation failures, which are not sufficiently addressed by existing Safe Harbor principles on data security.

The audit trail can be a very powerful tool in the fight against attack. Just as the audit trail offers forensic accountants a means to track down fraudulent behaviour in a company, so the audit trail in a cloud setting, providing it can be properly protected against attack, offers forensic scientists an excellent basis to track intrusions and other wrongdoing. In the event of a catastrophic attack, it should be possible to reconstruct the system that has been attacked, in order to either prove the integrity of the system values, or in a worst case scenario, reconstruct the system from scratch. The redundancy offered by the simple audit trail, often seen by many as an unnecessary duplication, will prove invaluable in the event of compromise.

Many cloud users are punctilious about setting up proper audit trails, but sometimes forget that when a virtual machine (VM) running in the cloud is shut down, everything, including the audit trail data they have so assiduously collected, disappears as soon as the VM shuts down [42], unless steps are taken to prevent this loss. In real world conditions, most database software ships with inadequate audit trail provision in the default settings. Anderson [45] states that the audit trail should only be capable of being read by users. While it is simple enough to restrict users to read-only access, this does not apply to the system administrators. This presents an issue where an intruder gets into a system, escalates privileges until root access is obtained, and is then free to manipulate, or delete the audit trail entries in order to cover their tracks.

Cloud users often assume that the VMs they are running will be under their sole control. However, the VMs run on someone else's hardware — the CSP's, who also employ system administrators, and sometimes employ temporary staff, some of whom are also system administrators. While the CSP may vet their own staff to a high level, this is often overlooked with temporary employees. Network connections too are often virtualised, opening up yet more avenues of attack.

A cloud user can take as many steps to secure their business as they wish, but a key ingredient in the equation is the fact that all cloud processes run on somebody else's hardware, and often software too — the CSP's. The cloud relationship needs to include the CSP as a key partner in the pursuit of achieving security [12]. Unless and until CSPs are willing to share this goal, technical solutions will be doomed to failure.

V. How can we improve the audit trail?

These vulnerabilities are not new, and are well known to security professionals, and while many companies do use security professionals, many do not. Regardless, companies continue to be breached. As stated in the introduction, achieving information security is not a trivial process, but in a cloud setting, this becomes far more difficult, due to the complexity of relationships between myriad actors involved in cloud ecosystems, as well as the other issues discussed in Section II. Audit is difficult. Cloud audit is far more difficult, with far more weaknesses to overcome. Proper audit trails alone will not solve cloud security, but will go a long way to helping with this goal if some simple steps are taken.

In the accounting world, an understanding of exactly what is meant by an audit trail, and its importance, is a fundamental part of the training every accountant undertakes. Looking at the literature on cloud audit, it is obvious that there is a need for input from the accounting profession, and the authors would wish to encourage and see more input from that source. It is clear that there is no shortage of input on the technical side, but the authors believe there is room for a valuable contribution to be made by the accounting profession, with many lessons learned over many decades. There is also no doubt that further work is needed, and work on audit trails can prove to be both cost effective and productive in helping with security.

Some interesting work is being done on information flow control and legal issues in cloud [46] [47] [48] [49] [50] [51] which we believe, while a little light on the audit side, has the potential to offer good improvements to cloud users to enhance their security and privacy, and to achieve compliance.

In looking at the current approach to the use of the audit trail, there are three fundamental weaknesses which need to be addressed, yet which are relatively simple to address. First, inadequate default logging options can result in insufficient data being collected for the audit trail. Second, there is a lack of recognition that the audit trail data can be accessed by a malicious user gaining root privileges, which can lead to the removal of key data showing who compromised the system, and what they did once they had control of it. Third, failure to ensure log data is properly collected and moved to permanent storage can lead to loss of audit trail data, either when an instance is shut down, or when it is compromised. These weaknesses apply equally to cloud and non-cloud systems.

On the first point, we look at one of the most popular open source database programmes in general use today — MySQL. The vast majority of implementations use standard default settings on installation, or install the programme as part of a standard Linux, Apache, MySQL and PHP (LAMP) server. With a LAMP server, all four of the constituent elements are set up using the default settings. This works very well for easy functionality "out of the box", which is the aim of a LAMP server. But, this does not adequately address security in the four elements of the LAMP server, and applies equally to cloud and non-cloud systems.

MySQL offers the following audit trail options:

- Error log — Problems encountered starting, running, or stopping mysqld;
- General query log — Established client connections and statements received from clients;
- Binary log — Statements that change data (also used for replication);
- Relay log — Data changes received from a replication master server;
- Slow query log — Queries that took more than long_query_time seconds to execute;
- DDL log (metadata log) — Metadata operations performed by Data Definition Language (DDL) statements.

By default, no logs are enabled, except the error log on Windows. Some versions of Linux send the Error log to syslog.

Oracle offer an audit plugin for Enterprise (paid) Editions of MySQL. This allows a range of events to be logged, but again, by default, most are not enabled. The MariaDB company, whose author originally wrote MySQL, have their own open source audit plug-in, and offer a version suitable for MySQL. It has the following functionality:

 ISBN: 978-1-61208-460-2

- CONNECTION — Logs connects, disconnects and failed connects (including the error code);
- QUERY — Queries issued and their results (in plain text), including failed queries due to syntax or permission errors;
- TABLE — Which tables were affected by query execution;
- QUERY_DDL — Works as the 'QUERY' value, but filters only DDL-type queries (CREATE, ALTER, etc);
- QUERY_DML — Works as the 'QUERY' value, but filters only Data Manipulation Language (DML) DML-type queries (INSERT, UPDATE, etc).

By default, logging is set to off. Thus, those users who rely on default settings for their systems are immediately putting themselves at a severe disadvantage.

On the second point, as Anderson [45] states, the audit trail should only be capable of being read by users. This presents a problem in a cloud setting, where the software being used is running on someone else's hardware. There is a risk of compromise from an outside user with malicious intent. There is also a risk of compromise by someone working for the CSP. While the CSP may well take vetting of staff seriously, there may be situations that arise where a temporary contract worker is engaged at short notice who is subject to lesser scrutiny. This applies equally to cloud and non-cloud systems.

Looking at the third point, where MySQL data logging is actually switched on, all data is logged to the running instance. This means the data remains accessible to any intruder who successfully breaches the system, allowing them to cover their own tracks by deleting any entries which relate to their intrusion of the system, or to simply delete the entire audit trail files. And, when the running instance is shut down, all the data disappears anyway. In a non-cloud situation, the data is still visible to the attacker, but the forensic trail may still be left for investigation. However, in a cloud instance, if the data is not safely stored and the running instance is shut down, the forensic trail is more likely to be permanently lost.

These three points are not generally considered, yet they present a serious weakness to the success of maintaining the audit trail. Yet, these are relatively trivial to address by simply turning on data logging and sending all log output to an independent secure server under the control of the cloud user. Adding an Intrusion Detection system (IDS) is also a useful additional precaution to take, and this should be run on an independent secure server under the control of the cloud user. Using an audit plug-in in addition to all the basic logging capabilities, is also a useful thing to do. While there may be some double processing involved, it is better to have more data than none at all. Where the MySQL instance forms part of a LAMP server, it would be prudent to make some elementary security changes to the setup of the Linux operating system, the Apache web server, and to harden the PHP installation.

It is rather worrying that in 2012, [52] report an average of 6 months between breach and discovery, a clear indication that very few firms scrutinise their server logs, with most discovery being advised by external bodies, such as customers, financial institutions or fraud agencies. It is encouraging to see that three years later [53], the time between breach to discovery has been drastically reduced. This still leaves a large gap where compromised systems may still be under the control of malicious users, which is a worry. Thus, in addition to making the simple suggestions we propose above, cloud users should also make sure they actually review their audit trail logs. It is vital to understand when a security breach has occurred, which records have been accessed, compromised or stolen. While this is not a foolproof method of achieving cloud security, it is an effective first step to help deliver far higher affordable security than many companies currently achieve.

The authors have over 50 years of experience of audit and internal audit in industry between them, and this knowledge has been brought to bear in addressing this work. This initial review of the state of the art raises serious concerns over how little is being done. With some input from and partnership with the accounting profession, it may be possible to work to achieve far more effective levels of cloud audit, which in turn, can lead to better levels of security being achieved.

VI. Conclusion

We have looked at some of the challenges facing companies who seek to obtain good cloud security assurance. We have seen how weaknesses in standard CSP SLAs can impact on cloud security. We have identified cloud security standards issues, and how that might impact cloud security. We have considered how the lack of accountability can impact security. We have discussed how these issues must also be addressed.

The practice of using default settings when installing software in a cloud environment is clearly asking for trouble, yet still persists. The simple steps proposed by the authors are relatively easy to implement, need not be particularly expensive to implement and maintain, and providing some on-going monitoring of the audit trail logs is carried out, will certainly prove beneficial. Inspecting the logs need not be challenging or costly — there are many software solutions available to address this task. Complicated solutions generally lead to complex problems, as the more complex the solution, the more the risk of ineffective configuration and maintenance can lead to compromise in security. Yet all too often, the simple steps that can really help improve security are ignored.

We certainly believe that more work needs to be done in this area, and it would be beneficial to encourage the accounting audit profession to get involved. After all, it is their neck on the block when they sign off an audit, and anything that can reduce risk to themselves, as well as their clients, has to be a good thing. We have touched on how these difficult areas of security might easily be approached as part of a comprehensive security solution using simple and inexpensive methods. Clearly, companies could benefit from further research in several of these areas. It is also clear that no one profession is equipped to deal with this challenge. However, we would caution that action is needed now, not several years down the line when research reaches a more complete level of success in these areas. The threat environment is too dangerous. Companies have to act now to try to close the door, otherwise it may be too late.

References

[1] B. Duncan and M. Whittington, "Compliance with Standards, Assurance and Audit: Does this Equal Security?" in Proc. 7th Int. Conf. Secur. Inf. Networks. Glasgow: ACM, 2014, pp. 77–84.

 ISBN: 978-1-61208-460-2

[2] B. Duncan, D. J. Pym, and M. Whittington, "Developing a Conceptual Framework for Cloud Security Assurance," in Cloud Comp. Tech. Sci. (CloudCom), 2013 IEEE 5th Int. Conf. (Vol. 2). Bristol: IEEE, 2013, pp. 120–125.

[3] M. Huse, "Accountability and Creating Accountability: a Framework for Exploring Behavioural Perspectives of Corporate Governance," Br. J. Manag., vol. 16, no. S1, 2005, pp. S65–S79.

[4] A. Gill, "Corporate Governance as Social Responsibility: A Research Agenda," Berkeley J. Int'l L., vol. 26, no. 2, 2008, pp. 452–478.

[5] C. Ioannidis, D. Pym, and J. Williams, "Sustainability in information stewardship: Time Preferences, Externalities and Social Co-Ordination," in Weis 2013, pp. 1–24.

[6] A. Kolk, "Sustainability, accountability and corporate governance: Exploring multinationals' reporting practices." Bus. Strateg. Environ., vol. 17, no. 1, 2008, pp. 1–15.

[7] F. S. Chapin, G. P. Kofinas, and C. Folke, Principles of ecosystem stewardship: Resilience-based natural resource management in a changing world. Springer, 2009.

[8] S. Arjoon, "Corporate Governance: An Ethical Perspective," J. Bus. Ethics, vol. 61, no. 4, 2012, pp. 343–352.

[9] B. Duncan and M. Whittington, "Reflecting on Whether Checklists Can Tick the Box for Cloud Security." in Cloud Comp. Tech. Sci. (CloudCom), IEEE 6th Int. Conf., Singapore: IEEE, 2014, pp. 805–810.

[10] B. Duncan and M. Whittington, "Company Management Approaches - Stewardship or Agency: Which Promotes Better Security in Cloud Ecosystems?" in Cloud Comput. 2015, Nice: IEEE, 2015, pp. 1–6.

[11] B. Duncan and M. Whittington, "The Importance of Proper Measurement for a Cloud Security Assurance Model," in Cloud Comp. Tech. Sci. (CloudCom), 2015 IEEE 7th Int. Conf., Vancouver, 2015, pp. 1–6.

[12] B. Duncan and M. Whittington, "Enhancing Cloud Security and Privacy: Broadening the Service Level Agreement," in Trustcom/BigDataSE/ISPA, IEEE. Vol. 1. IEEE, Helsinki, 2015, pp. 1–6.

[13] G. T. Willingmyre, "Standards at the Crossroads,"StandardView, vol. 5, no. 4, 1997, pp. 190–194.

[14] B. Duncan and M. Whittington, "Enhancing Cloud Security and Privacy: The Cloud Audit Problem," in press.

[15] PWC, "UK Information Security Breaches Survey - Technical Report 2012," PWC Tech. Rep. April, 2012.

[16] M. Vouk, "Cloud computing- Issues, research and implementations," ITI 2008 - 30th Int. Conf. Inf. Technol. Interfaces, vol. 16, no. 4, 2008, pp. 235–246.

[17] L. Wang, J. Zhan, W. Shi, Y. Liang, and L. Yuan, "In cloud, do MTC or HTC service providers benefit from the economies of scale?" Proc. 2nd Work. Many-Task Comp. Grids Supcomp. - MTAGS, 2009, pp. 1–10.

[18] N. Leavitt, "Is Cloud Computing Really Ready for Prime Time?" Computer (Long. Beach. Calif)., vol. 42, no. January, 2009, pp. 15–20.

[19] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the intercloud - Protocols and formats for cloud computing interoperability," in Proc. 2009 4th Int. Conf. Internet Web Appl. Serv. ICIW 2009, 2009, pp. 328–336.

[20] S. Pearson and A. Benameur, "Privacy, Security and Trust Issues Arising from Cloud Computing," 2010 IEEE Second Int. Conf. Cloud Comput. Technol. Sci., 2010, pp. 693–702.

[21] C. Wang, S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage in Cloud Computing," in IEEE Trans. Comput., vol. PP, no. 99, 2012, pp. 1–14.

[22] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, "Security and Privacy in Cloud Computing: A Survey," in 2010 Sixth Int. Conf. Semant. Knowl. Grids, 2010, pp. 105–112.

[23] Z. Chen and J. Yoon, "IT Auditing to Assure a Secure Cloud Computing," in 2010 6th World Congr. Serv., 2010, pp. 253–259.

[24] M. Armbrust et al., "A view of cloud computing," Commun. ACM, vol. 53, no. 4, 2010 pp. 50–58.

[25] S. Ramgovind, M. M. Eloff, and E. Smith, "The management of security in cloud computing," in Proc. Inf. Sec. S. A. Conf. ISSA, 2010, pp. 1–7.

[26] B. Grobauer, T. Walloschek, and E. Stöcker, "Understanding cloud computing vulnerabilities," IEEE Secur. Priv., vol. 9, no. 2, 2011, pp. 50–57.

[27] F. Doelitzscher et al., "Validating cloud infrastructure changes by cloud audits," in Proc. - 2012 IEEE 8th World Congr. Serv. Serv. 2012, 2012, pp. 377–384.

[28] M. L. Hale and R. Gamble, "SecAgreement: Advancing security risk calculations in cloud services," in Proc. - 2012 IEEE 8th World Congr. Serv., 2012, pp. 133–140.

[29] V. Pappas, V. Kemerlis, A. Zavou, M. Polychronakis, and A. D. Keromytis, "CloudFence: Enabling Users to Audit the Use of their Cloud-Resident Data," 2012.

[30] Y. Zhu, H. Hu, G.-J. Ahn, and S. S. Yau, "Efficient audit service outsourcing for data integrity in clouds," J. Syst. Softw., vol. 85, no. 5, 2012, pp. 1083–1095.

[31] T. Ruebsamen and C. Reich, "Supporting cloud accountability by collecting evidence using audit agents," in Proc. Int. Conf. Cloud Comput. Technol. Sci. CloudCom, vol. 1, 2013, pp. 185–190.

[32] F. Doelitzscher, M. Knahl, C. Reich, and N. Clarke, "Anomaly Detection In IaaS Clouds," in CloudCom, 2013, pp. 387–394.

[33] F. Doelitzscher et al., "[DRKK+13] Sun Behind Clouds - On Automatic Cloud Security Audits and a Cloud Audit Policy Language," J. Adv. vol. 6, no. 1, 2013, pp. 1–16.

[34] S. Thorpe et al., "Towards a forensic-based service oriented architecture framework for auditing of cloud logs," in Proc. - 2013 IEEE 9th World Congr. Serv. Serv. 2013, 2013, pp. 75–83.

[35] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," IEEE Trans. Comput., vol. 62, no. 2, 2013, pp. 362–375.

[36] J. M. López, T. Ruebsamen, and D. Westhoff, "Privacy-Friendly Cloud Audits with Somewhat Homomorphic and Searchable Encryption," in Innov. Com. Serv. (I4CS), 14th Int. Conf., 2014, pp. 95–103.

[37] A. Shameli-Sendi and M. Cheriet, "Cloud Computing: A Risk Assessment Model," Cloud Eng. (IC2E), IEEE Int. Conf., 2014, pp. 147–152.

[38] K. Xiong and X. Chen, "Ensuring Cloud Service Guarantees Via Service Level Agreement (SLA) -based Resource Allocation," in Dist. Comp. Sys. Work. (ICDCSW), IEEE 35th Int. Conf., 2015, pp. 35–41.

[39] OED, "Oxford English Dictionary," 1989. [Online]. Available: www.oed.com [Retrieved: Feb 2016]

[40] D. Gollmann, "Computer Security," NIST, Tech. Rep. 800, 2011.

[41] J. Chaula, "A Socio-Technical Analysis of Information Systems Security Assurance: A Case Study for Effective Assurance," Ph.D. thesis, 2006.

[42] R. K. L. Ko et al., "TrustCloud: A framework for accountability and trust in cloud computing," Proc. - IEEE World Cong. Serv., 2011, pp. 584–588.

[43] L. F. B. Soares, D. A. B. Fernandes, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, "Security, Privacy and Trust in Cloud Systems," in Secur. Priv. Trust Cloud Syst. Springer, 2014, ch. Data Accou, 2014, pp. 3–44.

[44] Eu, "Unleashing the Potential of Cloud Computing in Europe," 2012.

[45] R. J. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, C. A. Long, Ed. Wiley, 2008, vol. 50, no. 5.

[46] T. Pasquier, B. Shand, and J. Bacon, "Information Flow Control for a Medical Records Web Portal," Cl.Cam.Ac.Uk, 2013, pp. 1–8.

[47] J. Bacon et al., "Information flow control for secure cloud computing," IEEE Trans. Netw. Serv. Manag., vol. 11, no. 1, 2014, pp. 76–89.

[48] T. F. J. Pasquier, J. Singh, J. Bacon, and O. Hermant, "Managing Big Data with Information Flow Control," in Cloud Comput. Technol. Sci. (CloudCom), 2015 IEEE 7th Int. Conf., vol. 2, 2015, pp. 1–8.

[49] J. Singh et al., "Regional Clouds: Technical Considerations," no. UCAM-CL-TR-863, 2014.

[50] J. Singh, J. Powles, T. Pasquier, and J. Bacon, "Seeing through the clouds: Management, control and compliance for cloud computing," Cloud Comput., 2015, pp. 1–12.

[51] W. Hon, E. Kosta, C. Millard, and D. Stefanatou, "Cloud Accountability: The Likely Impact of the Proposed EU Data Protection Regulation," Queen Mary Sch. Law Leg. Stud. Res. Pap., no. 172, 2014, pp. 1–54.

[52] Trustwave, "2012 Global Security Report," Tech. Rep., 2012.

[53] Verizon, "Verizon 2015 Data Breach Investigation Report," Tech. Rep., 2015.

 ISBN: 978-1-61208-460-2

Online Traffic Classification Based on Swarm Intelligence

Takumi Sue, Yuichi Ohsita, and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan
{t-sue, y-ohsita, murata}@ist.osaka-u.ac.jp

***Abstract*—In this paper, we propose a new traffic classification method which constructs hierarchical clusters using the features of uncompleted flows. By constructing the hierarchical groups, we can identify the similarity of the groups. If flows of a new application construct a new group in the lower layer, but they are classified in an existing group in the upper layer, the manager can estimate the characteristic of the new application from the characteristic of the existing group. In our method, the hierarchical groups are constructed based on the clustering method called AntTree; the each flow moves over the tree and find the nodes whose similarity to the nodes exceeds the threshold. By setting the threshold based on the number of monitored packets of the flow, we classify the flow if the features of the flow become sufficiently accurate. Otherwise we wait another packets that improve the accuracy of the features.**

Keywords*—*Traffic Classification; Hierarchical Clustering; Swarm Intelligence

I. INTRODUCTION

As the Internet has become playing an important role in our society, the number of types of services provided through the Internet increases. The requirements to the network depend on the type of the service. The video streaming application requires enough bandwidth according to the bit rate of the video. On the other hand, the interactive application such as online game requires the communication with low latency instead of the large bandwidth.

The network managers should manage their network so as to provide sufficient network performance required by each service. For example, Miyamura et al. [1] proposed a method that constructs a virtual network for each service. In this method, each virtual network is dynamically reconfigured so as to provide a performance required by the service corresponding to the network. To manage the network based on the types of the service, we need to classify the traffic based on the application.

The traditional classification of the traffic uses the port numbers [2]. However, in recent years, a large number of types of the applications, such as YouTube and network game, have become provided through HTTP [3]. All of these applications uses 80 or 443 port. As a result, the traffic classification using the port numbers is no longer applicable in the Internet.

Therefore, the traffic classification methods based on the features of the traffic have been proposed [2], [4]–[6]. The packet sizes and packet arrival intervals depend on the types of the application, and the protocol used by the application. The traffic classification methods based on the features monitors the packet size or packet arrival intervals for each flow. Then, they classify the flows based on the monitored features by using the clustering methods, in which the flows are grouped so that the flows with the similar features belongs to the same group.

The traffic classification should be performed as soon as possible after the flow arrives. Even if the network manager sets the rule to relay the flow according to the types of the application, the rule cannot work before the classification of the flow is completed. The existing method, however, cannot classify the flow before monitoring the flow is completed. One approach is to classify the flow after the predefined number of packets are monitored. By setting the required number of packets to the small value, we can classify the flow soon after the flow arrives. However, the features of the flow obtained by monitoring the small number of packets may be inaccurate, and some application may be difficult to classify based on such inaccurate information.

Another problem in the existing traffic classification is that the group constructed by the classification methods does not imply the characteristic of the group. When flows of a new application comes, the flows are classified into a new group. However, the existing classification methods do not provide the information whether the newly constructed group has the similar characteristic to the existing other groups. As a result, it is difficult to estimate the characteristic of the new application.

In this paper, we propose a new traffic classification method to solve the above problems. Our method is based on the hierarchical clustering [7]. In the hierarchical clustering, the groups of the flows are hierarchically constructed; the flows are clustered into a small number of groups in the upper layer, and the flows belonging to the same group in the upper layers are clustered into several groups in the lower layer. By constructing the hierarchical groups, we can identify the similarity of the groups. If flows of a new application construct a new group in the lower layer, but they are classified in an existing group in the upper layer, the manager can estimate the characteristic of the new application from the characteristic of the existing group.

In our method, the hierarchical groups are constructed based on the clustering method called AntTree [8]. AntTree is inspired by the behavior of ants constructing the tree. In the AntTree, an ant, which corresponds to an item required to be classified, walks on the tree constructed by the other ants. Then, if the ant find the ant whose similarity exceeds the threshold, the ant is connected to the similar ant. If the nearby ants do not have the similarity exceeding the threshold, the ant updates the threshold and goes to another place on the tree. By continuing this process, the hierarchical tree, where similar

 ISBN: 978-1-61208-460-2

ants are connected, is constructed.

In our method, we extend AntTree to consider the accuracy of the features. The features of the flow become accurate as the number of monitored packets increases. We classify the flow if the features of the flow become sufficiently accurate. Otherwise we wait another packets that improve the accuracy of the features. To achieve this, we extend the AntTree by setting the threshold of the similarity to connect the ants considering the number of monitored packets. By doing so, if the quite similar group to the newly arrived flow exists, the flow is classified soon after the flow starts. Otherwise, our method waits another packet to improve the accuracy, and the flow is classified after the sufficient packets are monitored.

The rest of this paper is organized as follows. Section II explains the related work. Section III explains our online hierarchical traffic classification method. In Section IV , we conduct an experiment with real traffic data. The conclusion and future work are mentioned in Section V.

II. Related Work

This section explains the related work.

A. Traffic Classification

There are several papers proposing a method to classify the traffic using the features of the monitored traffic.

Roughan et al. proposed a method to classify the traffic through the supervised machine learning [4]. In this method, the newly obtained data is classified as the class of its nearest neighbor from the training data set. Zhang et al. improved the accuracy of the nearest neighbor approach when the number of the training data set is small by incorporating the correlation information of the flows [9]. Moore et al. also proposed a method to classify the traffic based on the supervised machine learning [5]. In this method, the traffic is classified by using the Naïve Bayes classifier. Nguyen et al. also used the Naïve Bayes to classify the traffic [10]. This method uses the features of a small number of most recent packets to obtain the features. Then, applying the Naïve Bayes, this method classify the current flows. Zhang et al. also used the Naïve Bayes classifier [11]. In this method, flow correlation information is modeled by bag-of-flow. Then, the features are extracted for the represent traffic flows. The traffic is classified by aggregating the output of the Naïve Bayes classifiers using the extracted features. Li et al. proposed a method to construct the decision tree from the training data set [12]. Then the traffic is classified based on the constructed decision tree. Jin et al. proposed a traffic classification method using multiple simple linear binary classifiers [13]. In this method, each classifier can be easily trained. Then, combining the multiple classifiers, we can accurately classify the traffic.

The supervised machine learning methods described above require the training data set. However, as we discussed in Section I, it is difficult to prepare the training data set including the suitable labels for the traffic, because the new applications, which are unknown when the system to classify the traffic starts, emerge.

The methods to classify the traffic based on the clustering have also been proposed.

Erman et al. applied the clustering method to the traffic classification [14]. They used the K-means method, DBSCAN, and AutoClass, and demonstrated that these clustering algorithms can classify the traffic accurately. Bernaille et al. proposed a method to classify the traffic online using the K-means method [6]. In this method, the clusters are constructed offline by using the training data set. Then, flows are classified online by searching the cluster corresponding to the flow. However, this approach cannot classify the traffic which corresponds to the application, which was not included in the training data set. The clustering method can be used to solve this problem. Zhang et al used the K-means method to detect the unknown flows [15].

Recently, Wang et al. extended the K-means method to improve the accuracy of the traffic classification [16]. In this method, the accuracy is improved by considering the information of the flow inferred from the packet headers as the constraint on the clustering.

However, the existing traffic classification methods based on the clustering have the following two problems. (1) These methods do not consider the case that the new applications emerges. Though the method proposed by Zhang et al [15] can detect the unknown flows, it cannot estimate the characteristic of the new flow. (2) These methods assume that the accurate features of the flow are obtained before classifying the traffic. However, the accurate features may not be able to be obtained before the flow is completed.

In this paper, we propose a clustering method which solves the above problems. Our clustering method can be applicable to the existing traffic classification method based on the clustering; our clustering method can be run by using any features of the flows.

B. Clustering

There are many algorithms to construct the clusters.

K-means is one of the most popular clustering algorithms. In the K-means method, the number of clusters k is given as a parameter. Then, the k clusters are constructed so as to minimize the distance from each data point to the center of the cluster the data point belongs to, which is defined by the mean of the data points within the cluster. However, the result of the K-means only indicates the cluster which each data point belongs to. Thus, we cannot understand the similarity of the data points belonging to different clusters.

The hierarchical clustering methods can solve the above problem. In the hierarchical clustering methods, the data points are clustered into a small number of groups in the upper layer, and the data points belonging to the same group in the upper layers are clustered into several groups in the lower layer. By constructing the hierarchical groups, we can identify the similarity of the groups.

The ClusTree is one of the hierarchical clustering methods, which allow to update the clusters online [7]. This method constructs the tree, including two kinds of nodes, inner node

 ISBN: 978-1-61208-460-2

and leaf node. The leaf node indicates a fine-grained cluster, and has the pointer to the feature values of the corresponding data points. The inner node corresponds to the coarse-grained cluster, which includes the data points included in its children nodes. The inner node has the pointer to the aggregated features of the data points included in its children nodes. In the ClusTree, the clusters can be updated by (1) searching the leaf node corresponding to the new data point from the root node, and (2) updating the features on the path from the root node to the leaf node.

The ClusTree can be updated online, but does not consider the case that the features are inaccurate. Therefore, in this paper, we propose a new clustering method based on the ClusTree, considering the case that the features are inaccurate.

AntTree [8] is another hierarchical clustering method. AntTree is a method inspired by the behavior of the ants constructing a tree. In this process, each data point acts as an ant; each data point walks around the tree to find the node similar to the flow, and connects it to the found node. In the AntTree, the nodes in the constructed tree are the data points. Therefore, it is difficult to interpret the constructed tree hierarchically; we need to determine the data points corresponding to the inner nodes when constructing the fine-grained flow. However, the idea of the AntTree is useful to handle the inaccuracy of the features. In the AntTree, each data point has a threshold to determine whether the other data points are similar to it. Then, each data point walks around the tree based on the threshold. Though the original AntTree updates the threshold based on the number of nodes the data points visited, we can consider the inaccuracy of the features by setting the threshold based on the accuracy of the features. Therefore, in this paper, we introduce the method based on the AntTree to search the cluster each data point belongs to.

III. Online hierarchical traffic classification method

This section explains our online hierarchical traffic classification method.

A. Overview

In this paper, we develop the traffic classifier, which classifies the flow passing the classifier. The flow is defined by the set of packets between the same IP address pairs using the same server port. We regard the well-known ports as the server ports, and the other ports as the client ports. In this paper, the packets using the same server port is regarded as the packets belonging to the same flow even if the packets have the different client port, because some application such as Web browser uses the multiple TCP connections for the same transaction.

The traffic classifier monitors the flows. When the traffic classifier receive a packet, it identifies the flow the packet belongs to. Then, it stores the information of the packet. The traffic classifier updates the features of the flow each time a packet of the flow are monitored.

The traffic classifier classifies the flow based on its features. To classify the flow, we use the hierarchical clustering methods. In the hierarchical clustering, the groups of the flows are hierarchically constructed; the flows are clustered into a small number of groups in the upper layer, and the flows belonging to the same group in the upper layers are clustered into several groups in the lower layer. By constructing the hierarchical groups, we can identify the similarity of the groups.

Each time the features of the flow is updated, the traffic classifier runs the clustering method. That is, the clustering is performed before the flow completed by using the inaccurate features, which leads to wrong classification. Therefore, we use the clustering method considering the accuracy of the features. The accuracy of the features of the flow increases as the number of monitored packets becomes large. Thus, we classify the flow if the features of the flow become sufficiently accurate. Otherwise we wait another packets that improve the accuracy of the features.

To achieve this, we extend the AntTree. In the AntTree, an ant, which corresponds to an item required to be classified, walks on the tree constructed by the other ants. Then, if the ant find the ant whose similarity exceeds the threshold, the ant is connected to the similar ant.

If the nearby ants do not have the similarity exceeding the threshold, the ant updates the threshold and goes to another place on the tree. By continuing this process, the hierarchical tree, where similar ants are connected, is constructed.

In this paper, we set the threshold of the AntTree based on the number of received packets; the flow with a small number of monitored packets is connected to the node only if the very similar node exists. On the other hand, the flow with a large number of monitored packets is easier to be connected.

B. Data structure

In this paper, we construct the hierarchical cluster based on ClusTree [7]. Figure 1 shows the data structure of the constructed cluster.

In this data structure, the feature values of the flows are stored in a table, and updated each time a packet corresponding the flow arrives. We denote the feature value of the flow f as the vector F_f.

This cluster has two kinds of nodes, *inner node* and *leaf node*. The leaf node indicates a fine-grained cluster, and includes l to$L$ flows. Each leaf node has the pointer to the feature values of the corresponding flows.

The inner node corresponds to the coarse-grained cluster, which includes the flows included in its children nodes. That is, by constructing the tree of inner nodes, we can hierarchically construct clusters; the inner node near root node corresponds to the coarser-grained cluster, and the node near leaf corresponds to the finer grained node.

The inner node construct the tree structure by connecting it to$m$ to M children. Each inner node has the entries corresponding to its children. Each entry has the abstracted clustering features and the pointer to the corresponding child.

 ISBN: 978-1-61208-460-2

The abstracted clustering features corresponding to the child node c have the following values.

- The number of flows included in the abstracted features n_c
- The sum of the features S^{linear}, which is calculated by

$$S_c^{\text{linear}} = \sum_{f \in F_c} F_f$$

where F_c is the set of flows included in cluster of the node c.

In our method, there are flows that have not been classified into any clusters. We maintain such flows in a list called *unclassified flows*.

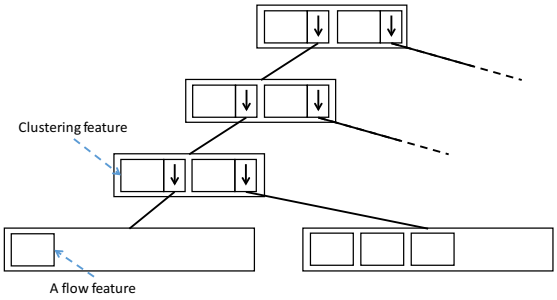


Figure 1. Data structure of the hierarchical cluster

C. Process to update the tree

When the traffic classifier receives a packet, it identifies the flow of the packet, and stores the packets. Each time a packet of the flow arrive, the features of the flow are updated. At the same time, the data structure of the tree is updated.

The process to update the tree depends on whether the flow is already classified into one of the clusters or included in the unclassified flows.

1) Update the data of the flows that is included in the unclassified flows: We denote the flow finding the cluster by f^{new}, the location of f^{new} by $p_{f^{\text{new}}}$, the node whose feature is the most similar to f^{new} among the children of $p_{f^{\text{new}}}$ by $c_{p_{f^{\text{new}}}}$. T_f^{sim} and T_f^{dissim} are the thresholds for the flow f. $Sim(f, c)$ indicates the similarity between the flow f and the node c, which is calculated by

$$Sim(f, c) = 1 - \frac{distance(f, c) - distance_{min}}{distance_{max} - distance_{min}} \quad (1)$$

where $distance(f, c)$ is defined by

$$distance(f, c) = \left| F_f + \frac{S_n^{\text{linear}}}{n_c} \right|.$$

In this paper, the flow f^{new} moves over the tree by performing the following rule once per one arrival packet.

- If $p_{f^{\text{new}}}$ is the root node
  1) if the root node has no children, make a leaf node and insert the pointer to f^{new} to the newly added node
  2) Otherwise, go to $c_{p_{f^{\text{new}}}}$
- If $p_{f^{\text{new}}}$ is not the root node
  1) If $Sim(f^{\text{new}}, c_{p_{f^{\text{new}}}}) \geq T_{f^{\text{new}}}^{\text{sim}}$,
     a) Go to $c_{p_{f^{\text{new}}}}$
     b) If $c_{p_{f^{\text{new}}}}$ is a leaf node,
        i) Insert the pointer to f^{new} to $c_{p_{f^{\text{new}}}}$
        ii) Update the abstracted clustering features of the ancestors a of $c_{p_{f^{\text{new}}}}$ by adding F_f to S_a^{linear} and 1 to n_a
        iii) If there exists a node whose number of children exceeds the upper limit (L or M), add new a node
  2) If $Sim(f^{\text{new}}, c_{p_{f^{\text{new}}}}) < T_{f^{\text{new}}}^{\text{sim}}$
     a) $Sim(f^{\text{new}}, c_{p_{f^{\text{new}}}}) < T_{f^{\text{new}}}^{\text{dissim}}$, go back to the parent node of $c_{p_{f^{\text{new}}}}$
     b) Otherwise, stay at $c_{p_{f^{\text{new}}}}$

In the above steps, $T_{f^{\text{new}}}^{\text{sim}}$ and $T_{f^{\text{new}}}^{\text{dissim}}$ is updated by

$$T_{sim}(a_i) = T_{sim}(a_i) \times \alpha_1, \quad (2)$$

and

$$T_{dissim}(a_i) = T_{dissim}(a_i) \times \alpha_2, \quad (3)$$

where α_1 and α_2 are the parameters.

Addition of the new node in the above steps is done by the following steps. First, we calculate $\sum_{c \in C_a} \frac{S_n^{\text{linear}}}{n_c}$ where a is a node whose number of entries exceeds the upper limit, and C_a is a set of the children of a. Then, we select the $c^{\max}$ whose $\frac{S_n^{\text{linear}}}{n_{c^{\max}}}$ is the most different from $\sum_{c \in C_a} \frac{S_n^{\text{linear}}}{n_c}$. Finally, we remove the entry for $c^{\max}$ from a and add the node including the entry for $c^{\max}$. The parent of the newly added node is set to the parent of a. If the parent of a also has more entries than the upper limit after the above process, we perform the same process for the parent again.

IV. Experiment

In this paper, we used our traffic classifier to classify the flows from one computer in our laboratory, where 43 flows are monitored. The computer accessed Web servers, an Exchange server, and so on through the 80 or 443 port. To classify the flows, we use the features shown in Table I. In this features, we define the downstream packets as the packets from the servers whose port number is a well-known port, and the upstream packets as the packets to the servers.

In this experiment, we set the initial values of $T_{sim}(a_i)$ and $T_{dissim}(a_i)$ to 1.0. We set α_1 and α_2 to 0.7. We started the classification after 5 packets per flow were received. The features of each flow were updated until more than 100 packets of the flow were received. We stopped updating the features after 100 packets were received, because the features do not change significantly after a sufficient number of packets are monitored.

Figure 2 shows the tree constructed by our classification. Table II shows the features of the flow grouped by the cluster in the lowest layer. Tables III and IV show the abstracted clustering features divided by the number of flows included in

 ISBN: 978-1-61208-460-2

TABLE I
FEATURES USED IN OUR EXPERIMENT

Name	Description
$E(s^{\mathrm{down}})$	Average of the size of the downstream packets
$E(s^{\mathrm{up}})$	Average of the size of the upstream packets
$\sigma(s^{\mathrm{down}})$	Standard deviation of the size of the downstream packets
$\sigma(s^{\mathrm{up}})$	Standard deviation of the size of the upstream packets
$E(i^{\mathrm{down}})$	Average of the interval of the arrival of the downstream packets
$E(i^{\mathrm{up}})$	Average of the interval of the arrival of the upstream packets
$\sigma(i^{\mathrm{down}})$	Standard deviation of the interval of the arrival of the downstream packets
$\sigma(i^{\mathrm{down}})$	Standard deviation of the interval of the arrival of the upstream packets

the abstracted clustering features. To illustrate the result of the clustering, we plot the scatter graph of the clustering features of each flow. In Figures 3 and 4, we colored the flows based on the group constructed in the 2nd layer and the 3rd layer of the tree, respectively.

These figures show that our clustering method can group flows into clusters so that the flows with the similar feature are included in the same cluster, in any layers. That is, our method can classify the flows even before the flow is completed.

These figure also indicates that the constructed clusters are mainly based on the packet sizes, and packet arrival interval does not have a large impact on the cluster.

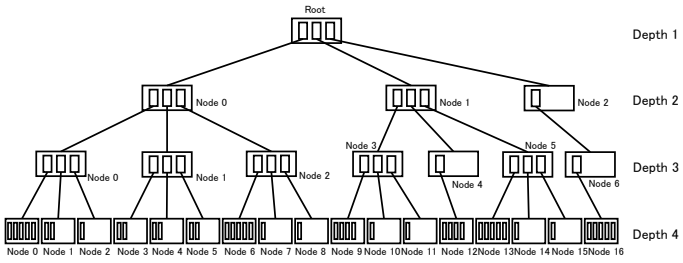


Figure 2. Tree constructed by our classification

V. CONCLUSION

In this paper, we proposed a new traffic classification method that construct hierarchical groups of the similar flows. Through the experiment, we demonstrated that our classification method enables grouping similar flows into the same clusters. That is, our method can classify the flows even before the flow is completed. Results also indicates that the constructed clusters are mainly based on the packet sizes, and packet arrival interval does not have a large impact on the cluster.

Our future work includes further verification of our method using larger traffic data and discussion on more appropriate features calculated from packet information.

REFERENCES

[1] Takashi Miyamura, Yuichi Ohsita, Shin'ichi Arakawa, Yuki Koizumi, Akeo Masuda, Kohei Shiomoto, and Masayuki Murata, "Network virtualization server for adaptive network control," in *Proceedings of 20th ITC Specialist Seminar on Network Virtualization - Concept and Performance Aspects*, May 2009.

[2] A. Callado, C. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A survey on internet traffic identification," *Communications Surveys & Tutorials, IEEE*, vol. 11, pp. 37–52, Aug. 2009.

[3] L. Popa, A. Ghodsi, and I. Stoica, "Http as the narrow waist of the future internet," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, (New York, NY, USA), pp. 6:1–6:6, ACM, 2010.

[4] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, (New York, NY, USA), pp. 135–148, ACM, 2004.

[5] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 50–60, June 2005.

[6] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the 2006 ACM CoNEXT Conference*, CoNEXT '06, (New York, NY, USA), pp. 6:1–6:12, ACM, 2006.

[7] P. Kranen, I. Assent, C. Baldauf, and T. Seidl, "The ClusTree: indexing micro-clusters for anytime stream mining," *Knowledge and Information Systems*, vol. 29, no. 2, pp. 249–272, 2011.

[8] H. Azzag, N. Monmarche, M. Slimane, and G. Venturini, "AntTree: a new model for clustering with artificial ants," *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 4, pp. 2642–2647, Dec. 2003.

[9] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, pp. 104–117, Jan 2013.

[10] T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and continuous machine-learning-based classification for interactive ip traffic," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 1880–1894, Dec. 2012.

[11] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, "Internet traffic classification by aggregating correlated naive bayes predictions," *Information Forensics and Security, IEEE Transactions on*, vol. 8, pp. 5–15, Jan 2013.

[12] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Comput. Netw.*, vol. 53, pp. 790–809, Apr. 2009.

[13] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang, "A modular machine learning system for flow-level traffic classification in large networks," *ACM Trans. Knowl. Discov. Data*, vol. 6, pp. 4:1–4:34, Mar. 2012.

[14] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pp. 281–286, 2006.

[15] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and A. Vasilakos, "An effective network traffic classification method with unknown flow detection," *Network and Service Management, IEEE Transactions on*, vol. 10, pp. 133–147, June 2013.

[16] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. Yang, "Internet traffic classification using constrained clustering," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, pp. 2932–2943, Nov 2014.

 ISBN: 978-1-61208-460-2

TABLE II
FEATURES OF THE FLOWS: DEPTH 4

flow number	$E(s^{\mathrm{down}})$	$E(s^{\mathrm{up}})$	$\sigma(s^{\mathrm{down}})$	$\sigma(s^{\mathrm{up}})$	$E(i^{\mathrm{down}})$	$E(i^{\mathrm{up}})$	$\sigma(i^{\mathrm{down}})$	$\sigma(i^{\mathrm{up}})$
Node 0								
1	0.72084	0.0982946	0.42354	0.114737	1.01073e-005	1.51213e-005	3.37173e-005	3.97892e-005
2	0.627727	0.106697	0.426203	0.126686	5.652e-005	0.000123704	9.43008e-005	0.000106826
3	0.713851	0.0930637	0.381509	0.123607	0.000609879	0.000972389	0.00136518	0.00154726
4	0.804338	0.0884975	0.378517	0.112426	2.81074e-005	6.92139e-005	6.58568e-005	8.45793e-005
5	0.763775	0.116724	0.386436	0.159844	0.0123994	0.0198522	0.0582105	0.072384
Node 1								
6	0.546043	0.101218	0.4561	0.115731	5.26767e-005	0.000108162	8.91574e-005	8.57252e-005
7	0.547945	0.104547	0.454592	0.128433	0.000902717	0.00128282	0.00165487	0.00184167
Node 2								
8	0.715821	0.144243	0.396433	0.246516	2.88237e-005	3.63364e-005	8.34268e-005	9.56236e-005
Node 3								
9	0.427973	0.293015	0.349902	0.284289	3.13866e-005	3.08145e-005	5.17207e-005	4.05901e-005
10	0.541256	0.504292	0.431807	0.441602	0.0011992	0.000964186	0.00186101	0.00159072
Node 4								
11	0.361422	0.248002	0.384193	0.309865	0.000404118	0.000285903	0.000602977	0.000354651
12	0.335109	0.277778	0.378048	0.320403	0.000174889	0.000145212	0.000162109	0.000161237
Node 5								
13	0.463263	0.20624	0.429613	0.281791	0.00074491	0.000975223	0.00158266	0.0018436
14	0.463263	0.226884	0.429613	0.292398	0.000326853	0.000513898	0.000358468	0.000505234
Node 6								
15	0.378691	0.0986175	0.412466	0.0966242	9.41781e-005	7.83116e-005	0.000343073	0.000214059
16	0.384721	0.0927376	0.397592	0.0789838	1.71589e-005	1.75337e-005	5.07459e-005	4.32654e-005
17	0.32715	0.165906	0.398255	0.151882	0.00278449	0.00313977	0.00231127	0.00203568
18	0.406562	0.102182	0.422961	0.0993938	0.000132655	0.000113818	0.000257254	0.000232465
19	0.402588	0.143075	0.437178	0.194868	6.77042e-005	0.00115388	0.00010484	0.0017407
Node 7								
20	0.274734	0.109589	0.369642	0.132467	0.000220875	0.000434846	0.000330604	0.00039426
Node 8								
21	0.479959	0.10624	0.441488	0.125773	0.0506274	0.0633271	0.101089	0.109396
Node 9								
22	0.0570776	0.0593607	0.0114155	0	0.0222335	0.0500198	0.0248326	6.13253e-006
23	0.105784	0.0410959	0	0	0.250259	0.250248	3.545e-006	6.00934e-005
24	0.105784	0.0410959	0	0	0.250257	0.250257	7.90377e-006	0.000219129
25	0.167047	0.328767	0.121385	0	0.323694	0.727734	0.36831	0.121438
Node 10								
26	0.0456621	0.0410959	0	0	0.0105705	0.00906027	0.022708	0.0213338
Node 11								
27	0.1207	0.0628615	0.0339138	0.0021309	0.00817775	0.00833702	0.00765219	0.00837687
Node 12								
28	0.293715	0.10136	0.342828	0.0946878	0.000162252	0.000128599	0.000429047	0.000293705
29	0.3431	0.128742	0.420671	0.191927	0.000136547	0.00013721	0.000158327	8.46613e-005
30	0.256722	0.0967783	0.350721	0.120472	0.00795401	0.00795477	0.00984362	0.0097643
31	0.222	0.107827	0.309717	0.0900621	0.000672249	0.000859237	0.00212733	0.00252721
Node 13								
32	0.153349	0.174458	0.190907	0.148768	2.02533e-005	2.86792e-005	5.66575e-005	6.71647e-005
33	0.102055	0.181602	0.0811775	0.176932	9.99759e-005	7.07622e-005	0.000182625	0.000142923
34	0.140665	0.171487	0.131139	0.178177	2.1e-005	1.51033e-005	2.84602e-005	1.38837e-005
35	0.150158	0.132479	0.151779	0.146517	8.32722e-005	8.31819e-005	0.000175796	0.000215069
36	0.100761	0.199391	0.107929	0.277773	0.0292866	0.023471	0.0504462	0.0461822
Node 14								
37	0.122273	0.129427	0.0958468	0.102736	0.044121	0.0536464	0.127781	0.138994
Node 15								
38	0.226636	0.0888128	0.170655	0.0909378	0.0431425	0.0196648	0.0638903	0.0455347
Node 16								
39	0.946356	0.0748792	0.208481	0.156522	3.45042e-005	0.000112505	6.80385e-005	0.000201225
40	0.756059	0.0831219	0.380303	0.112472	0.000272307	0.000536175	0.000593526	0.000963796
41	0.895826	0.0653595	0.278181	0.0857372	7.59293e-005	0.000120305	0.000233297	0.000279578
42	0.862609	0.0769847	0.324408	0.0999886	6.9086e-006	1.35787e-005	1.75598e-005	2.69238e-005
43	0.899784	0.088946	0.292179	0.123184	3.14463e-005	7.06452e-005	5.85814e-005	6.59837e-005

 ISBN: 978-1-61208-460-2

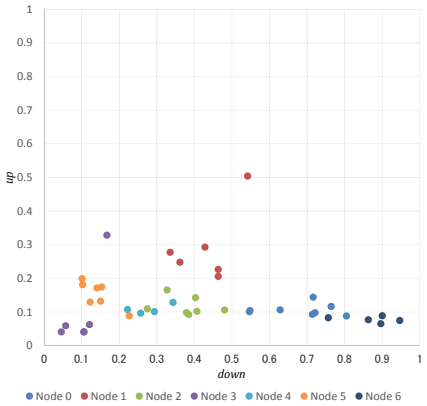
TABLE III
FEATURES OF THE FLOWS: DEPTH 3

flow number	$E(s^{\mathrm{down}})$	$E(s^{\mathrm{up}})$	$\sigma(s^{\mathrm{down}})$	$\sigma(s^{\mathrm{up}})$	$E(i^{\mathrm{down}})$	$E(i^{\mathrm{up}})$	$\sigma(i^{\mathrm{down}})$	$\sigma(i^{\mathrm{up}})$
Node 0								
1-5	0.726106	0.100655	0.399241	0.12746	0.0026208	0.00420653	0.0119539	0.0148325
6-7	0.546994	0.102882	0.455346	0.122082	0.000477697	0.00069549	0.000872014	0.000963698
8	0.715821	0.144243	0.396433	0.246516	2.88237e-005	3.63364e-005	8.34268e-005	9.56236e-005
Node 1								
9-10	0.484614	0.398653	0.390854	0.362946	0.000615291	0.0004975	0.000956365	0.000815653
11-12	0.348266	0.26289	0.38112	0.315134	0.000289504	0.000215557	0.000382543	0.000257944
13-14	0.463263	0.216562	0.429613	0.287094	0.000535882	0.00074456	0.000970564	0.00117442
Node 2								
15-19	0.379942	0.120503	0.41369	0.12435	0.000619237	0.000900662	0.000613436	0.000853234
20	0.274734	0.109589	0.369642	0.132467	0.000220875	0.000434846	0.000330604	0.00039426
21	0.479959	0.10624	0.441488	0.125773	0.0506274	0.0633271	0.101089	0.109396
Node 3								
22-25	0.108923	0.11758	0.0332002	0	0.211611	0.319565	0.0982885	0.0304309
26	0.0456621	0.0410959	0	0	0.0105705	0.00906027	0.022708	0.0213338
27	0.1207	0.0628615	0.0339138	0.0021309	0.00817775	0.00833702	0.00765219	0.00837687
Node 4								
28-31	0.278884	0.108677	0.355984	0.124287	0.00223127	0.00226995	0.00313958	0.00316747
Node 5								
32-36	0.129397	0.171883	0.132586	0.185633	0.00590221	0.00473374	0.0101779	0.00932424
37	0.122273	0.129427	0.0958468	0.102736	0.044121	0.0536464	0.127781	0.138994
38	0.226636	0.0888128	0.170655	0.0909378	0.0431425	0.0196648	0.0638903	0.0455347
Node 6								
39-43	0.872127	0.0778583	0.29671	0.115581	8.42191e-005	0.000170642	0.0001942	0.000307501

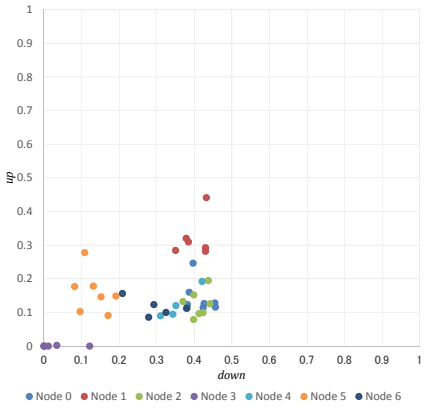
TABLE IV
FEATURES OF THE FLOWS: DEPTH 2

flow number	$E(s^{\mathrm{down}})$	$E(s^{\mathrm{up}})$	$\sigma(s^{\mathrm{down}})$	$\sigma(s^{\mathrm{up}})$	$E(i^{\mathrm{down}})$	$E(i^{\mathrm{up}})$	$\sigma(i^{\mathrm{down}})$	$\sigma(i^{\mathrm{up}})$
Node 0								
1-8	0.680042	0.106661	0.412916	0.140997	0.00176102	0.0028075	0.00769962	0.00952319
9-14	0.432048	0.292702	0.400529	0.321725	0.000480226	0.000485873	0.000769824	0.000749338
15-21	0.379201	0.116907	0.411369	0.125713	0.00770635	0.00975218	0.0149268	0.0162938
Node 1								
22-27	0.100342	0.0957128	0.0277857	0.00035515	0.144199	0.215943	0.0705857	0.025239
28-31	0.278884	0.108677	0.355984	0.124287	0.00223127	0.00226995	0.00313958	0.00316747
32-38	0.142271	0.153951	0.132776	0.160263	0.0166821	0.0138543	0.0346516	0.0330214
Node 2								
39-43	0.872127	0.0778583	0.29671	0.115581	8.42191e-005	0.000170642	0.0001942	0.000307501

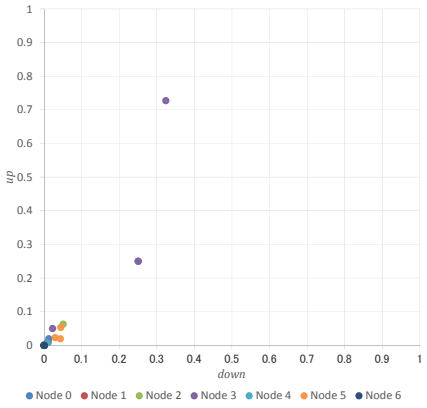
 ISBN: 978-1-61208-460-2



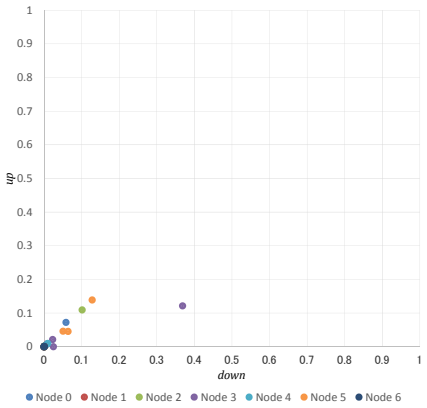
(a) $E(s^{\mathrm{up}})$ vs. $E(s^{\mathrm{down}})$



(b) $\sigma(s^{\mathrm{up}})$ vs. $\sigma(s^{\mathrm{down}})$

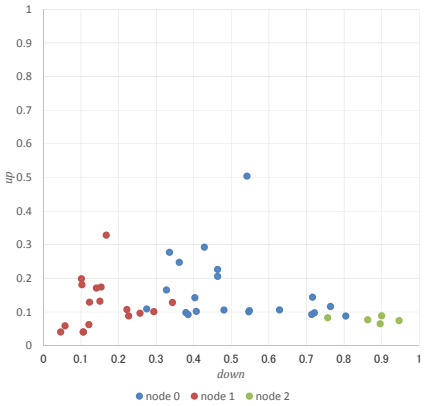


(c) $E(i^{\mathrm{up}})$ vs. $E(i^{\mathrm{down}})$

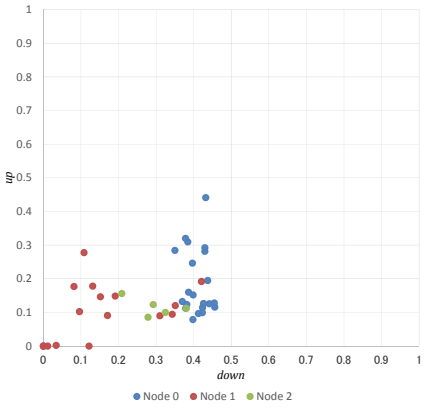


(d) $\sigma(i^{\mathrm{up}})$ vs. $\sigma(i^{\mathrm{down}})$

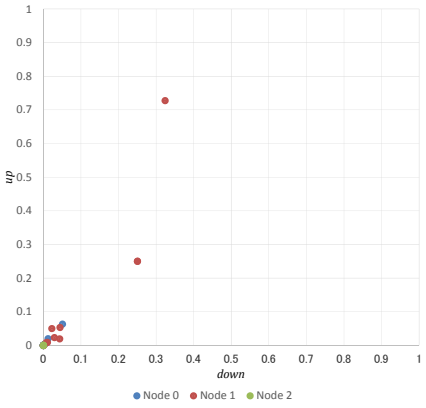
Figure 3. Features distribution: depth 3



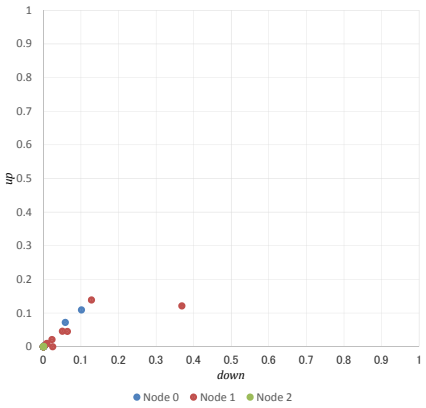
(a) $E(s^{\mathrm{up}})$ vs. $E(s^{\mathrm{down}})$



(b) $\sigma(s^{\mathrm{up}})$ vs. $\sigma(s^{\mathrm{down}})$



(c) $E(i^{\mathrm{up}})$ vs. $E(i^{\mathrm{down}})$



(d) $\sigma(i^{\mathrm{up}})$ vs. $\sigma(i^{\mathrm{down}})$

Figure 4. Features disribution: depth 2

 ISBN: 978-1-61208-460-2

Comparing Replication Strategies for Financial Data on Openstack based Private Cloud

Deepak Bajpai, Ruppa K. Thulasiram
Computer Science department
University of Manitoba
Winnipeg, Canada
Email: {bajpaid, tulsi}@cs.umanitoba.ca

***Abstract*—Private Cloud has substituted the traditional infrastructure due to its flexible model and privacy in the form of administration control and supervision. In this study, we have built a private Cloud using openstack Cloud based open-source software solution. We deployed financial application on VMs and generated a disaster recovery solution using openstack component Cinder and Swift. We presented an experimental analysis of two strategies, block storage and object storage, to derive the best solution for an organization using private Cloud. From the set of experiments considered Cinder proved preferable over Swift.**

Keywords- Cloud Computing, Private Cloud, Replication, Disaster Recovery, Block Storage, Object Storage

I. INTRODUCTION

With the exponential growth of Information Technology (IT) Infrastructure and increasing cost of IT from small scale to high end enterprise sectors, a workaround to effectively reduce the cost associated with infrastructure has become essential. Also, as the uncontrolled data growth raised concerns for the enterprise environment, storage moved outside the servers to individual identity that provided a significant solution [1]. Integration of Virtual Machines (VM) in the virtualization layer to the operating system environment added a robust and effective measure to share the CPU load and processing time [2]. With more advancement in technology, more cost effective solutions were required. It was this time when Cloud computing came into emergence.

Cloud computing is a major shift on how the information is stored and shared on the distributed platform to provide services to customer as per customer demand. Cloud has been defined as ”pay-as-you-go” architecture where customer pays for the resources when used. Introduction of Cloud service by Amazon, Microsoft and other Cloud service providers induced leading technology hubs like Oracle, HP, IBM, Adobe, etc., to expand their service areas for Cloud technology. When it comes to financial, health and defense services, public Clouds have several vulnerabilities, including security and privacy issues, and this is where private Cloud comes into the picture. Private Cloud has shown reliable solutions for these components of business and hence many organizations have jumped into private Cloud [3]. A private Cloud is more independent as the organisation build their own infrastructure, use their own data storage blocks and servers, and at times private Cloud owners may also outsource their requirement to the third party.

In public Clouds, the service provider has control of the IT infrastructure and, eventually, they control customers sensitive data which reside in their datacenter. Even though there could be regulatory procedures (such as Service Level Agreement) in place that ensures fair management and supervision of the customers privacy, this condition can still be perceived as a threat or as an unacceptable risk that some organizations are not willing to take. In particular, institutions such as government and military agencies will not consider public Clouds as an option for processing or storing their sensitive data [4]. When we put financial application and operations in public Cloud then there is severe threat of vulnerability as all the data is stored with third party Cloud vendor. To address this issue we built private Cloud and deployed financial application to provide security and privacy.

The main problem in the market of Cloud services is implementation and performance evaluation for the critical data. IT infrastructure organization has a central data repository that contains all the important data required for the proper functioning of the organization. This data can be classified as critical on the basis of usage by the users. In a Cloud environment, data remains at different geo-locations; VMs at different geo-locations hold that data which can be migrated from one host to another host as per the requirement of load balancing [5].

Some of the replication techniques which are used by market leaders are object storage or block storage replication. A cost effective dynamic replication management (CDRM) scheme has been introduced for storage clusters in Cloud [6] which has used Hadoop data filesystem. However, in CDRM, filesystem should be in a mounted state for initiating the replication. Various object storage replication techniques have been in market [7] and they are efficient, but for block storage replication in Cloud storage cluster, there is still demand for cutting edge strategy.

In this paper, we have proposed a new replication strategy for the duplication of data in openstack based private Cloud. In this strategy, we created the VM using the block storage component of openstack called Cinder. After the testing of VM on a financial application, we created snapshots and used them for later recovery process. This strategy is new, and we

 ISBN: 978-1-61208-460-2

show that this is 53% better in time than object storage based replication technique. Rest of the paper describes the attributes of openstack, the deployment of private Cloud and the empirical analysis of the replication strategy all in the related work section. Implementation and disaster recovery mechanism are described in Section III. We did experimental comparison with object based replication using Swift component of openstack as described in Section IV. After comparing these action plans, we have analyzed the results on the basis of best strategy available in Section V. We present our conclusions in Section VI. One of the financial applications we have considered is option pricing. Note that due to space limitation we are not describing the financial option pricing application and the data related to this application, brief description on which are available in [2].

II. Related Work

There are multiple software sources available to build a Cloud. CloudStack is an open source software that allows for a Private Cloud and Hybrid Cloud deployment [8]. The main functionality of CloudStack is to operate a large scale deployment of a virtual infrastructure by managing a large number of virtual machines. Eucalyptus is an open source software used to build Private Clouds and Hybrid Clouds that are compatible with Amazon Web Services [9]. Eucalyptus manages resources that allow for some dynamic allocation of resources. OpenNebula is an open source software that allows for the deployment of Public Clouds, Private Clouds, and Hybrid Clouds [10]. The main functionality of OpenNebula is to manage data that is distributed among datacenters and to ensure that these datacenters are able to exchange information with each other regardless of their infrastructure. OpenStack is an open source software that is primarily used to manage a virtual infrastructure using the Dashboard or the OpenStack API. What makes OpenStack advantageous compared to the other options is that OpenStack supports small scale deployment. OpenStack supports deployment onto a single, local machine for rapid application development and testing with minimal setup required. For this reason, OpenStack was selected in this study as the Cloud computing software to develop the disaster recovery application.

A. Openstack

Originally OpenStack was a project which was developed by Rackspace and NASA [11]. In 2010, OpenStack was released under the name Austin. Austin initially had very limited features. At the time Austin had support for object storage only. Companies started to contribute to OpenStack because they began to see its potential in the virtualization market. As contributors added more features to OpenStack, eventually in 2012 support for Cinder was integrated [3]. OpenStack provides a means for small companies to provide their customers with services without the need of a significant investment initially. It became easier for new service providers to start out who did not have a lot of investors. At the same time, OpenStack provided a means for big organizations to implement private clouds within their corporation without the need of big investments into physical hardware. OpenStack is currently an open source project and has hundreds of contributors. What makes OpenStack so advantageous is that if there is a need for a feature there are contributors constantly available to implement them and fix them if any bugs are discovered. Another reason OpenStack is advantageous is that it is an open source project and it does not require any subscription or an annual fee to use.

All of the modules contribute various components to OpenStack. Of the nine modules listed in Figure.1, only three of those modules provide storage mechanisms for OpenStack. Each of these three modules provide a different storage mechanism. The three modules are, Swift, Cinder, and Glance. Swift provides an object storage capability, Cinder provides a block storage capability, and Glance provides a repository to store the virtual machines a user creates in OpenStack or downloads them from the internet.

1) Cinder: Block storage was a fundamental milestone for Cloud computing because it provides the capability to store virtual machines along with the data those virtual machines use. Before block storage was integrated into OpenStack virtual machines used something called ephemeral storage [11]. Virtual machines using ephemeral storage have a significant, fundamental flaw. The flaw is that when the virtual machine powers down, all of the data and contents of the virtual machine are lost. This is a problem because a virtual machine may contain important data required to provide services to customers. Say the service being provided is banking transactions. If the virtual machine powers down, all of the customers banking information is lost. The organization wont know what the last balance on the customers account was if that information wasnt backed up. Simply powering on the virtual machine wouldnt solve the problem because even though we have the environment to provide the service, we need the data of past transactions. Finding a solution to ensure a virtual machine never powers down is not a realistic answer to the problem.

There was a need to provide a mechanism that allows data to persist in the event a virtual machine powers down. Block storage provides one possible solution to this problem. The data persists within the volume even though the virtual machine may power down and the data is available the next time the virtual machine powers up. Cinder was developed and integrated into OpenStack to provide access to and management of a block storage created by a user.

2) Swift: In OpenStack, the module named Swift provides the object storage component in the application. Swift is used to implement something called an OpenStack cluster. The OpenStack cluster is used to provide a distributed object store. The object store uses HTTP protocols PUT to update an object or GET to retrieve the most recent version of the object. Swift implements something called an OpenStack ring. This ring consists of a proxy server and a storage node. These two

 ISBN: 978-1-61208-460-2

components are used to store and retrieve objects upon a users request. The main functionality of the proxy server is to track the location of the most recent version of the object and to determine which storage server it should send the most recent version of the object. These OpenStack rings can be divided in various ways. The reason for the division is if in the event a request for an object fails, then there is another entity available who can fulfill this request. Four common divisions of OpenStack rings are by disk drive, server, zone, or region.

Within each of these divisions the data is replicated in the event one of the providers has a failure so another entity can fulfill the request. Through various testing it was determined maintaining three replicas was sufficient for reliability so by default OpenStack maintains three replicas of the data [3]. Swift implements eventual consistency. This means that when an object is updated not all of the storage servers are updated immediately to contain the most recent version of the object. Only one of the storage servers receives the most recent version of the object from the proxy server and then the storage server propagates that object to the rest of the storage servers as a background task. A risk with eventual consistency is that a storage server may receive an updated object but then may fail before it has a chance to propagate that updated object to another storage server. In a future updated versions of OpenStack, code named Grizzly, allowed users to maintain as many replicas as they wanted within the OpenStack ring. Another feature implemented was time based sorting where the proxy server would request the most recently updated object from the fastest responding storage server. The main reason time based sorting was implemented was because storage servers were beginning to be distributed over larger areas, for example, the other side of the country. A major advantage Swift provided was that it allows the capability to store objects on different platforms for example, Cleversafe, Scality, and Amazon S3[5].

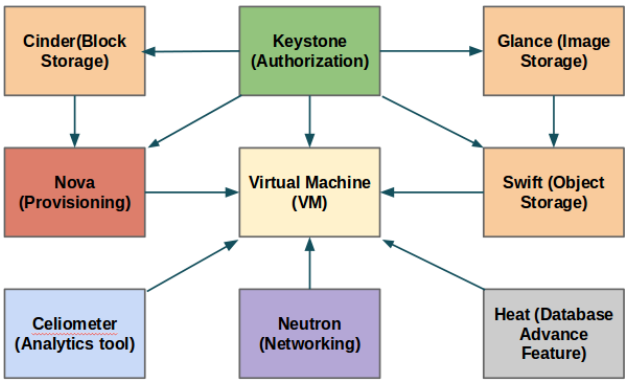


Figure. 1. Internal component connection in openstack.

III. Disaster Recovery and Implementation

”Disaster recovery (DR) is defined as the use of alternative network circuits to re-establish communications channels in the event that the primary channels are disconnected or malfunctioning, and The methods and procedures for returning a data center to full operation after a catastrophic interruption (e.g., including recovery of lost data). Disaster recovery involves in restoring the technology aspect of a service. In 1978, Sun Information Systems was the first American company to provide a hot site to organizations [12]. A hot site is a facility that an organization can relocate to in the event of a system failure due to a natural disaster or a human induced disaster to resume the operation of a service.

As businesses became more reliant on their IT infrastructure during the late 20th century, there was pressure for organizations to have a disaster recovery plan in place in the event a service becomes unavailable. As a result different mechanisms were being provided as solutions. Data centers were being built because on site recovery was no longer necessary due to the development of the internet. It has been statistically proven that spending one dollar on disaster recovery planning will save you four dollars in the long run when trying to recovery from a disaster[12]. An important step in disaster recovery planning is to identify which services are considered vital for an organization, how long can they afford to keep the service unavailable and identify which systems are associated with providing the service [12].

There are multiple strategies in designing an efficient disaster recovery plan, here are a few strategies. Backups of systems can be made and transported to an offsite location. A second alternative is to only forward data to an offsite location instead of copying the entire system. A third alternative is to use a Private Cloud to replicate virtual machines, disks, templates and store them in the Private Cloud. A fourth alternative is to use a Hybrid Cloud to back up the data stored onsite and offsite and in the event of a disaster launch the system from within the Cloud. A fifth alternative is to backup both the system and the data at an offsite location. For the purposes of this current study, a backup will be made of both the state of the system and the data it contains.

Implementation of private Cloud is a complex and cumbersome process. Its architecture is same as distributed system but the association of various components to make it service oriented architecture brings the complexity in design. After the implementation it is important to test the performance on the basis of various performance parameters such as response time, replication status, data integrity, accessibility on the data critical and time constraint application. The most suitable application are financial application where data is critical and time is also a big constraint. We built Openstack private cloud on three node architecture as shown in Figure 2. Below are the description of each nodes.

1) Controller node- Controller node is responsible for running the basic openstack services required for private Cloud environment to function. These nodes: (a) Access to API which is accessible by user for various functionality. It is also

 ISBN: 978-1-61208-460-2

entry point for the accessibility; (b) Run numerous services in high availability, utilizing components such as Pacemaker and HAProxy to provide a load-balancing and virtual server allocation functions so the controller node is being used; (c) Provide highly available ”infrastructure” services, such as MySQL and RabbitMQ that combine all the services; (d) Provide what is known as ”persistent storage” through services runs on the host as well. This persistent storage is backed onto the storage nodes for reliability.

2) Compute Node- Compute nodes host the virtual machine instances in OpenStack environment. They: (a) Run the minimum of services required to run these instances; (b) Use local defined storage on the nodes for the VM so that disaster recovery in case of node failure is possible.

3) Network Node- Network nodes provide communication channel and the virtual networking needed for users to create public or private networks. It also provide uplink to their virtual machines into external networks. (a) Form the only ingress and egress point for access and security feature of the Openstack running instance; (b) Run the environment's networking services other than networking API service. Virtual

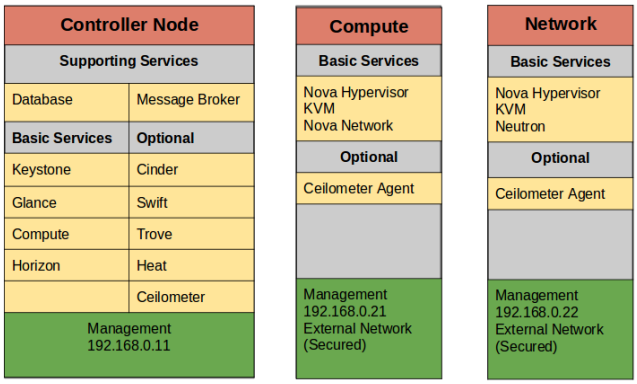


Figure. 2. Architecture of Openstack based Cloud.

machine provisioning is an important step for the private Cloud. Following are the basic components required for the VM provisioning:1) Block Storage (Cinder)- Provides persistence block storage for running VMs and create VMs as well using images, 2) Object Storage (Swift)-Stores and retrieve arbitrary unstructured data objects via a RESTful, Http based API, 3) Network (Neutron)-Provide communication channel as a service for Openstack services, such as Openstack compute, 4) Authorization(Keystone)-Provides an authorization and authentication service for all Openstack services, 5) VM Image (Glance)-Stores and retrieve VM disk images, 6) Provisioning (Nova-Compute, KVM hypervisor integrated), 7) Controller provisioning (Nova-Controller). Openstack provide all these as separate components as shown in Figure 1. These components help to establish services within the framework. The VMs on the compute nodes are working on arbitrary computation algorithms which is be memory intensive. After that, we used snapshot based technology to capture the state of VM image and these snapshots are scheduled to be triggered in a timely manner [13]. Along with the snapshots, block replication is initiated on the VMs, which captures the raw data on the data blocks and store them on other passive systems. These data blocks can be retrieved in case of disaster and which will provide same data as an active node. With snapshot and synchronous replication, data will be more secure.

IV. Experiments

To build a private Cloud, we used Dell R420 servers with multiple Ethernet ports. All servers have 4GB RAM and 8 Intel Xeon processors on each of them. Ubuntu 14.04 server was used as the operating system running on each of them. After the setup of virtual machines, IP association is done and testing for the connection is done using Address resolution protocol (ARP). Java runtime environment and MYSQL are deployed on all the nodes for multiple operations. After the setup of all three nodes using Openstack with VMs, we run the arbitrary computation tasks on the multiple VMs hosted on the compute node. The snapshots are captured as per pre-defined schedule using scripts. After that extensive testing is performed on financial application, where one of the VMs are triggered with configuration errors that is followed by powering off that VM. At that point, snapshot are retrieved for that VM and passive data block are activated. This produce a replicated system of powered off VM. By using this method, we are able to test the reliability of time constraint data and data critical application.

In our disaster recovery experiments, we checked if the data can be retrieved by using snapshot and volume replication technique then it will generate the model for disaster recovery within the Cloud. On the basis of data retrieval following the VM failures, we also analysed the best replication strategy that can be used in Cloud environment. A typical comparison of snapshot and volume replication will be part of the evaluation process that will yield best disaster recovery solution. The baseline used to check the best replication strategy would be synchronous data recovery.

There were four VMs deployed in the Cloud. Three of them were virtual machines instances and the fourth system was an instance being launched using a volume block created containing the Ubuntu image. Of the three virtual machine instances deployed, two of them were clients and the third was a server. The fourth instance was used as a server as well, but deployed using a volume block. Cinder would be used to take a snapshot of the instance deployed from the volume block and Swift would be used to take a snapshot of the server virtual machine instance. After the systems were deployed, a simple RMI application was developed and deployed on the instances to ensure they could communicate and transfer data between each other.

We created multiple shell scripts for automation of snapshot and recovery process. These scripts were helpful to generate timely triggers to create and delete snapshots. We also created script for meaningful data generation on Linux system. Scripts that take snapshots of each instance at timed intervals and

 ISBN: 978-1-61208-460-2

maintain a specified number of backups were needed to perform the experiment. For object storage, there were three scripts created for taking a snapshot of the instance. The first script would maintain the iterations, decide which instance to target for the snapshot, and finally launch the other two scripts over every iteration which performed the actual snapshot. The second script would first delete any old snapshots exceeding their lifespan and then create a new snapshot. The third script was used to force the data to be written into memory and prevent the instance from accepting input while the snapshot is taken to prevent any data inconsistencies as this was documented as best practice in the OpenStack documentation [14]. For taking a snapshot using block storage, there were two scripts associated with the process.

Again, one script was used to maintain the iterations, which volume to target, and to launch the other script at timed intervals. The second script would delete old snapshots exceeding their lifespan and then create a new snapshot of the volume. Generating data for the experiment manually would have been a time consuming process. For this reason a script was developed in order to create text files with readable information within in order to have sample data during each iteration. Due to the time it took to create the script for generating the data, a snapshot was taken every two hours instead of every three hours. Here is a tabular form of the data collected during the experiment.

TABLE I
VM ACCESS AND PREPARATION

Iteration	Test-Mem	Mem-G	ST	VT	SD	VD
I-1	100 MB	105.4 MB	108	1	143	13
I-2	200 MB	207.6 MB	128	2	147	13
I-3	300 MB	309.8 MB	151	2	150	13
I-4	400 MB	411.0 MB	170	3	153	13
I-5	500 MB	513.2 MB	193	3	155	13

Table-1 is used to determine which methodology, object storage or block storage, is faster in terms of the time it takes to create a snapshot, the time it takes to deploy a snapshot and the amount of data associated with each iteration. Here Test-Mem denotes Test memory used as financial data block size, Mem-G denotes memory generated after taking snapshot, ST refers to time taken to create VM snaphsot(in seconds), VT is time taken to create Volume snapshot(in seconds), SD and VT are the time taken to deploy Vm and volume snapshot respectively.

TABLE II
VM RECOVERY DATA

Iteration	Tot-Bytes	V-Snap	Lost-Vm	V-vm	Lost-vol
I-1	10538598	10538598	0	10538598	0
I-2	20762214	20762214	0	20762214	0
I-3	25480806	25480806	0	25480806	0
I-4	15257190	15257190	0	15257190	0
I-5	26929875	26929875	0	26929875	0

Table-2 is used to determine which methodology, object storage or block storage, is better at preserving consistent data by measuring how many bytes of data are lost at each iteration.In this table, V-snap refers to bytes in Vm snapshot, Lost-Vm denotes bytes lost in Vm snapshot after deploying as new Vm, V-Vm refers to bytes in volume snapshot and Lost-V as bytes lost after deployment of new Vm using volume snapshot.

V. RESULTS

Analyzing the results from Figure 3. we can see that in terms of creating a backup, and deploying the backup, block storage (Cinder) is much faster than object storage (Swift). For an organization, this is a significant factor to consider when creating a disaster recovery plan because we want to make backups quickly to minimize overhead on the system so there is minimal interference with service quality. The system can be recovered a lot quicker using Cinder instead of Swift which means the system will be down for shorter periods of time. After some statistical calculations it was determined that using Cinder to deploy a backup is approximately 53 times faster than deploying a backup using Swift. As more data is stored on these systems we can expect the time to create a backup and backup deployment to increase. Let us assume the ratio calculated holds for various sizes of data. If it takes one minute to deploy a backup using block storage, then we can expect a deployment using object storage to take approximately 53 minutes. This is a significant difference and having a service down for approximately an hour may not be acceptable to an organization who relies heavily on their IT infrastructure. In terms of recovering a system from a disaster, it appears using block storage is the better option.

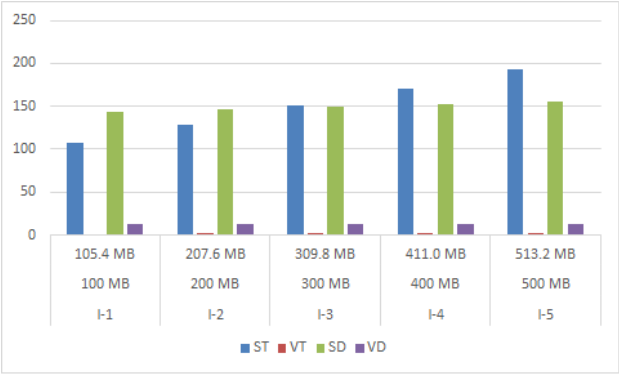


Figure. 3. VM preparation analysis

Analyzing the results in Figure 4. we can observe that both components, Cinder and Swift, experienced no data loss. As the amount of data increases on each system, it would be expected Swift and Cinder would begin to experience some data loss. Therefore we can conclude, based on the results, that both Cinder and Swift are effective in preserving small sets of data. In future experiments, bigger sizes of data would be used to test how much data we can have before we experience loss from using Cinder and Swift. We would expect to lose data when more data is stored on the system when using Swift or

 ISBN: 978-1-61208-460-2

Cinder but the goal would be to determine which component provides more consistent data when it gets large. In terms of providing consistent data, for this experiment, both Swift and Cinder performed equally well.

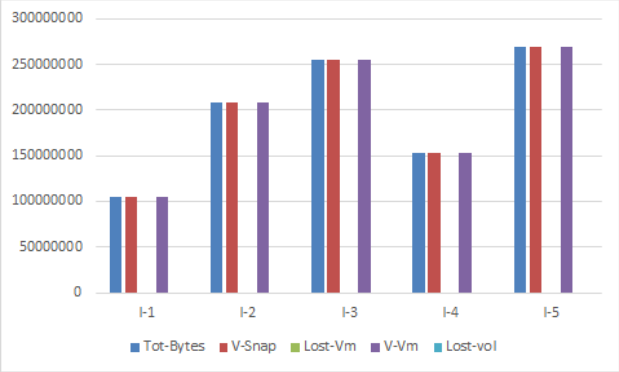


Figure. 4. VM data recovery analysis

Based on the results from Table 1 and 2, we can conclude that if our main objective of a disaster recovery plan is to restore service as quickly as possible then Cinder would be the correct path to go down. We can also conclude that both components function with equal reliability if our system is small. At the beginning all systems begin relatively small therefore if we have 2 options with both providing equal reliability and Cinder being 53 times faster than Swift, then Cinder would be the optimal choice at that particular point in time. As the system grows, more experiments could be performed to help determine which provides more consistent data as the size of the data grows and determine the trade-offs if any.

VI. Conclusion

This paper shows how private cloud built on Openstack platform was used to identify new unique strategy of volume replication. It provides efficient storage mechanism in comparison to object based replication. In consideration with strategies used in recent times, this study can help derive an effective replication solution for openstack private cloud. Given the following mechanism, we can implement similar strategies in different private Cloud platforms such as Cloudstack, OpenNebula and Eucalyptus. We can see some limitations in public clouds as datacenters can be situated in different geo-locations which can add to hop count wait time and produce delay in replication.

In conclusion, the experiment was a success for the reason being that we wanted to find which method, block storage or object storage, provided the best disaster recovery plan. In this particular environment with a relatively small dataset Cinder was proven to be the preferable choice as it held no additional advantage over Swift in terms of data consistency but it had a significant advantage in terms of the time it takes to create a backup and deploy it. Overall, we have achieved both goals for this study. We were able to build an environment for the purposes of deploying a Private Cloud, deploying a financial application utilizing RMI, and creating scripts for the purpose of managing backups of systems deployed in the Private Cloud. This work can be explored in future for the other private Clouds and Inter-Clouds. This strategy can also be used on Bigdata, which can produce interesting result.

Acknowledgment

The first author acknowledges graduate enhancement of tri-council stipends (GETS) from the Faculty of Graduate Studies, University of Manitoba. The second author acknowledges Natural Sciences and Engineering Research Council (NSERC) Canada and University of Manitoba for partial financial support for this research through Discovery Grant and University Research Grant Programs.

References

[1] M. Mesnier, G. R. Ganger, and E. Riedel, "Object-based storage," *Communications Magazine, IEEE*, vol. 41, no. 8, pp. 84–90, 2003.

[2] A. N. Toosi, R. K. Thulasiram, and R. Buyya, "Financial option market model for federated cloud environments," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2012, pp. 3–12.

[3] G. Suciu, E. G. Ularu, and R. Craciunescu, "Public versus private cloud adoptiona case study based on open source cloud platforms," in *Telecommunications Forum (TELFOR), 2012 20th*. IEEE, 2012, pp. 494–497.

[4] S. Srirama, O. Batrashev, and E. Vainikko, "Scicloud: scientific computing on the cloud," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2010, pp. 579–580.

[5] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on*. IEEE, 2010, pp. 89–96.

[6] Q. Wei, B. Veeravalli, B. Gong, L. Zeng, and D. Feng, "Cdrm: A cost-effective dynamic replication management scheme for cloud storage cluster," in *Cluster Computing (CLUSTER), 2010 IEEE International Conference on*. IEEE, 2010, pp. 188–196.

[7] M. J. Brim, D. A. Dillow, S. Oral, B. W. Settlemyer, and F. Wang, "Asynchronous object storage with qos for scientific and commercial big data," in *Proceedings of the 8th Parallel Data Storage Workshop*. ACM, 2013, pp. 7–13.

[8] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *Internet computing, IEEE*, vol. 13, no. 5, pp. 14–22, 2009.

[9] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*. IEEE, 2009, pp. 124–131.

[10] D. Milojičić, I. M. Llorente, and R. S. Montero, "Opennebula: A cloud management tool," *IEEE Internet Computing*, no. 2, pp. 11–14, 2011.

[11] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.

[12] V. Chang, "Towards a big data system disaster recovery in a private cloud," *Ad Hoc Networks*, vol. 35, pp. 65–82, 2015.

[13] V. Padhye and A. Tripathi, "Scalable transaction management with snapshot isolation on cloud data management systems," in *Cloud computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 542–549.

[14] K. Pepple, *Deploying openstack*. " O'Reilly Media, Inc.", 2011.

 ISBN: 978-1-61208-460-2