CLOUD COMPUTING 2014

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

ISBN: 978-1-61208-338-4

May 25 - 29, 2014

Venice, Italy

CLOUD COMPUTING 2014 Editors

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

Yong Woo Lee, University of Seoul, Korea

Christoph Reich, Furtwangen University, Germany

CLOUD COMPUTING 2014

Foreword

The Fifth International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2014), held between May 25-29, 2014 in Venice, Italy, was intended as an event 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.

We take here the opportunity to warmly thank all the members of the CLOUD COMPUTING 2014 Technical Program Committee, as well as all of the 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 who dedicated much of their time and efforts to contribute to CLOUD COMPUTING 2014. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the CLOUD COMPUTING 2014 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that CLOUD COMPUTING 2014 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the areas of cloud computing, GRIDs and virtualization.

We are convinced that the participants found the event useful and communications very open. We hope that Venice, Italy, provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city.

CLOUD COMPUTING 2014 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
Aaron McConnell, University of Ulster, UK
Christoph Reich, Furtwangen University, Germany
Wolfgang Gentzsch, The UberCloud, Germany
Tony Shan, Keane Inc., USA
Donglin Xia, Microsoft Corporation, USA
Laura Moore, SAP, UK
Anna Schwanengel, Siemens AG, Germany
Atsuji Sekiguchi, Fujitsu Laboratories Ltd., Japan
Aljosa Pasic, ATOS Research, Spain
Chih-Cheng Hung, Southern Polytechnic State University - Marietta, USA
Jorge Ejarque, Barcelona Supercomputing Center, Spain
Javier Diaz, Indiana University, USA
Nam Beng Tan, Nanyang Polytechnic, Singapore
Ivan Rodero, Rutgers the State University of New Jersey/NSF Center for Autonomic Computing, USA

Hong Zhu, Oxford Brookes University, UK
Qi Yu, Rochester Institute of Technology, USA
Arden Agopyan, ClouadArena, Turkey
Dariusz Król, Academic Computer Center CYFRONET - Cracow, Poland

CLOUD COMPUTING 2014

Committee

Table of Contents

Data Security in Cloud Storage Services

Mai Mansour Dahshan
Department of Computer Science and Engineering
The American University in Cairo
Cairo, Egypt
mdahshan@aucegypt.edu

Sherif ElKassass
Department of Computer Science and Engineering
The American University in Cairo
Cairo, Egypt
sherif@aucegypt.edu

***Abstract*— Cloud storage services have changed the way used to manage and interact with data outsourced to public domains. With these services, multiple subscribers can collaboratively work and share outsourced data without any concerns about their data consistency, availability and reliability. Examples of these services include Dropbox, Box.net, UbuntuOne or JungleDisk. Although these cloud storage services offer seductive features, many customers are not rushing to move their data into these services. Since data stored in these services is under the control of service providers which makes it more susceptible to security risks. Therefore, using cloud storage services for storing users data depends mainly on whether it is sufficiently secure or not. From the way cloud storage services are constructed, we can notice that these storage services don't provide users with sufficient levels of security leading to an inherent risk on users' data from external and internal attacks. To deal with these security problems, this paper proposes a novel data sharing mechanism that simultaneously achieves data confidentiality, fine-grained access control on encrypted data and user revocation by combining ciphertext policy attribute-based encryption (CP-ABE), and proxy re-encryption (PRE).**

Keywords-Secure Storage; Cloud Computing; Proxy Re-encryption; Ciphertext Policy Attribute Based Encryption.

I. INTRODUCTION

Cloud storage is a newly developed concept in the field of cloud computation. It can be defined as a system that is composed of cluster, grid and distributed file systems that using application software coordinates a variety of different type's storage devices together to provide data storage and access service. Cloud storage allows users to outsource their data that has been managed internally within the organization or by individual users to the cloud. By doing so, users eliminate the concerns associated with the installation of the complex underlying hardware, save increasing high cost in data center management and alleviate the responsibilities of its maintenance [1]. Although cloud storage services are offering this number of benefits, they are facing many challenges for securing data in public clouds, which are generally beyond the same trusted domain as data owners.

Challenges associated with cloud storage services are unique ones because all of the involved entities (i.e., Cloud Service Provider (CSP) as Dropbox and subscribers seeking access to the outsourced data) can behave maliciously. CSPs, which provision the outsourced data, can assist unauthorized subscribers to gain illegal access to data or learn about user's confidential information leading to potential loss of privacy. On the other hand, subscribers of CSPs can utilize data sharing and collaborative functionalities of a cloud storage service, complimented with malicious intent of CSP, to compromise privacy of the outsourced data. In addition, most of these subscribers are unaware about the security measures adopted by a CSP, how often they are evaluated, and how well these security measures conform to standards and government regulations [2][3][4].

Importing users' data into cloud storage services (Dropbox, box, SpiderOak, etc.) can face at least one of the following threats. First, service operators can steal users' data and credentials because they have the capacity and the authority to get access to users' data easily. Second, they can authorize other users to access users' data. Last but not least, unlike data stored in users' personal computer, the data stored in the cloud is not isolated from other people’s data. Therefore, the risk of data being attacked by the cloud increases.

The rest of the paper is organized as follows: Section 2 gives an insight into the problem. Section 3 reviews some related work that has been done to solve the addressed problem in cloud security. Section 4 presents our solution. The paper is concluded in Section 5.

II. PROBLEM STATMENT

A recent security flaw in the Dropbox authentication mechanism [5] begins the debate about whether cloud storage services are sufficiently secure to store sensitive data or not. A recent research [6] about Dropbox has shown that it suffers from three types of attacks which are hash value manipulation attack, stolen host id attack and direct download attack. Moreover, another cloud storage service as Box may not encrypt user files via Secure Sockets Layer (SSL) during transfer to/from Box and may not encrypt data within Box servers [7]. Even in the more secure storage service, SpiderOak, user's data is encrypted with his own private encryption key and his password which can make it inaccessible in case of password loss [8]. Furthermore, Hu et al. [9] evaluated four cloud storage systems: Mozy, Carbonite, Dropbox, and CrashPlan. After the evaluation, it was found out that none of these systems can provide any guarantees for data integrity, availability, or even confidentiality. Motivated by these limitations, we need to design secure cloud storage architecture. Such architecture aims to encrypt the data that will be uploaded into cloud and protect the keys used for encryption. More precisely, such architecture should provide:

1) Confidentiality: encryption of data before uploading it to the public cloud and decryption of data after downloading from the cloud.
2) Secure data sharing: only authorized users have access to data.

III. RELATED WORK

Public cloud storage services interact with their customers either through web interface, Application Programming Interface (API) or proprietary software clients. Therefore, these services allow the users to share and synchronize data files without any direct interaction between users or knowledge about data encryption, access control polices and key management. In addition, in file sharing, the users use the web interface to share data subscribers and non-subscribers according to certain privileges. This implies that the cloud storage services are responsible for data encryption, key management, file sharing, and synchronization which make it vulnerable to all of the above threats. Therefore, we need to a secure data in cloud storage service by enforcing data confidentiality and access control to outsourced data.

A. Data Confidentiality

Current cloud storage services try to secure user's data by encrypting them either on server side or client side. In server side encryption as it is the case in dropbox, the data owner relies on the service for securing its data; however, this solution isn't feasible for two reasons. First, the user will send his plaintext to service which exposes it to internal attacks where the attacker can exploit vulnerabilities of servers to achieve user’s data. In other words, user's data, in addition to, encryption keys are stored at provider's servers. Second, there is no guarantee that the service will encrypt the data before uploading it to the cloud [10].

On the other hand, in client side encryption as it is the case in Wuala [11], the service encrypts user's data locally before it is uploaded to the cloud. Although client side encryption appears to be good method for securing users' data, it isn't efficient to do so. Since the keys involved in the process of encryption are managed by software manner. Moreover, these cloud storage services may be exposed to the following threats: a) key disclosure: the client software uses the decryption key stored on user machine to decrypt the encrypted data send from the cloud storage provider to obtain the clear text. The client software might send this key to the provider or some other unauthorized parties; b) Manipulated file content: since these cloud services support public key cryptography, the public keys of the users are known by some parties, including the provider. Server software may encrypt a malicious content making use of user's public key. The user can decrypt this content without detecting the fraud. This usually takes place because the data is usually not signed; and c) the most dangerous threat is a secret agent working at the provider. This agent may be able to manipulate the client software by injecting a malware in the customer's system [12].

B. Fine Grained Access Control

One of the most challenging issues in current cloud-based file sharing service is the enforcement of access control policies and the support of policies updates. However, current cloud storage services separate the roles of the data owner from the CSP, and the data owner does not interact directly with data users for the purpose of data access service, which makes the data access control in cloud storage services a challenging issue. Moreover, the current deployment model of cloud storage services cannot be fully trusted by data owners; as a result, traditional server-based access control methods are no longer applicable to cloud storage systems.

To prevent the un-trusted servers from accessing sensitive data in a traditional server-based system, traditional methods usually encrypt files by using the symmetric encryption approach with content keys and then use every user’s public key to encrypt the content keys and only users holding valid keys can access the data. These methods require complicated key management schemes and the data owners have to stay online all the time to deliver the keys to new user in the system. Moreover, these methods incur high storage overhead on the server, because the server should store multiple encrypted copies of the same data for users with different keys [13] [14]. In addition, these methods [15] [16] [17] deliver the key management and distribution from the data owners to the remote server under the assumption that the server is trusted or semi-trusted. However, the server cannot be trusted by the data owners in cloud storage systems and thus these methods cannot be applied to access control for cloud storage systems[18][19].

Attribute-based encryption (ABE) [20] is regarded as one of the most suitable technologies for realizing a fine-grained attribute-based access control mechanism. Since its introduction, two complementary schemes have been proposed, which are: key-policy ABE (KP-ABE) [21] and CP-ABE [22]. In a KP-ABE scheme, the ciphertext is defined by a set of attributes; while the secret keys of the user are associated with an access policy (access structure).A user can decrypt the ciphertext, if and only if he has the required secret keys corresponding to attributes listed in the ciphertext. As a result, the encryptor does not have entire control over the encryption policy because the encryption policy is described in the keys. Therefore, the encryptor has to trust the key generators for issuing correct keys for authorized users. On other hand, in a CP-ABE scheme, the ciphertext is associated with an access policy (access structure); while the secret keys of the user are defined by a set of attributes. A user can decrypt the ciphertext, if and only if his attributes satisfy the access policy. Therefore, it is more convenient for use in the cloud environment, because the encryptor holds the ultimate authority about the encryption policy. Moreover, in CP-ABE schemes, the access policy checking is implicitly conducted inside the cryptography. That is, there is no one to explicitly evaluate the policies and make decisions on whether allows the user to access the data [22][23].

C. *Revocation*

In most data sharing services, users may join and leave the system frequently. This requires a periodical re-encryption of data, and regeneration of new secret keys to remaining authorized users. However, this traditional revocation scheme isn't applicable in cloud sharing services that have high turnover rate. Therefore, Pirretti et al. [24] introduces the first revocation scheme for ABE in which the attributes are extended with expiration dates. An improvement to this scheme [22] issues a single key with some expiration dates rather than a separate key for every time period before it. However, these methods aren't able to achieve user revocation in a timely fashion. They can just disable a user secret key at a designated time, but are not able to revoke a user attribute/key on the ad hoc basis. A better solution can be achieved by delegating the re-encryption and key generation to a third party. This third party has the capabilities to execute these computational intensive tasks, e.g., re-encryption, while leaking the least information. Proxy re- encryption [25][26] is a good choice, where a semi-trusted proxy is able to convert a ciphertext that can be decrypted by a user into another ciphertext that can be decrypted by another, without knowing the underlying data and user secret keys.

IV. OUR SOLUTION

Our goal is to design a secure cloud storage service that ban the cloud from getting access to owner's plaintext or credentials, perform user's revocation without re-encrypting the affected files. In order to achieve these goals, we utilize and uniquely combine the following advanced cryptographic techniques: CP-ABE and PRE. Particularly, the proposed service transfers the trust from the cloud to a trusted third party (TTP) service. Since the currently deployed encryption services in either the cloud or inside client side cloud storage services are vulnerable to security attacks, we address these vulnerabilities by using a TTP service. This TTP service has encryption/decryption service that can be employed either locally or on top of the cloud storage. Since not all data offers the same value and not all require the same degree of protection even if it is encrypted locally on user ma-chine. Therefore, the service offers different encryption algorithms according to data's severity. For achieving data confidentiality against unauthorized users, the TTP service collaborates with an Attribute Authority (AA) through CP-ABE to achieve fine grained access control. Last but not least, PRE is used to issue different encryption key with each revocation to prevent revoked users from access the data.

Our main contributions are: 1) to design trusted third party service that enables users to share data over any web-based cloud storage platform while data security is preserved. This service protects the confidentiality of the communicated data and it can be employed locally or remotely;2) to present a CP-ABE scheme which allows users to share data between owners and users while maintaining fine grained access control scheme;3)to propose an efficient revocation scheme for CP-ABE scheme through PRE scheme. The proposed scheme tries to resolve the flaw in the granting pattern of most PRE that is employed in combination with CP-ABE. In addition, the scheme shall delegation most of computational tasks to CSP.

A. *Models and Assumptions*

- The cloud severs are honest and curious, which means that the cloud administrators can be interested in viewing user's content, but cannot modify it.
- Neither data owners nor users will be always online. They come online only when necessary.
- Legitimate users behave honestly, by which we mean that they never share their decryption key with the revoked users.
- All communications between users/clouds are secured by SSL/TLS protocol.

B. *Definition of System Model and Framework*

The system consists of the following five entities:

AA is an independent attribute authority that is responsible for issuing, revoking and updating user's attributes according to their role or identity in its domain. The authority computes a system-wide public key that is used for all operations within the system, and master key at the initialization phase in order to generate private keys for data users.

CSP is a semi-trusted entity that includes a proxy server. It is responsible for providing data storage service (i.e., Backend Storage Servers). Proxy servers are servers that are always available for providing various types of data services (i.e., proxy re-encryption technique).

TTP is an independent entity that is trusted by all other system components, and has expertise and capabilities to perform extensive tasks. The trusted third party contains two services for ensuring data confidentiality: data encryption service and data decryption service. The data encryption service is in charge of encrypting users' data. It doesn't keep any data after the encryption. On the other hand, the data decryption service only decrypts the data. In addition, it would not store any data at its end, it only stores keys. These keys are stored on hardware devices for better security.

Data owner encrypts the data with the help of TTP service (which could be local or remote). Then, the owner defines the access policies over a set of attributes

Data user has a global identity in the system with which he is entitled a set of attributes.

C. *Ensure confidentiality of data*

Since cloud storage services allow users to access data from anywhere and from any device. This means that cloud storage services serve two types of users: desktop users and mobile users. These users trust the cloud storage service by different levels. Specifically, there are three levels of trust to cloud storage service: full trust, partial trust, and no trust.

For these reasons, we offer three security mechanisms for ensuring data confidentiality for the three levels of trust.

- No trust: in this state the cloud users don't trust the cloud or any trusted third party for its data.

Therefore, cloud users shall employ the TTP on a private cloud.

A private cloud is solely owned by a single organization and managed internally or a trustworthy third party. As the private cloud is meant for a single organization, the threat of compromise of data with outside world is mitigated. Enforcement and management of security policy become easier as these things have to deal with a single organization.

- Partial trust: in this state the cloud users may trust any trusted third party that is employed on top of cloud server for its data.
- Full trust: in this state the cloud users trust the CSP for its data. As a result, the CSP handles all operations related to data confidentiality.

Not all data offer the same value and not all require the same degree of protection even if it is encrypted by locally on user machine. Therefore, any organization must adopt data classification schemes according to their level of confidentiality. Different encryption algorithms can be used to each data type according to its importance. In our scheme each trust level provides a user with three levels of data classification (low, average, high). Each level is associated with an encryption algorithm that is suitable for its security level so as to achieve better performance [27].

D. Secure data sharing

Secure outsourcing data to an un-trusted server has been studied for decades. Researchers have proposed many solutions to protect confidentiality and control the access to the outsourced data. In this paper, we combine proxy based encryption and CP-ABE [22] in an efficient way to achieve fine grained access control. In our method, we propose two layers of encryptions: owner's level encryption and access control's level encryption to achieve efficient data sharing.

To upload a file to the CSP, the TTP, that is employed either locally or remotely, encrypts the file with a symmetric encryption algorithm based on the sensitivity level selected from the owner to this file. This requires the first encryption level (inner layer). Next, it encrypts the data file symmetric encryption key (DEK) with a public key generated from the AA of CP-ABE based on users' credentials to produce the second level of encryption(outer layer). After that, the file is uploaded to CSP. The main contribution of our scheme is separation of the encrypted data from the access control. Therefore the first layer of encryption is for encrypting the data while the second one is for the access control. Therefore, any change in the access policies will only affect the outer layer (data access layer) without affecting the encrypted data. In addition, we allow the cloud to re-encrypt both data and attributes without disclosing owner's data, keys and attributes to any party. The only information disclosed to the cloud is the proxy keys provided by the TTP. These keys will be used to re-encrypt user's data and attributes.

When a new user wants to join the system, the data owner has to define the role of user and sends this information to the AA to generate a secret key based on user's role. The user in turn can use this secret key to access the data.

E. User Revocation

Users may join and leave the system frequently, leading to constant key re-generation and re-distribution through additional communication sessions to handle user revocation. In a highly scalable system composed of thousands of users, such events may occur at relatively high frequency. Researchers have proposed revocation by attaching an expiry date to the keys or introducing proxies [22] [28]. However, these approaches suffer from delay in revocation, increasing the size of ciphertext, or affecting (re-keying) all the users including both the revoked and non-revoked ones. In addition, most of the proxies have a deficiency in their granting pattern which is: “all or nothing”. If the proxy knew the proxy key from user A to user B, all A’s ciphertexts can be re- encrypted to ciphertexts of B. In other words, we have to fully trust the proxy because it has full control over the re-encryption keys. Therefore, we tried to handle problem associated with proxy server with the help of [29].

Whenever a user revocation take place, the AA just generates proxy re-encryption keys. However, it will not be sent them directly to the proxy. Instead, every time a revocation takes place, AA generates a one-time re-encryption key for this session (revocation event) and sends it to the proxy. The one-time key is a randomization of the original re-encryption key which can be used re-encrypt data in the same session. Therefore, the proxy cannot re-encryption any new data with previous re-encryption keys generated in the previous session.

V. CONCLUSION AND FUTURE WORK

In this paper, we defined a new framework for data security in cloud storage services. Through this framework, we were able to achieve data confidentiality and fine grained access control. In addition, our scheme was able to shift most of the extensive computation load to the cloud as data re-encryption to the cloud. We also proposed a technique of flexible revocation that enables owners to revoke users with less computational requirements and avoids collusion between the proxy and the users. Our future work is to evaluate this system by implementing the entire architecture nd testing its behavior in order to prove its efficiency.

REFERENCES

[1] M. Armbrust et al., “Above the clouds: A Berkeley view of cloud computing,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS- 2009, Feb. 2009.

[2] M. S. Varshini, “An Improved Security Enabled Distribution of Protected Cloud Storage Services by Zero- Knowledge Proof based on RSA Assumption,” International Journal of Computer Applications, vol. 40, no. 5, Feb. 2012, pp. 18–22.

[3] W. Jansen and T. Grance, “Guidelines on security and privacy in public cloud computing,” NIST Special Publication, Dec. 2011, pp. 800–144.

[4] Cloud security appliance: Security guidelines for critical areas of focus in cloud computing.[Online]. Available:

https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf [retrieved: Feb. 2014]

[5] "Dropbox authentication: insecure by design", Derek Newton, April 7, 2011, [Online]. Available: http://dereknewton.com/2011/04/dropbox-authentication-static-host-ids/ [retrieved: Jan. 2014]

[6] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. Weippl, " Dark clouds on the horizon: Using cloud storage as attack vector and online slack space," Proc. of the 20th USENIX Conference on Security, Aug. 2011, pp. 5-5.

[7] http://techlogon.com/2012/03/09/box-com-security-issues-for-personal-accounts/ [retrieved: Mar. 2014]

[8] https://spideroak.com/ [retrieved: Feb. 2014]

[9] W. Hu, T. Yang, and J. N. Matthews, "The good, the bad and the ugly of consumer cloud storage," ACM SIGOPS Operating Systems Review, Jul. 2010, pp. 110–115.

[10] B. Chacos, "How to encrypt your cloud storage for free," PCWorld, Sep 25, 2012, [Online]. Available:http://www.pcworld.com/article/2010296/how-to-encrypt-your-cloud-storage-for-free.html [retrieved: 3, 2014]

[11] "Wuala," [Online]. Available: http://www.wuala.com (last accessed 7/1/2014)

[12] M. Borgmann et al, "On the Security of Cloud Storage Services," Fraunhofer Institute for Secure Information Technology SIT, March. 2012, pp. 44-47, [Online]. Available: http://www.sit.fraunhofer.de/en/cloudstudy.html [retrieved: Mar. 2014]

[13] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, “Plutus: Scalable secure file sharing on untrusted storage,” Proc. of the 2nd USENIX Conference on File and Storage Technologies. Berkeley, CA, USA, USENIX Association, Mar. 2003, pp. 29–42.

[14] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, “SiRiUS: Securing remote untrusted storage,” Proc. of the Tenth Network and Distributed System Security (NDSS) Symposium, Citeseer, Feb. 2003, pp.131–145.

[15] D. Naor, M. Naor, and J. Lotspiech, “Revocation and tracing schemes for stateless receivers,” 21st Annual International in Advances in Cryptology–CRYPTO 2001, Springer, Aug. 2001, pp. 41–62.

[16] S. D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “Over-encryption: management of access control evolution on outsourced data,” Proc. of the 33rd international conference on Very large data bases, Sep. 2007, pp. 123–134, 2007.

[17] S. D. C. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “A data outsourcing architecture combining cryptography and access control,” Proc. of the 2007 ACM workshop on Computer security architecture, CSAW ’07. New York, NY, USA, ACM, Oct. 2007, pp. 63–69.

[18] A. Sahai, and B.. Waters, "Fuzzy Identity-based Encryption," Proc. of the 24th annual international conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT'05),Springer, May. 2005, pp. 457–473.

[19] W. Wang, Z. Li, R. Owens, and B. Bhargava, “Secure and efficient access to outsourced data,” Proc. of the 2009 ACM workshop on Cloud computing security, ACM, Nov. 2009, pp. 55–66.

[20] "Mozy," [Online]. Available: http://www.mozy.com [retrieved: Feb. 2014]

[21] R. Ostrovsky, A. Sahai, and B. Waters," Attribute-based Encryption with Non-monotonic Access Structures," Proc. of the 14th ACM conference on Computer and communications security (CCS 2007), Alexandria,VA, USA, Oct. 2007, pp. 195–203.

[22] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute based encryption,” IEEE Symposium on Security and Privacy, May. 2007, pp. 321–334

[23] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," Proc. of the 4th International Conference on Practice and Theory in Public Key Cryptography (PKC’11), Springer, Mar. 2011, pp. 53–70.

[24] M. Pirretti, P. Traynor , P. Mcdaniel, and B. Waters "Secure attribute-based systems," Proc. of the 13th ACM Conf. on Computer and Communications Security. New York, ACM Press,Oct. 2006, pp. 99-112.

[25] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," Proc. of International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT), May. 1998, pp. 127–144, 1998.

[26] M. Green and G. Ateniese, "Identity-based proxy re-encryption," Proc. of the 5th international conference on Applied Cryptography and Network Security, Sep. 2007, pp. 288–306.

[27] H. Chuang, S. Li, K. Huang, and Y.Kuo, "An effective privacy protection scheme for cloud computing," 2011 13th International Conference on Advanced Communication Technology (ICACT) , Feb. 2011, pp.260-265.

[28] S. Tu, S. Niu, H. Li, Y. Xiao-ming, and M. Li, “Fine-grained Access Control and Revocation for Sharing Data on Clouds,” 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum , May. 2012, pp. 2146–2155.

[29] X. Wu, L. Xu, and X. Zhang. " CL-PRE: a certificateless proxy re-encryption scheme for secure data sharing with public cloud," Proc. of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS '12), May. 2012, New York, NY, USA, ACM, pp. 87-88.

A Loosely-coupled Semantic Model for Diverse and Comprehensive Cloud Service Search and Retrieval

Daren Fang, Xiaodong Liu, Imed Romdhani
School of Computing
Edinburgh Napier University, UK
emails: {d.fang, x.liu, i.romdhani}@napier.ac.uk

Abstract—**As cloud services propagate along with the rapid development of cloud computing, issues raised in service selection and retrieval processes become increasingly critical. While many approaches are proposed on the specification and discovery of cloud services, existing service models and recommendation systems cannot achieve ultimate effectiveness while dealing with a variety of cloud services of different categories/levels/characteristics. To this extent, this paper proposes a Cloud Service Explorer (CSE) tool, which takes advantage of a Loosely-Coupled Cloud Service Ontology (LCCSO) model as the knowledge base to assist user-friendly service search and service information access. Demonstrated using a number of real world cloud services and their official service data as examples, it proves that the model and tool are capable of offering an efficient and effortless means of handling diverse service information towards ultimate service discovery and retrieval.**

Keywords-cloud computing; cloud service; semantic model; ontology; service recommendation system.

I. INTRODUCTION

Cloud Computing (CC) introduces a revolutionary information and communication technology (ICT) paradigm to the world, known as on-demand provisioning, pay-per-use self-service, ubiquitous network access and location-independent resource pooling [1]. It provisions reliable, scalable and elastic computational resources that effectively adapt to nearly all kinds of needs. Within barely a decade, CC has permeated into all major industry sectors. However, as the number of cloud services continues growing whilst the market becomes more complex, it would take considerable time and efforts for service users to find the targeted services, by researching a great many service descriptions, properties, Service Level Agreements (SLAs), ratings, reviews, trials, etc. Moreover, regarding services' functionality, usability, customizability, etc., existing cloud providers pose various interfaces, standards, service level data and explanations [2][3], which result into serious difficulties in service information retrieval, interpretation and analysis.

Consequently, the rapid development of CC has imposed urgent needs yet great challenges on the specification and retrieval of cloud services, whereas an effective means of cloud service search and discovery is under demand for a diversity of users. A successful model of cloud services should be able to cope with the following challenges: 1) Cloud services and resources involved in CC systems are across multiple abstraction levels, from Infrastructure-as-a-Service (IaaS), to Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Such a matrix structure has incurred that cloud services contain dependencies over one another, and there are close or loose relations across the different levels of cloud services/resources. 2) Cloud services are rather complicated in both functional and non-functional aspects, whereas certain well-resourced services can be used flexibly to achieve diverse ranges of functions. These can be seen as the potential capabilities of the services. Accordingly, a superior service explorer system which works on top of the model should then be capable of providing ultimate service search and retrieval to meet complex requirements for all users regardless of their types or levels of expertise.

Many efforts have been made on the semantic specification of cloud services using OWL/OWL2 [4] ontology modeling approach, e.g., [5][6][7][8][9][10]. Yet, the majority of the existing models focus on specifying certain specific service categories, whereas very few of them utilize the various types of OWL 2 property assertions for ultimate service information specification. These drawbacks expose significant issues when users try to retrieve cloud services from the pool, since there is not a unified knowledge base that can hold comprehensive cloud service information for all service categories (all IaaS, PaaS and SaaS services of different infrastructure provision, platform provision and software functions).

In this paper, the authors propose a Loosely-Coupled Cloud Service Ontology (LCCSO) model, which takes advantage of flexible concept naming as well as loosely-coupled axiom assertions to ultimately comprise comprehensive specifications of diverse cloud services from distinct levels and categories. Moreover, LCCSO employs the full range of OWL 2 axiom assertions including annotation, class, Object Property (OP), Data Property (DP) assertions. This consequently enhances service specification by involving various forms of service information. Using the semantic model as the central cloud service knowledge base, a Cloud Service Explorer (CSE) prototype tool is also developed to best assist users in cloud service search and retrieval tasks. The contributions of the paper are: 1) a loosely-coupled cloud service semantic model that is able to present cloud services and comprehensive service information regardless of their categories, levels, characteristics, functional or non-functional properties; 2) a cloud service search and retrieval tool that is capable of providing an effective means of service discovery and service information access.

The rest of the paper is organized as follows: Section 2 discusses the related work. In Section 3, the design and implementation of LCCSO are demonstrated. Section 4 outlines the architecture of CSE prototype whilst the details of the system components are explained. The search and retrieval functions of CSE tool are further explored in Section 5, where examples of cloud service retrieval and tool screenshots are illustrated. Section 6 concludes the paper with summaries and future work.

II. Related Work

Ontology-oriented or OWL-based approaches have been used widely in assisting service discovery, recommendation and composition. While its basic semantics provide comprehensive service annotations and descriptions, schema mapping and model reference, interface and operation parameter information, Description Logic (DL) reasoning is able to reveal additional inferred knowledge intelligently [2].

In the efforts of cloud computing/service semantic specification, existing ontology models expose various limitations considering the comprehensiveness and types of the knowledge revealed, whereas many of them are designed concentrating on certain restricted service categories, e.g., infrastructure services [5][9][10][11][12], platform services [7], software services [6][13]. More specifically, in Cloudle [9] and CSDS [12], both models developed cover only fundamental CC concepts, such as delivery models (e.g., IaaS, PaaS, SaaS), cloud hardware (e.g., CPU, memory, disk), software (e.g., OS, DBMS), programming languages (e.g., Python, C#, Java), etc. This implies that the specification would be limited to ordinary basic IaaS, PaaS and SaaS services. On the other hand, very few of the ontology models adopt comprehensive ontological assertion types to enhance the service specification. For instance, annotation properties are the main focus of the semantic platform for cloud service annotation and retrieval [6], whereas many others [9][11][12] are primarily concentrating on data-relevant service properties. Consequently, current existing cloud service models fail to deliver the best means of service specification.

Recently, various cloud service recommendation systems are developed, which rely on diverse service modeling and matchmaking techniques. The collaborative service recommender mechanism [14] is argued to specifically deal with consumer rated service quality matchmaking based on individual user's profile. While the prototype mainly works for non-functional (response time, availability, price, etc.) aspects-oriented service discovery, it suggests that such would not handle the diverse functional aspects effectively. In contrast, many others employ ontology models for cloud service discovery and recommendation; they produce enhanced results, yet still can be improved. CSDS [12] is primarily designed to deal with IaaS services; therefore the search/discovery is restricted to only that category. CloudRecommender [11] offers enhanced functions for various types of infrastructure service recommendations by allowing users to enter both functional and non-functional requirements. Nonetheless, it still cannot work perfectly for PaaS or SaaS service discovery. The cloud repository and discovery framework [7] advocates using a unified business and clouding service ontology to assist service discovery tasks. Although the proposed approach can support PaaS and IaaS services, the search functions are poorly provided due to the business function/process-oriented design where users have to specify certain business-relevant service requirements. Indeed, existing cloud service discovery/ recommendation systems cannot facilitate comprehensive service lookup across different service levels/categories whilst they fail to involve service characteristics aspects as search requirements.

In summary, current semantic models of cloud computing/services are not able to provide the best means of service specification due to the conventional category-restricted design and uneven use of ontology assertion types; existing cloud search/discovery approaches cannot effectively deal with the various cloud services of distinct categories, levels, characteristics or functional/non-functional properties.

III. Loosely-Coupled Ontology Design

In comparison with other work, LCCSO incorporates flexible concept naming and loosely-coupled axiom assertions to cover diverse service information across different service layers and categories. It utilizes a wide range of ontology assertion types to comprehensively reveal details regarding service functions, characteristics and features.

A. Cloud services and service models

While other work classifies cloud services according to the delivery or deployment models (e.g., Amazon EC2 belongs to IaaS or Public cloud), LCCSO gathers all cloud services and their originated companies into one class named "Registered Cloud Entity". The class can be seen as a "service registry" within the ontology (see Fig. 1). The advantages of the design is seen as: 1) it specifies clear subsumption relationships between a cloud company/ provider (class) and the services (individuals) it owns, whereas it allows to assert relevant relationships among cloud companies and services; 2) it enables effective service retrieval, lookup and processing from one united class, instead of extracting services from multiple service model classes; 3) it achieves flexible service specification assertions for cloud services, e.g., Amazon S3 can be asserted as "belongs to" multiple service models such both IaaS and PaaS, or PaaS; 4) the comparison among cloud

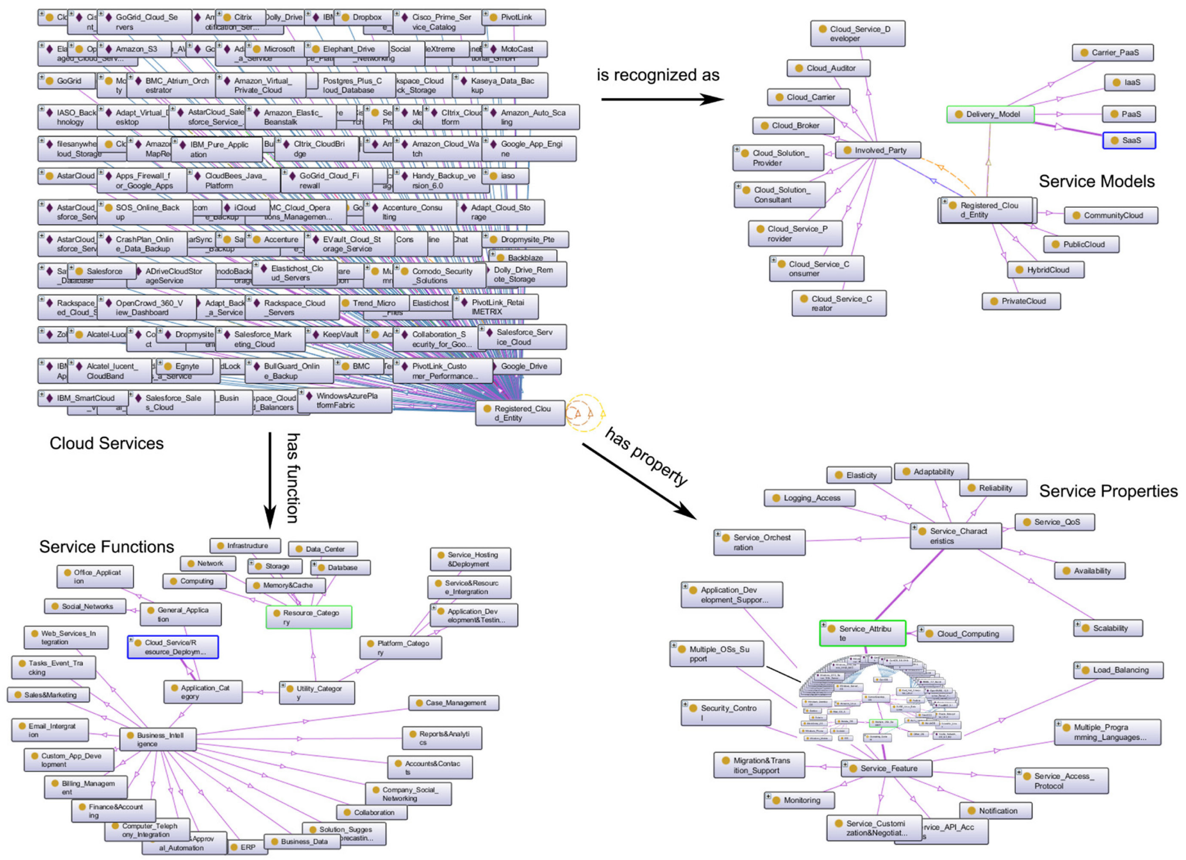


Figure 1. Loosely-Coupled Cloud Service Ontology.

companies and services can be effectively implemented, since they belong to a single class where their axiom assertions follow the same pattern.

B. *Service functions*

In LCCSO, “Utility Category” class comprises all the service functions that cloud services are capable of providing. As illustrated in Fig. 1, they are divided into three categories: resource, platform and application. “Resource Category” involves the various computational resources that cloud services (typically IaaS) can provision, i.e., “Computing, Storage, Database, Network, Data Center”, etc. “Platform Category” covers the usages of provisioned cloud platforms (most likely PaaS), seen as “Application Development and Testing, Service and Resource Integration, Service Hosting and Deployment”, etc. “Application Category” gathers the various application-alike (SaaS) cloud service functions, including “General Applications, Business Intelligence, and Cloud Service or Resource Deployment and Management”, etc. The use of these classes is to reveal the exact capability of cloud services, instead of simply justify the primary designed function(s). For instance, a SaaS service may only provide one specific or very limited function(s) (in Software Category). Some PaaS services, however, can have multiple functions (from Platform Category), e.g., for application testing and deployment, whereas any service functions provisioned on top of the platforms can also be attributed to these PaaS services. Likewise, typical IaaS services, like VM provision services, can be used for multiple resource functions, such as provisioning computing, database, network and storage flexibly. Plus, for certain well-resourced IaaS services which are capable of providing application development platforms (used as PaaS services); their additional usages would be even more diverse [15]. As such, these service function specifications provide a comprehensive view for cloud services of different models.

C. *Service properties*

Service properties are divided into “Service Characteristics” and “Service Features”, in LCCSO. While the former involves the common and unique cloud service properties such as “Adaptability, Elasticity, Scalability, Availability, Reliability”, etc., the latter consists of relevant “extra” cloud service properties such as “Service Access Protocol, Service API Access, Service Customization and Negotiation, Security Control, Multiple operating system (OS)/Programming Language/Platform Support, Monitor, Notification”, etc. (demonstrated in Fig. 1). By associating these properties with relevant cloud services, the comparison among services can be implemented effectively.

D. *Ontology axiom assertions*

In contrast with other existing cloud (service) ontology models, which only utilize partial ontology assertion types, LCCSO adopts full range of OWL2 assertions for comprehensive service specifications. Firstly, all CC concepts (including cloud services) come with Annotation assertions which provide general knowledge of them. Secondly, Description assertions (or Class assertion), in the form of "individual of/subclass of assertions" and "individual-to-class" and "class-to-class" OP assertions, are employed to declare overall specifications of CC concepts (e.g., to specify overall cloud companies affiliations and relationships, delivery/deployment models, service characteristics and features information). Thirdly, additional CC concepts specifications (e.g., to clarify specific details of the individuals) are revealed by using Property assertions. These involve both DP assertions and "individual-to-individual" OP assertions.

This way, the axiom assertions in LCCSO become logical. To some extent, the Description assertions can be considered as further information of the Annotation assertion of a cloud service, whereas Property assertions can be regarded as extensions of the contents appeared in the Description assertions. Moreover, with flexible and loosely-coupled naming, domain, ranges, classification and OP deployment, a number of classes, individuals and properties are being used for multiple assertion needs where potential redundancy issue can be avoided.

Take virtual server IaaS services as an example, in their service annotations there would be relevant OS provision descriptions as such are typical aspects of the services; therefore, these services could own the "Multiple OS Support" assertion (Description assertion). Likewise, a large number of SaaS services are offered with multiple OS platforms accessibility, which means they could also be asserted with the "Multiple OS Support" assertion. As a consequence, the "OS" class would then consist of all current mainstream OSs involved (for both IaaS and SaaS) here: "Mobile OS" class comprises "Android, IOS, Windows Mobile, Blackberry OS", etc.; "Server/ desktop OS" class comprises different OS platforms, e.g., "Debian", "Fedora", "Gentoo", "Solaris", "Ubuntu", "Windows", etc., where each one has own detailed OS versions (refer to Fig. 1). Then, in order to clearly present the differences between the two "Multiple OS Support", OP "supports VM OS of" is used to relate the IaaS services to their achievable OSs whilst OP "offers management application for OS of" is adopted to relate the accessible OSs with the SaaS services.

IV. CLOUD SERVICE EXLORER SYSTEM OVERVIEW

As depicted in Fig. 2, CSE prototype employs LCCSO as the formal and consistent knowledge source base to provide cloud service discovery and retrieval functions. The system consists of three main components, namely, Ontology Manager, Service Search Engine, and User Interface.

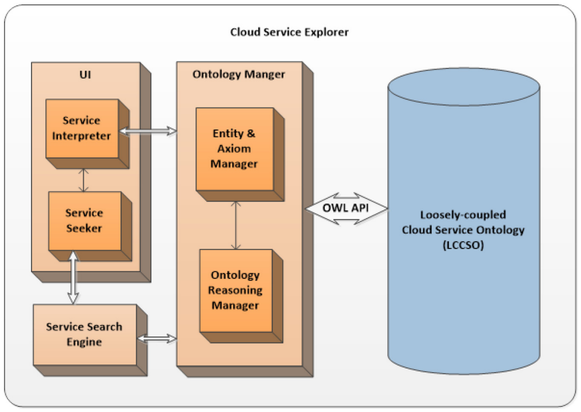


Figure 2. Cloud Service Explorer system architecture.

While the role of **LCCSO** is to provide comprehensive raw data of cloud services and other relevant CC concepts, CSE interprets it through OWL API [16] and translate the information into various general and detailed service descriptions.

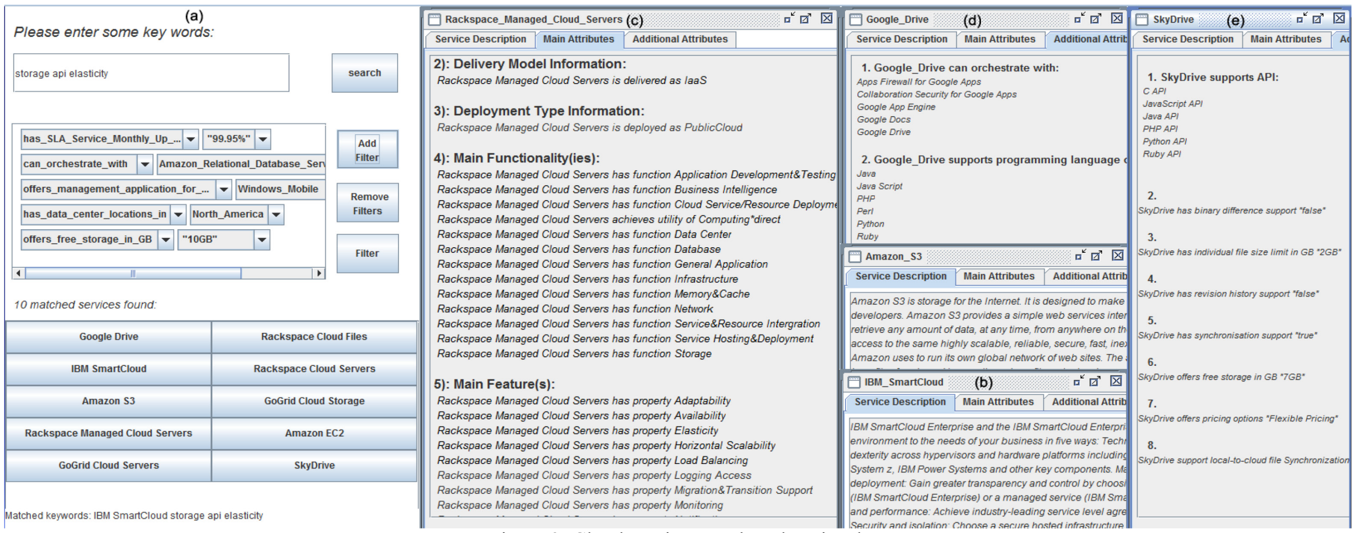
Ontology Manager manages raw data extraction from LCCSO. It incorporates Entity and Axiom Manager and Ontology Reasoning Manager, which together manage the ontology parsing tasks. While **Entity and Axiom Manger** reads all types of asserted ontology concepts and axioms, including individual, class, OP and DP concepts, class, and OP and DP assertions, **Ontology Reasoning Manager** handles ontology consistency checks and inference controls by importing OWL2 ontology reasoner. Here, FaCT++ is chosen due to its faster reasoning process and better syntax and property characteristics support than other reasoners such as HermiT and Pellet [17]. Here, to take advantage of ontological modeling, any new knowledge discovered after reasoning process (inferred axioms) is also be used for service discovery and explore.

Service Search Engine accepts user's keywords or/and filters entries input to provide service/concept search functions. Through Ontology Manager, it extracts asserted service descriptions, attributes and other data details and analyzes such against users' input. As the information pairing process is complete, the component outputs a list of relevant cloud services User Interface.

User Interface comprises Service Interpreter and Service Seeker which interact with system users while they search/view cloud services. **Service Interpreter** translates the raw ontology axioms into naturally described contents so that they can be easily understood. **Service Seeker** interacts with Service Search Engine and manages the display of search keywords, filters and service list result.

V. CLOUD SERVICE SEARCH AND RETRIEVAL

CSE prototype is implemented in Java. Currently, with specifications of approximately two hundred cloud services and companies/providers available in LCCSO, it can provide flexible service search, logical service property



Please enter some key words:

storage api elasticity

has_SLA_Service_Monthly_Up_...	"99.95%"
can_orchestrate_with	Amazon_Relational_Database_Serv
offers_management_application_for_...	Windows_Mobile
has_data_center_locations_in	North_America
offers_free_storage_in_GB	"10GB"

10 matched services found:

Google Drive	Rackspace Cloud Files
IBM SmartCloud	Rackspace Cloud Servers
Amazon S3	GoGrid Cloud Storage
Rackspace Managed Cloud Servers	Amazon EC2
GoGrid Cloud Servers	SkyDrive

Matched keywords: IBM SmartCloud storage api elasticity

Rackspace_Managed_Cloud_Servers

2): Delivery Model Information:
Rackspace Managed Cloud Servers is delivered as IaaS

3): Deployment Type Information:
Rackspace Managed Cloud Servers is deployed as PublicCloud

4): Main Functionality(ies):
Rackspace Managed Cloud Servers has function Application Development&Testing
Rackspace Managed Cloud Servers has function Business Intelligence
Rackspace Managed Cloud Servers has function Cloud Service/Resource Deployme
*Rackspace Managed Cloud Servers achieves utility of Computing*direct*
Rackspace Managed Cloud Servers has function Data Center
Rackspace Managed Cloud Servers has function Database
Rackspace Managed Cloud Servers has function General Application
Rackspace Managed Cloud Servers has function Infrastructure
Rackspace Managed Cloud Servers has function Memory&Cache
Rackspace Managed Cloud Servers has function Network
Rackspace Managed Cloud Servers has function Service&Resource Intergration
Rackspace Managed Cloud Servers has function Service Hosting&Deployment
Rackspace Managed Cloud Servers has function Storage

5): Main Feature(s):
Rackspace Managed Cloud Servers has property Adaptability
Rackspace Managed Cloud Servers has property Availability
Rackspace Managed Cloud Servers has property Elasticity
Rackspace Managed Cloud Servers has property Horizontal Scalability
Rackspace Managed Cloud Servers has property Load Balancing
Rackspace Managed Cloud Servers has property Logging Access
Rackspace Managed Cloud Servers has property Migration&Transition Support
Rackspace Managed Cloud Servers has property Monitoring

Google_Drive

1. Google_Drive can orchestrate with:
Apps Firewall for Google Apps
Collaboration Security for Google Apps
Google App Engine
Google Docs
Google Drive

2. Google_Drive supports programming language o
Java
Java Script
PHP
Perl
Python
Ruby

Amazon_S3

Amazon S3 is storage for the Internet. It is designed to make
developers. Amazon S3 provides a simple web services inter
retrieve any amount of data, at any time, from anywhere on th
access to the same highly scalable, reliable, secure, fast, inex
Amazon uses to run its own global network of web sites. The

IBM_SmartCloud

IBM SmartCloud Enterprise and the IBM SmartCloud Enterpri
environment to the needs of your business in five ways: Techn
dexterity across hypervisors and hardware platforms including
System z, IBM Power Systems and other key components. Ma
deployment: Gain greater transparency and control by choosi
(IBM SmartCloud Enterprise) or a managed service (IBM Sma
and performance: Achieve industry-leading service level agre
Security and isolation: Choose a secure hosted infrastructure

SkyDrive

1. SkyDrive supports API:
C API
JavaScript API
Java API
PHP API
Python API
Ruby API

2.
SkyDrive has binary difference support "false"

3.
SkyDrive has individual file size limit in GB "2GB"

4.
SkyDrive has revision history support "false"

5.
SkyDrive has synchronisation support "true"

6.
SkyDrive offers free storage in GB "7GB"

7.
SkyDrive offers pricing options "Flexible Pricing"

8.
SkyDrive support local-to-cloud file Synchronization

Figure 3. Cloud service search and retrieval.

view and effortless service comparison functions regardless of distinct service models, functions, or user expertise levels.

A. *Cloud service search*

In CSE, the two search options offered keywords and filters. Basically, system users can enter keywords first and then add filters to view the matched service result, or vice versa. The search requirements can be of any service functions, categories, levels, characteristics, features, functional or non-functional properties, etc., which achieves a flexible and user-friendly cloud explorer experience regardless of the expertise level of users.

The search and filter tasks are implemented by calling Ontology Manager to walk through the ontology to collect entire service properties and data of all cloud services for keyword/filter matches. Here, any services involved in those asserted/inferred axioms where certain information fits the keywords/filters are extracted for shown in the search result.

The purpose of the service filter mechanism is to enhance service discovery experience, especially for those users of limited CC knowledge. It allows them to view and select from a comprehensive list of service properties and property values (so that applicable cloud services can be extracted). The service data options are automatically retrieved from LCCSO, shown in the form of drop down lists.

As depicted in Fig. 3(a), by searching with example keywords "storage, api and elasticity" plus restrictions of a series of filters, a number of applicable cloud services are returned. Users can select them to view and compare the comprehensive service information stored in LCCSO.

B. *Cloud service retrieval*

As a system user attempts to retrieve cloud services, Service Interpreter reads the raw collected cloud service specification data from Ontology Manager and categorizes the interpreted service information into "Service Description", "Main Attributes" and "Additional Attributes" three categories. As seen in Fig. 3, they are displayed in three switchable tabs.

"Service Description" tab outlines the general descriptions of cloud services by extracting the annotations of them. To ensure the availability and reliability of the contents, such information is collected from the services' official resources. As depicted in Fig. 3(b), the descriptions of Amazon S3 [18] and IBM SmartCloud [19] enable effortless information access which can effectively help users understand the basics of the services.

"Main Attributes" tab comprises a service's overall service properties, e.g., the delivery and deployment model, the primary designed service functions and other additional usages, plus the service characteristics and features. These are in fact the translations of the service's class assertions in the ontology. Using Rackspace Managed Cloud Servers [20] as an example (see Fig. 3(c)), the tab comprehensively illustrates a series of information: 1) "Main Functionalities": due to the fact that the service is seen an IaaS service which provisions virtual servers for general computing needs, it can be used for various other functionalities across multiple function categories, e.g., using such for storage and database provision as well as for application development and deployment. The arrangement helps users understand the ultimate capability of a cloud service on top of its main designed function. 2) "Main Features": the collected service characteristics and features are displayed here, regardless of whether functional or non-functional. These help users view the full picture of a cloud service clearly.

"Additional Attributes" tab displays the various additional service data by examine both cloud service's individual-to-individual OP and DP assertions in LCCSO. Due to the individual assertions for cloud services, they can have a variety of relationships among each other plus other CC concepts individuals. For instance, a service "can orchestrate with" another and "supports OS/programming language/API of" certain OSs/programming languages/APIs. As demonstrated in Fig. 3(d), Google Drive [21] can orchestrate with a number of services as listed, whereas the supported programming languages are also given. On the other hand, cloud services are asserted with a series of DP axioms. They involve clarifying a range of detailed service data: VM services have a series of VM-specific data, i.e., virtual CPU frequency, hard drive capability, memory size, network through output, etc.; storage services own a number of storage-specific data, i.e., free storage, pricing, certain particular feature supports, etc. An example of SkyDrive's [22] additional attributes is illustrated in Fig. 3(e).

VI. CONCLUSION AND FUTURE WORK

The growing number and complexity of cloud services bring us numerous advantages whilst the issues exposed while searching and retrieving the services are becoming increasingly critical. Despite of the many efforts made on the semantic specification of cloud computing/services and service discovery/recommendation approaches, current service models and recommendation systems cannot work effectively on various service and usage scenarios to provide comprehensive service search and retrieval functions. This paper presented the design and implementation of the LCCSO model and the CSE tool, which together can facilitate effective service discovery and service information access regardless of service's functions, levels, categories, or characteristics. The novel LCCSO model employs flexible concepts naming and the full range of OWL2 axiom assertion types which ultimately comprise diverse types of service specification data and information, whereas the CSE prototype tool adopts a user-friendly design that enables effortless service search and retrieval for all CC user regardless of different level of expertise.

In future work, we intend to enhance the model by focusing on the granular details of unique cloud service properties (e.g., agility, elasticity). Moreover, we are experimenting on additional tool functions, including service rating, review, benchmarking and unified service access portals.

REFERENCES

[1] R. Buyya, C. Vecchiola, and S. Thamarai, "Master cloud computing: Foundations and applications programming", Elsevier, Waltham, USA, pp. 3-27, 2013.

[2] J. M. S. Orozco, "Applied ontology engineering in cloud services, networks, and management systems", Springer Science+Business Media, pp. 23-52, 2012.

[3] J. Shen, G. Beydoun, G. Low, and L. Wang, "Aligning ontology-based development with service oriented systems", Future Generation Computer Systems, vol. 32, 2014, pp. 263-273.

[4] M. Horridge, "A practical guide to building OWL ontology's using protégé 4 and CO-ODE tools", The University of Manchester, edition 1.3, available at: http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf, 2011 [retrieved: May, 2014].

[5] R. Aversa, et al., "An ontology for the cloud in mOSAIC", Cloud Computing Book 2010: Cloud Computing: Methodology, System, and Applications, CRC Press, pp. 467-486, 2011.

[6] M. A. Rodríguez-García, R. Valencia-García, F. García-Sánchez, and J. J. Samper-Zapater, "Creating a semantically-enhanced cloud services environment through ontology evolution", Future Generation Computer Systems, vol. 32, 2014, pp. 295-306.

[7] A. Tahamtan, S. A. Beheshti, A. Anjomshoaa, and A. M. Tjoa, "A Cloud Repository and Discovery Framework Based on a Unified Business and Cloud Service Ontology" IEEE World Congress on Services (SERVICES), 2012, pp. 203-210.

[8] T. B. Pedersen, D. Pedersen, and K. Riis, "On-demand multidimensional data integration: toward a semantic foundation for cloud intelligence", The J. Supercomputing, vol. 65, no. 1, 2013, pp. 217-257.

[9] J. Kang and K. Sim, "Cloudle: A Multi-criteria Cloud Service Search Engine," IEEE Asia-Pacific Services Computing Conference (APSCC), 2010, pp. 339-346.

[10] G. Manno, W. W. Smari, and L. palazzi, "FCFA: A semantic-based federated cloud framework architecture", International Conf. High Performance Computing and Simulation (HPCS), 2012, pp. 42-52.

[11] M. Zhang, et al., "An ontology-based system for Cloud infrastructure services' discovery", 8th International Conf. Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012, pp. 524-530.

[12] T. Han and K. M. Sim, "An Ontology-enhanced Cloud Service Discovery System", International Multi-Conf. Engineers and Computer Scientists (IMECS), available at http://www.iaeng.org/publication/IMECS2010/IMECS2010_pp644-649.pdf, 2010 [retrieved: May, 2014].

[13] M. A. Rodríguez-García, R. Valencia-García, F. García-Sánchez, and J. J. Samper-Zapater, "Ontology-based annotation and retrieval of services in the cloud", Knowledge-Based Systems, vol. 56, 2014, pp. 15-25.

[14] K. Tserpes, F. Aisopos, D. Kyriazis, and T. Varvarigou, "A Recommender Mechanism for Service Selection in Service-oriented Environment", Future Generation Computer Systems, vol. 28, no. 8, 2012, pp. 1285-1294.

[15] D. C. Marinescu, "Cloud computing: Theory and practice", Elsevier, Waltham, USA, pp. 339-356, 2013.

[16] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL Ontologies Semantic Web Journal 2(1)", Special Issue on Semantic Web Tools and Systems, 2011, pp. 11-21.

[17] D. Tsarkov and I. Horrocks, "FaCT++ description logic reasoner: system description", 3ird international joint conf. Automaitc Reasoning, 2006, pp. 292-297.

[18] Amazon Web Services, Inc., "Amazon S3 Developer Guide", available at: http://awsdocs.s3.amazonaws.com/S3/latest/s3-dg.pdf, 2013 [retrieved: May, 2014].

[19] IBM Corporation, "IBM SmartCloud Entry user guide, version 3.2", available at: https://www.ibm.com/developerworks/community/wikis/form/anonymous/api/wiki/e6a8193b-bd0a-4c69-b641-328d20a4f69c/page/4cd16bdc-e3cb-4d71-948e-46f0e6548561/attachment/816e4610-008d-487a-9764-a60399ab6e1b/media/SCE_User_Guide_32.pdf, 2013 [retrieved: May, 2014].

[20] Rackspace, Inc., "Next Generation Cloud Servers Developer Guide", available at: http://docs.rackspace.com/servers/api/v2/cs- devguide/cs-devguide-latest.pdf, 2013 [retrieved: May, 2014].

[21] Google, Inc., "About Google Drive", available at: http://www.google.com/drive/about.html, 2013.

[22] Microsoft, "Live SDK developer guide", available at: http://msdn.microsoft.com/en-us/library/live/hh243641.aspx, 2013 [retrieved: May, 2014].

A Simulation Framework to Model Accountability Controls for Cloud Computing

Nick Papanikolaou
Security and Cloud Lab
HP Labs
Bristol, United Kingdom
Email: nick@nick-p.info

Thomas Rübsamen, Christoph Reich
Cloud Research Lab
Hochschule Furtwangen University
Furtwangen, Germany
Email: {ruet, rch}@hs-furtwangen.de

***Abstract*—In this paper, we present an implemented system to model and visually represent the functioning of accountability mechanisms for cloud computing (such as policy enforcement, monitoring, intrusion detection, logging, redress and remediation mechanisms) over provider boundaries along the supply chain of service providers. Service providers can use these mechanisms, among others, in a variety of combinations to address data protection problems in the cloud, such as compliance failures, losses of governance, lock-in hazards, isolation failures, and incomplete data deletion. The focus here is on technical mechanisms for the purposes of simulation (the currently implemented tool demonstrates policy enforcement, monitoring and logging); in general, an accountability approach requires a combination of technical measures and legal and regulatory support, of course. We survey existing work on accountability in the cloud and discuss ongoing research in the context of the Cloud Accountability project. We discuss modelling considerations that apply in this context namely, how accountability may be modelled statically and dynamically. Details of the current implementation of the Accountability Simulation Engine (ASE), and the first version of a graphical animation of data flows in the cloud, are described.**

***Keywords*–accountability; data protection; modelling language; simulation; visualisation; sticky policies; policy enforcement; logging; redress**

I. Introduction

In this paper, we present the background and modelling considerations associated with Accountability Simulation Engine (ASE), a simulation framework to model and visualize accountability mechanisms for cloud computing. We will discuss the motivation and objectives behind ASE, as well as the features that have been implemented so far. As this is still ongoing work, the primary purpose of the paper is to inform the community and to impart some structure on the development activities; a detailed discussion of future work has also been included.

The starting point for this work is the realization that both cloud computing service providers, as well as customers of cloud computing services, need to have a good understanding of the controls that may be used for managing data flows in the cloud while complying with prevailing data protection laws, rules and regulations, as well as industry standards, best practices, and corporate data handling guidelines in an efficient yet demonstrable manner. The massive scale of cloud computing infrastructures, as well as the enormous complexity of legal and regulatory compliance across multiple jurisdictions, makes this a significant and difficult challenge that service providers, customers, regulators and auditors need to meet on a continual basis.

Accountability for cloud computing service provision is emerging as a holistic approach to this set of issues, and is being actively developed in the context of the Cloud Accountability Project (A4Cloud) [1]. Drawing on a multitude of sources, including legal and regulatory frameworks for accountability, as well as technical solutions for achieving data protection compliance, this project aims to provide cloud service providers, auditors, regulators and others with a concrete set of tools for achieving accountability. The simulation framework described in this paper is a research tool whose purpose is to demonstrate how such tools might work, and what problems they are intended to address.

As seen in Section II, accountability encompasses a number of different controls that may be used by a cloud service provider to ensure appropriate governance of their customers data. By developing a simulation of how these controls might or should function, we can reason about their necessity and suitability to particular data handling scenarios. We are mainly interested in technical, automated means of achieving accountability; higher-level mechanisms (e.g., legal rulings or precedents, new regulations, ethical guidelines) are only implicitly modelled as rules incorporated in technical enforcement mechanisms (such as privacy or access control policies).

In order to better understand what form of simulation would be appropriate, we surveyed a number of existing simulation tools and frameworks (Section III). We identified two classes of tools discrete-event modelling formalisms with mostly textual output, as well as visual simulation tools, which permit rapid prototyping, and the creation of graphical animations.

An important part of this work has been identifying what components a suitable simulation model might include, as well as what use cases and scenarios might best illustrate the functionality of accountability mechanisms. These topics are discussed in Section IV. It is interesting to note that there are both static and dynamic aspects of accountability, and different types of simulation are suited to these aspects.

The next section details our model namely, what actors, behaviours and relationships we have chosen to include. The design choices are not definitive, and are likely to vary across use cases. However, this section establishes which kinds of issue could be demonstrated during a simulation, and which responses or mechanisms are appropriate when an accountability-based approach is taken. Section V describes our current implementation of an accountability simulation engine (ASE), which comprises (i) a domain-specific modelling language for accountability scenarios, (ii) an actual simulator for accountability related events, (iii) a web-based user interface for inputting scenario descriptions and observing simulation output, and (iv) a web service which links (ii) and (iii) together. An example of ASEs functionality is given in Section VI, with a simulation of the dynamics of a simple cloud service provision chain. Finally, we conclude with a discussion of future work; this includes prototyping a visual animation of data flows using existing simulation tools and extending the current implementation of ASE with graphical output.

II. The Need for Accountability in the Cloud

As identified by Pearson [2], accountability ”for complying with measures that give effect to practices articulated in given guidelines” has been present in many legal and regulatory frameworks for privacy protection; certainly the notion originates from the data protection context, and carries with it the idea of responsible data stewardship. The Galway project [3] attempted to define accountability in this context as follows:

Accountability is the obligation to act as a responsible steward of the personal information of others, to take responsibility for the protection and appropriate use of that information beyond legal requirements, and to be accountable for any misuse of that information.

Pearson [2] observes that the key elements of notion of accountability implied by this definition are *transparency, responsibility, assurance and remediation*. In Pearson and Wainwright [4] it is argued that, to support these elements, it is possible to co-design legal and technical controls for cloud service providers belonging to three categories (i) preventive controls (e.g., risk analysis decision support tools, policy enforcement using machine-readable policies, privacy enhanced access control and obligations), (ii) detective controls (e.g., intrusion detection systems, policy-aware transaction logs, and reasoning tools, notification mechanisms), and (iii) corrective controls (e.g., liability attribution tools, incident management tools). Other categories of controls exist for different kinds of participants in a cloud service provision ecosystem, including end users and regulators.

Our interest is in creating a simulation framework, which enables us to address concerns such as the following:

- What problems can arise in a cloud service provision chain when controls such as those described above are absent from service providers infrastructures;
- What benefits the adoption of such controls can have on service providers operational responses to problems, such as data breaches;
- How accountability can be maintained along a supply chain of cloud service providers;
- What potential impact the introduction of a new control can have on a service providers operations.

While the goal of designing the simulation is to demonstrate the added benefits of adopting an accountability approach, in order to do so it is necessary first to identify the problems and events of interest that this approach provides responses for; both the events and the responses to these events can then be explicitly accounted for in the simulation model. Additionally, audit plans can be derived from this model more precisely and more fully.

A. Data Protection Problems

Based on the risk categorization presented by Haeberlen et al. in [5], we identify here five typical classes of data protection problems that cloud service providers need to mitigate:

- Compliance failures
- Losses of governance (e.g., data breaches)
- Lock-in hazards
- Isolation failures
- Incomplete data deletion

Compliance failures. As mentioned in the introduction, cloud service providers need to ensure compliance with prevailing laws and regulations [6][7] in the jurisdictions where customers data are stored. This is a non-trivial matter, given that cloud data centres are located in multiple, different locations across the globe, and data often needs to be relocated from one data centre to another for efficiency, bandwidth or other considerations. To ensure compliance on an ongoing basis, applicable local rules need to be checked before, during and after data relocation and evidence has to be given by audits. What makes this particularly complex is that rules are not consistent everywhere, and often transformations need to be applied to the data itself (e.g., in the case of anonymisation of personal data) before a transfer can occur. Any failure to comply with laws and regulations carries significant consequences for the reputation and profits of a service provider; therefore, it is of paramount importance to ensure immediate corrective action if any case of non-compliance is detected (e.g., by audits).

Compliance hazards are not confined to legal and regulatory requirements, of course; in order to maintain industrial certifications and badges, service providers need to ensure compliance with appropriate industry standards, whether specific to cloud computing practices, data handling practices, or quality control, among other things. These are typical tasks of a cloud audit system. Failure to maintain such compliance can result in loss of accreditation and, again, loss of reputation for a cloud service provider.

Losses of governance. As data flows from service provider to service provider and beyond, problems can occur at the boundaries: the controls employed by a service provider can only directly ensure appropriate governance of data within the boundaries of that providers infrastructure. The primary cloud service provider within a service provision chain namely, the main cloud service provider in a chain, interacting directly with an enterprise customer loses control over data as it is handed over to that customer. If an entity with malicious intent gains control at the cloud service provider customer interface, this loss of governance on the part of the cloud service provider can have serious consequences for the confidentiality, integrity and availability of the customers data. Data provenance mechanisms, which are not restricted to a single cloud service provider might help to mitigate these problems [8].

Lock-in Hazards. Cloud service providers can create vendor lock-in issues for customers by forcing them to use particular formats for data. If those formats are not widely accepted, it may be very difficult to extract and convert the data for use further down the cloud service provision chain. A hazard can occur during an attempted conversion of data to another format particularly if the format in which the data is stored is encrypted, as such encryption is necessarily lost during the process, thus revealing the data to a potential attacker.

Isolation failures. In a multi-tenanted cloud environment, multiple customers data are stored on the same infrastructure by a cloud service provider; a standard contractual requirement in such a scenario is that isolation of different customers data and operations is maintained; in the absence of such isolation, attacks and hazards affecting one customer can affect another, due to interactions occurring on the common underlying infrastructure. Isolation failures can cause rapid propagation of viruses, worms and similar infections, affecting multiple customers data and damaging the cloud service providers reputation.

Incomplete data deletions. Data retention laws, typically, require cloud service providers to maintain customer data for a certain period of time after service has terminated. After this period has lapsed, the data has to be deleted from the cloud service providers infrastructure and, depending on the contractual terms applicable for the particular customer, disposed of using particular technical means. Failure to delete data in accordance with the relevant contractual terms can have serious consequences, and could even cause integrity issues for new customers using the same infrastructure if only partially overwritten.

Data protection problems such as the above are illustrative of issues that we need to instantiate in a simulation framework for accountability in the cloud.

B. Addressing Data Protection Problems: Controls for Accountability

While an accountability-based approach to data governance combines a number of mechanisms, ranging from high-level, legal obligations, all the way down to technical controls, our interest is in demonstrating just the latter namely, how technical measures, particularly automated tools, can be introduced into a cloud service providers infrastructure to address issues such as those presented in the previous section. As we have seen, we can classify controls into three categories, namely, preventive, detective, corrective depending on whether they are intended as measures to be deployed prior to or after a problem occurs.

Next, we describe the types of controls that we are modelling in the ASE framework.

Among preventive controls, we focus on **policy enforcement mechanisms**, in particular tools that allow organisations to ensure that pre-defined, machine-readable policies are enforced automatically within their information technology (IT) infrastructures. For the purposes of simulation, we will define **accountability policies** and the types of rules that may be encountered in such policies.

Detective controls usually take the form of background processes or aspects in a system; such controls can be active or passive, or some combination of the two. Active controls such as **monitoring** or **intrusion detection** react to particular events and patterns of behaviour, such as threats or data breaches. Passive controls include, for example, **logging tools**, whose function is to record all events that take place (with the source of the event, type of event, and other details) so that a service provider can (i) trace particular activities and identify sources of problems (this relates to attribution capabilities needed for accountability), (ii) prove compliance (with rules, regulations, standards, best practices and more) to external parties such as auditors. An example for such passive controls is Amazon's AWS CloudTrail [9], which provides cloud customers with an API call history and logs.

For corrective controls, there is a lack of previous work; mechanisms that are relevant are tools for providing **redress** to customers in cases where data protection problems have not been mitigated by preventive or detective controls. **Incident management tools** are relevant here, but exactly what remedies or responses are appropriate for different types of incidents remains an ongoing research challenge. For the purposes of simulation using ASE, we will assume that financial remedies (including **payment of fines** and other **penalties** for service providers) are suitable responses. The introduction of additional preventive measures, such as storage encryption depending on the type and sensitivity of stored data may also be a response. However, for the purpose of this paper, this is out of scope.

III. Review of Selected Current Simulation Tools and Platforms

Although we have developed the ASE simulation framework from the ground up, we have surveyed and experimented with a number of existing simulation tools; only discrete-event simulation tools have been considered, since our interest is in understanding behaviours and mechanisms that can be effectively modelled using this paradigm.

The tools of interest include software libraries providing dedicated simulation functionality, such as built-in data structures for event queues, random number generation using different probability distributions, timing information and more, as well as visual tools for designing simulations using predefined components.

Discrete-event simulation is detailed in the authoritative text by Law [10], which also includes a library for use in C programs, named *simlib*. This enables one to make use of commonly used data structures for simulation, as mentioned above. There are other libraries with similar capabilities, and indeed *simlib* has been rewritten and adapted for use in other programming languages (e.g., Brian J. Huffman has produced a Java version of the library [11]). We are also aware of the Java-based *Greenfoot* framework [12], which allows simulations to be prototyped easily; so far, we have not found a way to turn *Greenfoot* code into web-based applications, which was desirable for our purposes.

An interesting, more recent Java-based simulation library that we experimented with is the agent-based simulation framework MASON [13], which also includes graphical animation capabilities. The distinctive feature of MASON is that it allows one to produce animated graphical user interfaces to demonstrate interactions in multi-agent systems. Since the demonstrations we have been building are currently relatively small-scale, as opposed to its usual applications, we abandoned MASON early on. Nevertheless, its modular design and graphical capabilities may well be used in future versions of the accountability simulator.

Another approach that we considered included the use of industrial-strength visual simulation tools, such as MATLAB™-Simulink™ [14] and Simio™ [15]. Using the trial version of Simio™, we were able to produce a simple, 3D graphical animation of data flows between cloud service providers, as shown in Figure 1. We were not able to simulate accountability mechanisms using the trial version, as this would require building/coding a significant number of Simio™ processes, a feature that is limited. This will be included in our future work. However, we were able to gain visual insight into the nature and purpose of the simulation, which will be discussed in the next section.

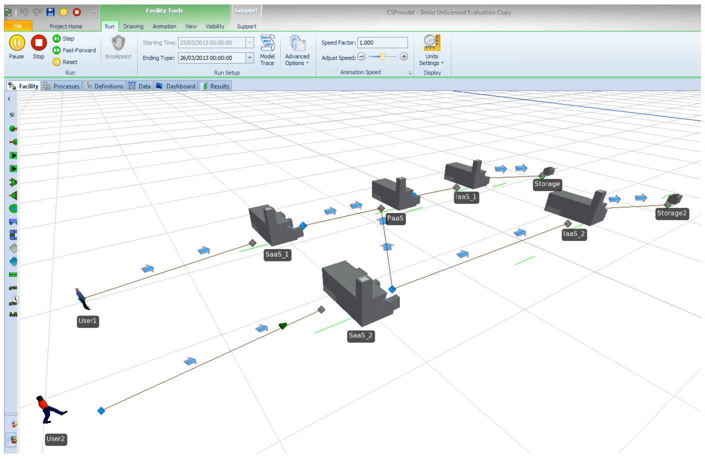


Figure 1. Screenshot of simulated data flows between cloud service providers

IV. MODELLING CONSIDERATIONS

As we have seen in previous sections, in order to build a simulation of accountability in the cloud, we need to identify a way to show (i) data protection problems that arise in cloud ecosystems, and (ii) how accountability controls or mechanisms work to mitigate and respond to these problems. The objective of this work is to build a graphical simulation which can provide insight for a variety of stakeholders, including cloud service providers, regulators, auditors and even the general public interested in how accountability can be achieved in a complex chain of cloud service provision. But what should be the underlying conceptual model of the simulation? There are different aspects to consider here.

A. *Static Modelling: Actors and Relationships*

One aspect to consider is the set of relationships (and the properties of these relationships) between different cloud service providers in a service provision chain. From the data protection point of view, there are different roles for cloud service providers when it comes to handling personal

TABLE I. THE POSSIBLE ROLES THAT THE DIFFERENT KINDS OF ACTORS CAN TAKE ON IN A GIVEN SCENARIO AS PER OUR MODEL.

Actor Type	Possible Roles
Individual	Data subject
Cloud service provider	Data controller Data processor
Third party	Data controller Data processor Accountability Agent
Auditor	(Auditor) Accountability Agent
Regulator	(Regulator) Accountability Agent

data terms used in the European Data Protection Directive 95/46/EC [7] for these roles are *data controller* and *data processor*. Depending on the service offering, providers may take one or both of these roles, with complex and ambiguous cases arising frequently. Modelling what this implies in terms of concrete obligations for cloud service providers is what we will refer to as static *modelling of accountability*.

The static modelling of a cloud service provision chain involves identified actors, roles and responsibilities and the relationships between them.

1) Actors, Roles and Responsibilities: In our model, in a cloud ecosystem there are five different kinds of actors **individuals, cloud service providers, third parties, auditors** and **regulators**. We classify the different types of roles that these actors may take on in a particular scenario into six kinds **data subject, data controller, data processor, accountability agent, auditor** and **regulator**. A particular scenario is defined as a specified set of roles for a specified set of actors.

The possible roles that the different kinds of actors can take on in a given scenario as per our model are defined in Table I.

The roles that we have included take into account the static modelling discussion in Section IV-A. *Accountability agent* represents a role that is intended to encompass internal oversight activities within an organization (e.g., self-auditing), as opposed to the roles of *auditor* (an external entity performing an audit on behalf of enterprise) and *regulator* (typically a government entity responsible for setting, implementing and monitoring standards), which by definition correspond to oversight external to an organization; note that we model two classes of organizations here cloud service providers and third parties, the latter being providers of non-cloud services. The distinction becomes clearer when we consider relationships that can exist between actors.

2) Relationships: Cloud service providers are characterized in the model by the kind of relationship they have with other providers, in particular, what kind of service they offer to others. A cloud service provider provides one of three kinds of service: **IaaS** (infrastructure-as-a-service), **PaaS** (platform-as-a-service), **SaaS** (software-as-a-service). These are the only kinds of relationships considered here between cloud service providers. Third parties are entities that enter into complex contractual relationships with cloud service providers, relationships that are not of the same kind. Further investigation is needed here; but, for the purposes of modelling and simulation we do not need to restrict the kinds of relationship that third parties may have (with each other and with cloud service providers).

B. Modelling System Dynamics: Data Transfers and Accountability Mechanisms

While static modelling would enable us to simulate what effect particular assumptions might have on the obligations of a cloud service provider, *modelling system dynamics* enables us to simulate data flows between cloud service providers, data protection problems and their consequences when accountability controls are in place (and similarly when such controls have not been introduced). For a dynamic simulation, the main entities that need to be modelled are *personal data*; at each step of the simulation, personal data flow through a chain/sequence of service providers, which are predefined, and together constitute a model of a real-world cloud service provision chain. The purpose of the simulation becomes to show what happens to the data as they flow through the chain, and what effect these flows have on properties of the overall system.

So, what is our model? The entities modelled have been discussed in the previous subsection, along with their relationships; next, we discuss their expected behaviours, and the types of issues or problems that can be simulated using our model, and the responses that different entities can have and should have to such problems if accountability is to be achieved.

1) Behaviours corresponding to different types of role: First, consider the behaviours of individual data subjects. In our model, a data subject is an entity that can engage in one of the following actions at any time during a simulation:

- Create data (a datum is modelled simply as a pair of strings an identifier and a value)
- Modify/edit data
- Delete data
- Change preferences regarding usage of data (initially, a data subjects policy is simply a statement of for what uses data can be processed, and whether the data can be shared with third parties)
- Request summary of data and preferences held

For service providers, which are typically data controllers or data processors, the following actions are possible:

- Store data
- Encrypt and store data
- Decrypt data
- Check preferences and share data
- Create new policy over data
- Generate log of activity over data
- Enable/disable logging mechanisms
- Enable/disable monitoring mechanisms
- Enable/disable policy enforcement mechanisms

Regulators and auditors are modelled as having the following possible actions:

- Check compliance of data controller/processor with a specified rule or set of rules
- Check compliance of organizational policies with minimum requirements
- Create new rules specifying allowed uses of customer data, and penalties/remedies in case of non-compliance or other problem
- Create new rules specifying mechanisms that must be used to protect data subjects, and penalties/remedies in case of non-compliance or other problem
- Enforce penalty or other remedy in case of non-compliance or other problem
- Audit a data controller/processors system logs (esp. check origin, route, destination of data; intended use; protection mechanisms used, whether customers preferences were enforced)

Accountability agents are initially to be modelled as a variant of auditor, with the only difference that they cannot perform enforcement, only (implicitly) inform the organization they are associated with of any events of interest (e.g., failures, non-compliance). Further work may reveal other actions/events of interest.

2) Simulated Issues: In the simulation, we should be able to represent and visualize some of the issues discussed previously in Section II.A. Compliance failures, data breaches and direct attacks on a service providers infrastructure are specific events that we have so far considered in this work.

3) Simulated Responses: In Table II, we can see how the different accountability mechanisms considered in our model can help to address the simulated issues. We note that this list is not exhaustive, as it only includes the mechanisms we have considered so far; other mechanisms could include, for example, automated tools for punishment or remediation; also, we have so far avoided detailing what types of rules are allowed in policies. In the Cloud Accountability Project, which is the context in which the simulation has been built, there is an ongoing work on developing an accountability policy language; for the purposes of our simulation, we have so far assumed that rules restrict to whom and under what conditions data can flow; the distinction between data controller and data processor may well imply additional restrictions, and similarly there are restrictions on when data can be transferred to third parties (this is modelled as a preference that data subjects can set).

As we can see from the table, when none of the accountability mechanisms are enabled in a simulation, none of the problems considered trigger any response (thus allowing hazards and failures to occur). Of interest is the fact that hazards can then propagate (cascade) from one provider to another, and/or to any third parties. All other cases cause a response, thus demonstrating how accountability mechanisms work in practice.

V. IMPLEMENTATION

We have implemented three software components as part of the accountability simulator: a simulation engine, a web-based animation of data flows between cloud service providers,

TABLE II. PROBLEM AND THE SPECIFIC RESPONSES TRIGGERED BY ACCOUNTABILITY MECHANISMS IN THE SIMULATION MODEL

Problem	Mechanism			
	None	Policy enforcement	Monitoring	Logging
Compliance failure	X	All problems correspond to specific policy violations; Policy violation will be detected; Parties notified	Patterns of non-compliance can be detected	All failures will be logged and shown to auditors
Data breach	X		Monitoring interaction of service providers with untrusted third parties can provide advance warnings	All breaches will be logged and shown to auditors
Attack	X		Intrusion detection systems can be used to thwart attacks	All attacks will be logged and shown to auditors

and a web service that draws data from the simulation engine. Currently, we are continuing implementation work until all three components have been fully integrated. In this section, we present the functional structure of the simulator and then detail each of the implemented components.

The input file is a description of a scenario to be simulated, and is written in the accountability model description language, described in the next section. A scenario consists of a specified set of actors (so far, we have not made the distinction between actors and roles in the language, but this is forthcoming in future versions), a specified set of relationships, configuration of options/parameters and the triggering of actions of particular actors.

When an input file is supplied to the simulator (via a web-based interface or through the command line), its contents are parsed using the language interpreter, which invokes appropriate methods in the accountability simulation engine. The accountability simulation engine contains the current state variables and the log of events executed so far; it constitutes the *backend* of the application and is written in plain Java.

In order to feed the state of the simulation, timings and outcomes to the web-based user interface, we have implemented a RESTful web service using the Java-based Restlet EE framework [16].

The initial version (designated *v1*) of the web-based user interface was implemented using HTML forms, and the data it receives from the web service consist of plain text strings that are displayed and updated as the simulation progresses, without any graphical animation.

The latest version (designated *v2*) of the web-based user interface has been developed separately, as a graphical animation, and work is ongoing to link it up to the web service. This will be discussed further in Section V-C below.

A. Accountability Model Description Language

Scenarios to be simulated are described by the user of the simulator in a custom modelling language we have built for this purpose. At this stage of development we have only included a core set of commands and mechanisms that can be included in scenarios, but we expect to add constructs in the language so as to allow inclusion of detailed privacy policies, access and usage controls, and other features. In particular, in the Cloud Accountability Project there is ongoing work on developing a dedicated accountability policy language, and it is likely that the constructs of that language will be incorporated into the language of the simulator.

Listing 1 shows the productions for the languages grammar, using the syntax of the SableCC [17] parser generator that we have been using to build the interpreter. The nonterminals in the grammar are *command* (for top-level commands), *declare* (used to declare actors of different types), *type* (representing the different actor types, namely user, cloud service provider, auditor and regulator), *servicetype* (for the different types of cloud service), *objectaction* (this is for expressions representing a property or action of a given actor), action (properties or actions), mode (for data protection problems that can be simulated using the trigger command), *mech* (for accountability mechanisms that can be enabled or disabled as needed). Commands *setgraph*, *setpolicy* and *setconstraint* are experimental; the command *graph* allows us to access the internal data structure of the accountability simulation engine and visualize it using AT&T GraphViz [18].

B. Accountability Simulator (Backend component)

The accountability simulation engine is responsible for maintaining and updating the current state of the simulation, and currently its main visible function is to display that state on the console or supply it (through a web service) to another application.

We can denote the internal state of the simulation engine by a tuple (see Equation 1)

$$(A, T, \rho, \sigma, M, \xi) \tag{1}$$

where A is the set of actors that have been declared, T is the set denoting the types of the actors, ρ is the set of relationships between cloud service providers (the only relationships modelled are between these types of actor), σ is the store of data values held by the different actors (indexed by A), M is the set of accountability mechanisms enabled and ξ is the output stream (this represents, e.g., the standard output or a pipe to another application, or a web service).

```
command =
{ declaration }  declare  |
{ custdeclaration } customer  lparen
          [ fst ]:  identifier   [q]:comma
          [snd]:  identifier   [z]:comma
          servicetype  rparen  |
{action}  objectaction  |
{ trig }  trigger  mode  identifier  |
{setgraph}  setgraph   arglist  |
{setpol}  setpolicy  lparen   identifier   comma
          str  rparen  |
{setcons}  setconstraint   lparen  [ fst ]:  identifier
          [q]:comma [snd]: identifier   [z]:comma
          str  rparen  |
{enablemech} enable mech |
{disablemech}  disable  mech |
{graphing} graph;

declare  = {declplain} type   identifier   |
           { declqualified } type   identifier    str ;

type =   {userdec} user  |
         {cspdec} csp  |
         {auddec} auditor  |
         {regdec}  regulator  ;

servicetype  = {iaastype} iaas  |
         {paastype} paas  |
         {saastype} saas ;

objectaction  = { actionref }   identifier   dot  action ;

action  = { actionsenddata } senddata
          lparen   identifier   comma str rparen  |
          { evalstate }  state ;

mode =   {databreach} databreach  |
         { attack }  attack ;

mech =   {polenfmech} polenf |
         {logmech} logging  |
         {monmech} monitoring;
```

Listing 1. SableCC grammar of the Accountability Simulator input language (only the main productions are shown)

The output of the simulation depends on which accountability mechanisms have been enabled; if no mechanisms are enabled (in which case the value of M above would be the empty set, $\emptyset$), then there is no change in the output ξ when a problem is triggered. However, when mechanisms are enabled and a problem is triggered, the effect (as described in Table II) is made visible on the output. In other words, the Function of the simulator (see Function 2) can be summarized operationally as a state transition of the form:

$$(A, T, \rho, \sigma, M, \xi) \rightarrow (A, T, \rho, \sigma', M, \xi') \qquad (2)$$

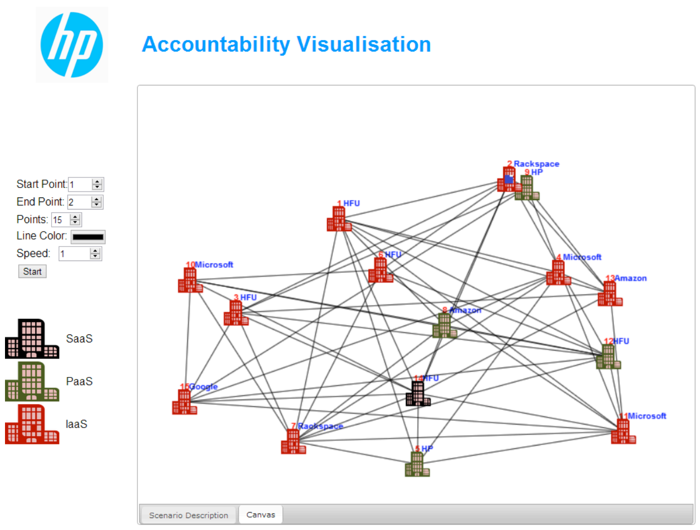
such that ξ' differs from ξ as it contains a notification of a compliance failure, data breach or attack when the trigger command is issued and one of the following is true:

$$(\{policyenforcement : enabled\} \in M) \ or$$
$$(\{logging : enabled\} \in M) \ or$$
$$(\{monitoring : enabled\} \in M)$$

It would not be difficult to use the above notation to derive a full operational semantics for the simulator, but for our purposes here it is sufficient to note that the main function of the tool, which is to behave differently depending on which type of problem is being simulated, and which mechanisms are enabled. So far we have assumed ξ represents textual output, namely strings describing the overall system state, such as lists of actors, their data values and more. Of course, the exact output consists of messages corresponding to the responses shown in Table II. In the next section we turn to work we have done on developing a graphical visualization.

C. Web-based Front-End

The vision for the web-based user interface has always been to have a graphical animation of data flows between individuals, cloud service providers, auditors and regulators and third parties. Demonstrating flows of data and the changes that occur to data and providers in the process emphasizes the dynamic aspect of the simulation. A screenshot of our current prototype of version v2 is shown in Figure 2.



Start Point: 1
End Point: 2
Points: 15
Line Color:
Speed: 1

Figure 2. Visualisation front-end for the Accountability Simulator

It is very important to note that this version has been developed as a separate, standalone animation. Thus, it has not yet been linked to the web service component and, hence, the main simulation engine. However, it does show an instantiation of a random set of cloud service providers of different kinds (IaaS, PaaS, SaaS, and how a data item can be routed between providers. In the animation, the scenario is assumed to be random, rather than specified in the accountability modelling language; this is changing presently.

We expect to have dedicated controls (form buttons) to trigger particular data protection problems, and panels showing the responses produced by the simulator. In the screenshot in Figure 2 two tabs are shown at the bottom. While the (currently random) animation is shown on the *Canvas* tab, the other tab (titled *Scenario Description*) will allow the user of the simulation to supply an input file in future, written in the accountability modelling language of Section V-A, and this will be used to generate the animation in the next version.

VI. An Example Input File

Listing 2 shows an example input file that we have tested with the current version of the simulator. It describes a scenario in which there are two individual users, four cloud service providers, an auditor and a regulator. The relationships between the individuals and service providers are then declared. John and Mary then create some data, which is sent and stored on the specified service providers infrastructure. In line 15 a data breach problem is triggered; this has no effect when simulated as no accountability mechanisms have been enabled; the remaining lines enable different mechanisms, and trigger a data breach and attack. Naturally, the simulator produces a sequence of long warnings when interpreting lines 17, 20 and 22, as the policy enforcement, monitoring and logging mechanisms kick in.

```
User john ”John Wayne”;
User mary ”Mary Wollstonecraft”;
CSP salesforce ”Salesforce .com”;
CSP amazon ”Amazon Web Services”;
CSP rackspace ”Rackspace”;
CSP hpcs ”HP Cloud Services”;
Auditor kpmg ”KPMG”;
Regulator cnil ”CNIL”;
Customer(john,amazon,SaaS);
Customer(mary, salesforce ,SaaS);
Customer(rackspace,hpcs,IaaS);
Customer( salesforce ,rackspace,PaaS);
john.Senddata(amazon,”somedata”);
mary.Senddata(hpcs,”marydata”);
Trigger Databreach salesforce ;
Enable Polenf;
Trigger Databreach salesforce ;
Enable Monitoring;
Enable Logging;
Trigger Databreach salesforce ;
amazon.State;
Trigger Attack amazon;
```

Listing 2. Example script written in the accountability simulation language.)

VII. Conclusion and Future Work

In this paper, we have presented the design and implementation of a simulator for accountability mechanisms in the cloud. We have discussed data protection problems, and how mechanisms for accountability such as policy enforcement, monitoring and logging can help to address such problems; the simulator we have built is a tool to assist understanding and modelling of real-world scenarios and will hopefully be a useful aid to cloud service providers, regulators and end users as it is extended with more features.

Future work will focus on integrating the v2 web-based UI with the accountability simulation engine and web service, and enriching that UI with more controls. Subsequently we will work on animating the accountability mechanisms and modelling additional ones.

It is also worth noting that a new EU Data Protection Regulation will eventually replace the current directive, which is under discussion in the European Parliament. This is likely to include new accountability rules and obligations, and must be taken into consideration in future work. For the purposes of this paper, however, we have focused on modelling the dynamics of accountability controls and how they impact data and data flows in cloud infrastructure.

Acknowledgment

Nick Papanikolaou wishes to cordially thank Siani Pearson and Nick Wainwright for their support and feedback during the development of this work. Special thanks are due to Fabian Reich, who spent a month at HP Labs working with Nick Papanikolaou and actually coded version v2 of the web-based GUI, navigating the vagaries of the HTML5 Canvas control. This work is supported by the European FP7 Programme A4CLOUD: Accountability for the Cloud and Other Future Internet Services.

References

[1] “Cloud accountability project (a4cloud),” http://www.a4cloud.eu/, [retrieved: 2014.04.08] 2014.

[2] S. Pearson, “Toward accountability in the cloud,” Internet Computing, IEEE, vol. 15, no. 4, pp. 64–69, July 2011.

[3] Centre for Information Policy Leadership as Secretariat to the Galway Project, “Data protection accountability: The essential elements - a document for discussion,” http://www.informationpolicycentre.com/files/Uploads/Documents/Centre/Centre_Accountability_Compendium.pdf, [retrieved: 2014.04.08] 2009.

[4] S. Pearson and N. Wainwright, “An interdisciplinary approach to accountability for future internet service provision,” International Journal of Trust Management in Computing and Communications, vol. 1, no. 1, pp. 52–72, 01 2013.

[5] T. Haeberlen, L. Dupre, D. Catteddu, and G. Hogben, “Cloud computing: Benefits, risks and recommendations for information security,” enisa - European Network and Information Security Agency, Tech. Rep., 2012.

[6] “Apec privacy framework,” http://publications.apec.org/publication-detail.php?pub_id=390, [retrieved: 2014.04.08] 2005.

[7] “Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data,” http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML, [retrieved: 2014.04.08] 1995.

[8] O. Q. Zhang, M. Kirchberg, R. K. L. Ko, and B. S. Lee, “How to track your data: The case for cloud computing provenance,” HP Labs, Tech. Rep., 2012.

[9] “Amazon aws cloudtrail,” https://aws.amazon.com/de/cloudtrail/, [retrieved: 2014.04.08].

[10] A. Law, Simulation Modeling and Analysis (McGraw-Hill Series in Industrial Engineering and Management). McGraw-Hill Science/Engineering/Math, 2006.

[11] B. J. Huffman, “An object-oriented version of simlib (a simple simulation package),” Informs Transactions on Education, vol. 2, no. 1, pp. 1–15, 2001.

[12] M. Koelling, Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations. Pearson Education, 2009.

[13] “Mason multiagent simulation toolkit,” http://cs.gmu.edu/~eclab/projects/mason/, [retrieved: 2014.04.08].

[14] “Matlab and simulink,” http://www.mathworks.co.uk/products/simulink/, [retrieved: 2014.04.08].

[15] “Simio,” http://www.simio.com/, [retrieved: 2014.04.08].

[16] “Restlet framework,” http://restlet.org, [retrieved: 2014.04.08].

[17] E. Gagnon, “Sablecc, an object-oriented compiler framework,” http://www.sablecc.org/, [retrieved: 2014.04.08].

[18] “Graphviz – graph visualization software,” http://www.graphviz.org, [retrieved: 2014.04.08].

Mobile Cloud and Grid Web Service in a Smart City

Jong Won Park, Chang Ho Yun, Hae Sun Jung, Yong Woo LEE (Corresponding Author)
School of Electrical & Computer Engineering
The Smart City (Ubiquitous City) Consortium, the University of Seoul
Seoul, South Korea
emails: {comics77, touch011, banyasun, ywlee}@uos.ac.kr

***Abstract*— Smart City is a future city that heavily utilizes information and communication technology and allows the users to use the smart city services anytime, anywhere and with any accessing devices. In our smart city called UTOPIA, which has 3 tiers architecture, such as the smart city infra, the smart city middleware called SOUL and the smart city portal, administrators as well as citizens, can use smart phones, tablet personal computers, and other modern mobile devices to utilize smart city services through the smart city portal. Since big data are usually processed in real-time to provide smart city services, it is not easy at all to give the mobile cloud and grid web services in a smart city. In this paper, we introduce our mobile cloud and grid web services in a smart city, show some venues and explain the mechanisms of the cloud and grid web service in SOUL and UTOPIA.**

Keywords-Smart (Ubiquitous) City; Cloud Computing; Grid Computing; Web-Service.

I. INTRODUCTION

Recent advances in information and communication technology has enables us to build smart cities, future cities which are converged outputs of modern information and communication technology, urban engineering technology and other modern science and engineering technologies.

In the smart city, citizens do not feel alone and are not isolated in a huge modern city anymore but become a real owner of the city by being provided by the rich smart city service. Indeed, the smart city is toward the city which is of the citizen, by the citizen and for the citizen.

Ubiquitous city is a kind of smart city and tries to give users freedom of using the smart city service by allowing user to access the smart city service anytime, anywhere and with any accessing devices. Thus, administrators as well as citizens should be able to use smart phones, tablet personal computers, and other modern mobile devices to enjoy smart city services in the smart city.

Smart city covers a wide range of areas, such as environment, accident, facility, etc., deals with huge volumes of data usually and analyzes or/and converges a wide range of data to create meaningful information often in real-time. All these requirements are not easy at all to be satisfied and really challenging tasks. We have used state-of-the-art cloud computing and Grid computing technology to find our solutions.

We have been developing a paradigm of the smart city called UTOPIA [1]. It has a unique 3 tiers architecture and consists of the smart city infrastructure, the smart city middleware called SOUL and the smart city portal [1][2][3][4].

UTOPIA is based on a unique smart city Middleware called SOUL, which does brain role of the smart city; thus, we call it soul. It has an important part of the solutions for the mobile cloud and grid web services in UTOPIA. It manages the big data to provide smart city services in the environment of heterogeneous hard-wares, heterogeneous operation systems, and heterogeneous networks in UTOPIA.

UTOPIA provides user friendly web portal which is connected to SOUL, the smart city middleware [2][3]. For example, a user can use two dimensional or three dimensional visualization services in the portal. The portal supports a variety of smart phones. The wide and popular use of mobile phone brings the era of mobile service.

Grid computing has brought us the era of cloud computing and now mobile cloud computing is a very attractive state-of-the-art technology. With mobile cloud computing, smart phones and tablet personal computer are now expanding their territories beyond their limitations in computing power, data storage and software [5].

UTOPIA enables users to use services anytime, anywhere and with any device, thus, supports mobile devices and provides mobile cloud and Grid web services.

This paper introduces our mobile cloud and grid web services in UTOPIA, shows some venues and explains the architecture and principle of UTOPIA and SOUL and the mechanisms of the cloud and grid web service in SOUL.

This paper is organized as follows. Section 2 introduces UTOPIA and SOUL. Section 3 presents the Mobile Cloud and Grid Web Services. Section 4 explains the architecture and operational principle of the cloud and grid computing platform. Section 5 explains related works and compares them with our work. Finally, Section 6 gives the conclusion.

II. UTOPIA

The Smart City Consortium for Seoul has been leading the five million smart city project funded and supported by Seoul Metropolitan Government of Korea since 2005. It includes SK Telecom, LG CNS, etc., as industry members, and the laboratories from many well-known Korea Universities in Seoul as academic members.

We have developed a smart city paradigm called UTOPIA as shown in Figure 1. UTOPIA consists of 3 tiers, such as the smart city portal tier, the smart city middleware tier and the smart city infrastructure tier [1].

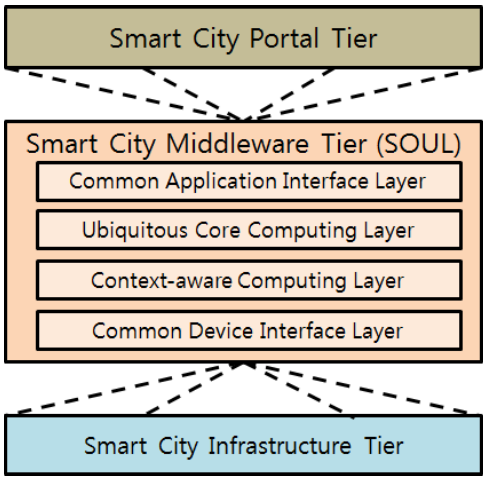

Smart City Portal Tier
Smart City Middleware Tier (SOUL)
Common Application Interface Layer
Ubiquitous Core Computing Layer
Context-aware Computing Layer
Common Device Interface Layer
Smart City Infrastructure Tier

Figure 1. The three tier paradigm of UTOPIA for the Smart City.

The Smart City Infrastructure Tier includes sensors networked through USN, video cameras, GPS and appliances, and so on. It plays the role of a human body in a human being. The Smart City Middleware Tier, which we call SOUL, plays the role of soul or brain in a human being. The smart city portal tier provides web services and enables us to use mobile cloud and grid web service. It makes the other two layers transparent to users since users do not have to know their internals.

III. Mobile Cloud and Grid Web Services

UTOPIA provides many intelligent services for the various kinds of applications. Some typical mobile cloud and grid web services are introduced in this section, as examples in UTOPIA.

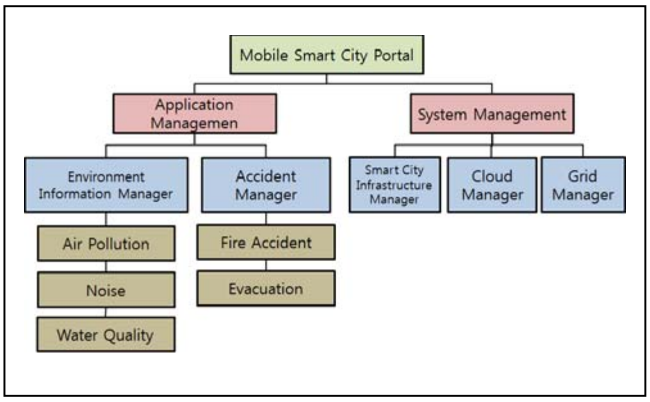


Figure 2. Typical Mobile Cloud and Grid Web Services in UTOPIA.

Some typical mobile cloud and grid web services are shown in Figure 2. The class of the mobile cloud and grid web service is largely divided into two parts. There are “Application Management” which is classified into the “Environment Information Manager” and the “Accident Manager”, and “System Management” which is classified into the “Smart City Infrastructure Manager”, “Cloud Manager”, and “Grid Manager”.

“Application Management” is an interface to deal with applications in SOUL. Environment Information Manager of Application Management System displays air pollution information, noise information and water quality information by using 2D/3D visualization functions in SOUL. “Accident Manager” of “Application Management” can provide fire accident management information, such as an evacuation path for emergency due to fire accident.

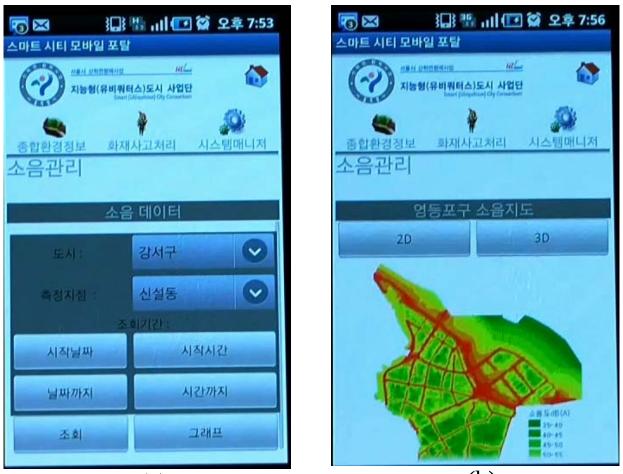
“System Management” consists of “Smart City Infrastructure Manager”, “Cloud Manager” and “Grid Manager”. “Infrastructure Manager” is a system component of SOUL to manage smart city infrastructure, such as Ubiquitous Sensor Network (USN), scalable video streaming, ad-hoc networking. Privileged users can control them through mobile web service. SOUL includes the cloud platform which is implemented with OpenNebula for big data processing in UTOPIA [6]. “Cloud Manager” is used to manage the “Cloud platform: of SOUL [5]. SOUL supports Grid resource management to process big data in real time with distributed computing resources. The Grid resource management system contains real time performance monitoring subsystem, process management subsystem with resource brokering, account management subsystem, unified file transfer management subsystem and access Grid management subsystem [7]. “Grid Manager” in “System Management” provides user interface to the Grid resource management system in SOUL [8][9].

A. *Environment Information Manager*

The “Environment Information Manager” shows noise information, air-pollution information and water quality information which are converged with GIS (Geographical Information System) map. Figure 3a shows snapshots of “Environment Information Manager”.

“Noise” menu shows noise data and the 2D/3D noise map of a selected specific area that is generated by SOUL. To generate the noise map, SOUL uses cloud computing platform and Grid Computing platform. For the cloud computing, MapReduce is used [10][11][12]. The noise map can be visualized in 2D and 3D. As shown in Figure 3a and 2b, a user can select a specific zone and would see the noise level data and the noise map. The noise map shows the noise level distribution with specific color marks according to the predefined noise level which can be adjusted by users.

“Air pollution” menu shows the air pollution data and the 2D/3D air pollution map. SOUL also uses cloud computing platform which uses MapReduce in order to process the massive air pollution data and generate the air pollution map [13]. SOUL uses the 3D GIS data model that is based on the 2D GIS topology in order to make the 3D air pollution map. It enables users to use 3D spatial queries, analyses as well as the 3D visualization. A user can also select a specific zone for the visualization of the air pollution map and would see the air pollution level data and the air pollution map. The air pollution map shows the air pollution level distribution with specific color marks according to the predefined air pollution level which can be adjusted by users.



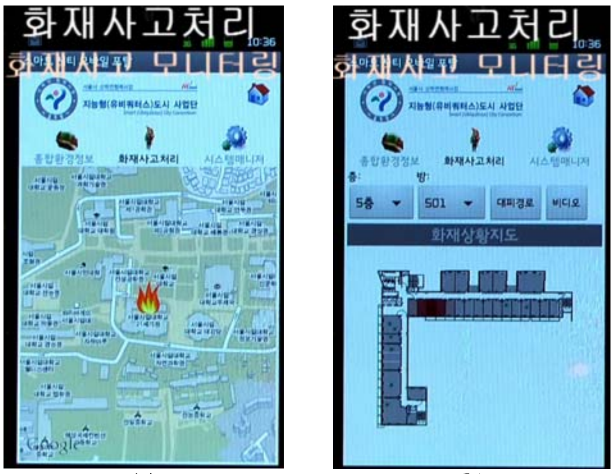
(a) (b)

Figure 3. Snapshots of the Environment Information Manager of UTOPIA in a smartphone : (a) A Noise Monitoring, (b) A 2D/3D Noise Map.

"Water Quality" menu supports tele-monitoring and tele-management [14][15]. Administrators of the smart city can monitor quality and quantity of water, observe the circumference in the canals, the waterways, the rivers, and the dams. They can control the water pumps, gates of canals, waterways, and dams remotely through mobile cloud and grid web service.

B. *Fire Accident Manager*

"Fire Accident Manager" informs administrator and citizen of UTOPIA the occurrence of fire event and optimal evacuation paths, and calls the most appropriate internal services of UTOPIA among services available at the situation.



(a) (b)

Figure 4. Snapshots of the Fire Accident Managerof UTOPIA in a smartphone: (a) A GIS map provided by the Fire Accident Monitoring, (b) An Evacuation path provided by the Fire Accident Monitoring.

Sensors report the data through ubiquitous sensor network to SOUL and SOUL infers the fire event using the sensed data [16]. When a fire event is inferred, SOUL calls context-aware services and sends the administrator the fire alarm through the mobile cloud and grid web service.

"Fire Accident Manager" shows the fire accident place in the GIS map as shown in Figure 4a. Figure 4b shows the current fire accident location in the fire accident building and an optimal evacuation route from a selected room in the building to exit safely.

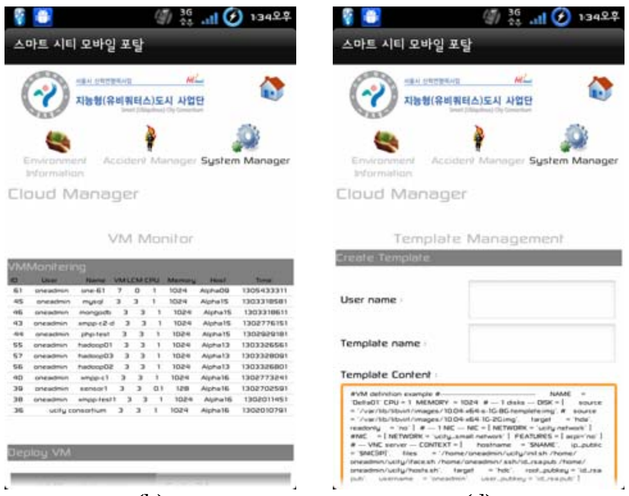
C. *Cloud Manager*

UTOPIA Mobile cloud and grid web service is the world's first approach to implement the cloud and grid platform into a smart city middleware so that the users can monitor the smart city and execute smart city services in Android based smartphone or tablet personal computers [5].

The Mobile cloud and grid web service is supported by Cloud Infrastructure Manager, Virtual Machine Job Manager, SSH client and vnc client. Cloud Infrastructure Manager plays the role of VM monitoring, VM template management, hosts management, users management and networks management. Figure 4 shows how it works in a smartphone in two cases.

Cloud Manager has sub-menus, such as VM Monitor, Template Management, Hosts Management, Networks Management and Users Management.

VM Monitor can show the state of virtual machine instances, enable users to deploy virtual machine instances in virtual machine template and to remove virtual machine instance in "Cloud Platform" of SOUL and provides vnc and SSH connection interfaces to access virtual machine instance.



VMMonitering

ID	User	Name	VM	LCM	CPU	Memory	Host	Time
61	oneadmin	one-61	7	0	1	1024	Alpha09	1305433311
45	oneadmin	mysql	3	3	1	1024	Alpha15	1303318581
46	oneadmin	mongodb	3	3	1	1024	Alpha15	1303318611
43	oneadmin	xmpp-c2-d	3	3	1	1024	Alpha15	1302776151
44	oneadmin	php-test	3	3	1	1024	Alpha15	1302929181
55	oneadmin	hadoop01	3	3	1	1024	Alpha13	1303326561
57	oneadmin	hadoop03	3	3	1	1024	Alpha13	1303328091
56	oneadmin	hadoop02	3	3	1	1024	Alpha13	1303326801
40	oneadmin	xmpp-c1	3	3	1	1024	Alpha16	1302773241
39	oneadmin	sensor1	3	3	0.1	128	Alpha16	1302702591
38	oneadmin	xmpp-test1	3	3	1	1024	Alpha16	1302011451
36	ucity consortium		3	3	1	1024	Alpha16	1302010791

User name :

Template name :

Template Content :

```
#VM definition example #———————————— NAME =
'Delta01' CPU = 1 MEMORY = 1024 # — 1 disks — DISK = [ source
= '/var/lib/libvirt/images/10.04-x64-s-1G-8G-template.img', # source
= '/var/lib/libvirt/images/10.04-x64-1G-20.img', target = 'hda',
readonly = 'no' ] # — 1 NIC — NIC = [ NETWORK = 'ucity network' ]
#NIC = [ NETWORK = 'ucity_small network' ] FEATURES = [ acpi='no' ]
# — VNC server — CONTEXT = [ hostname = '$NAME', ip_public
= '$NIC[IP]', files = '/home/oneadmin/ucity/init.sh /home/
oneadmin/ucity/iface.sh /home/oneadmin/.ssh/id_rsa.pub /home/
oneadmin/ucity/hosts.sh', target = 'hdc', root_pubkey = 'id_rsa
pub', username = 'oneadmin', user_pubkey = 'id_rsa.pub' ]
```

(b) (d)

Figure 5. The snapshot of Cloud Manager of mobile cloud and grid web service of UTOPIA in a smartphone: (a) VM Monitor, (b) Templet management.

Figure 5a shows the user interface of VM Monitor. Deployment of virtual machine instances is made by pre-defined template image. Template Management manages description of template images.

Figure 5b shows the user interface of Template Management. Hosts Management manages hosts in the cloud platform of SOUL. It can create and delete a host and enable and disable the given host.

D. Grid Manager

The Grid Manager consists of many components, such as real time performance monitoring, process management with resource brokering, account management, unified file transfer management and access Grid management. These components inherited the Grid resource manager of Seoul Grid Portal [17]. Figure 6 shows the snapshot of Grid Manager of mobile cloud and grid web service in a smartphone.

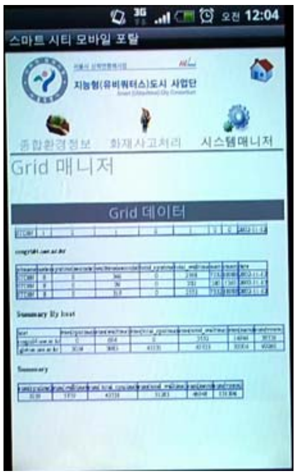
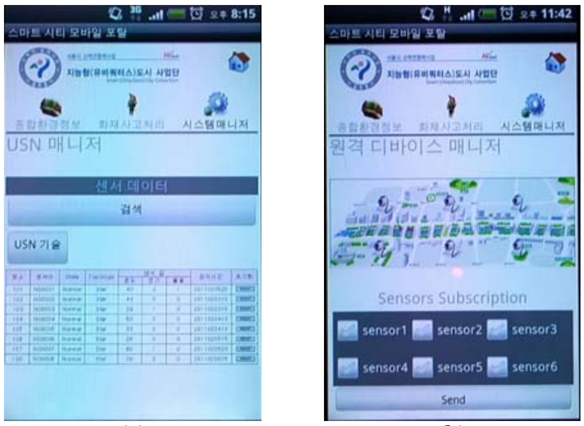


Figure 6. The snapshot of Grid Manager of mobile cloud and grid web service of UTOPIA in a smartphone.

E. Smart City Infrastructure Manager

“The Smart City Infrastructure Manager” enables the administrator of UTOPIA to do Ubiquitous Sensor Network (USN) Management, Scalable Video Streaming, Ad-hoc Network Management, Network Management, and so on, in SOUL with a smartphone or a tablet personal computer.



(a) (b)

Figure 7. The snapshot of operation of components in the Infrastructure Manager in a smartphone: (a) USN Manager, (b) Remote Device Manager.

Figure 7a shows how the USN Manager monitors USN in a smartphone. The USN Manager provides the administrator of UTOPIA with information, such as sensor ID, USN ID, USN topology, sensor working history, MAC address, IP address and so on of each sensor so that the administrator can manage these devices.

Figure 7b shows the operation of the Remote Device Manager in a smartphone. Functions of SOUL to control remote devices in a smartphone are based on Grid technologies, such as those in Globus Tele Control Protocol (GTCP) and Globus Toolkit 4 [18][19].

IV. CLOUD AND GRID COMPUTING PLATFORM

SOUL is an intelligent ubiquitous middleware which manages the Smart City Infrastructure Tier, offers smart city services, make a smart decision and give support the smart city portal tier. It is composed of four layers, such as Common Device Interface Layer, Context-aware Computing Layer, Ubiquitous Core Computing Layer and Common Application Interface Layer.

The Common Device Interface Layer (CDIL) provides the common device interface to the devices in the Smart City Infrastructure Tier. The Context-aware Computing Layer (CCL) processes data obtained from the CDIL and provides the intelligent context service to The Ubiquitous Core Computing Layer (UCCL). It does the intelligent reasoning. The UCCL is the core layer and has the cloud and Grid computing platform that provides various kind of cloud and Grid services. The Common Application Interface Layer (CAIL) support applications and provides a set of applications, such as the Environment Management, the Fire Accident Management, the Traffic Accident Management and so on.

The 4 layers cooperate together so that SOUL can provide various services. The administrator can control SOUL through the Smart city portal which has user friendly interface and provides web services. Furthermore, we added mobile and Grid web service providing facilities for the user who uses Android based smartphones and tablet personal computers [5]. Thus, users of UTOPIA can control and manage the SOUL and the smart city infrastructure tier easily, comfortably, conveniently and efficiently.

The mobile cloud and grid web service requires the cooperation of Cloud Manager, Virtual Machine Job Manager, SSH client and vnc client.

The Cloud Manager is the interface in the smart city portal and enables the smart city administrator to manage the cloud computing platform. It performs VM Monitoring, VM Template management, host management, user management and network management. Figure 8 shows the operation of the Cloud Manager.

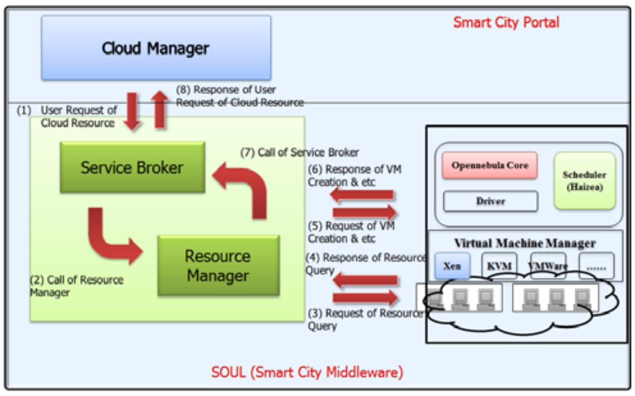

Smart City Portal
Cloud Manager
(8) Response of User Request of Cloud Resource
(1) User Request of Cloud Resource
Service Broker
(7) Call of Service Broker
(6) Response of VM Creation & etc
(5) Request of VM Creation & etc
(4) Response of Resource Query
(3) Request of Resource Query
Resource Manager
(2) Call of Resource Manager
Opennebula Core
Driver
Scheduler (Haizea)
Virtual Machine Manager
Xen
KVM
VMWare
......
SOUL (Smart City Middleware)

Figure 8. The cooperation of Cloud Manager in the Smart City Portal of UTOPIA with SOUL.

When a user requests the cloud resource to Cloud Manager, the Service Broker which is a gateway to the UCCL of SOUL processes the request. It calls the Resource Manager which manages the computing resource and cloud and grid platform in SOUL.

In order to find computing resources to respond to the user request, the Resource Manager queries a request to the Cloud Computing Platform and receives the result through XML-RPC API of OpenNebula. The Resource Manager sends the user request such as creation and deletion of virtual machine instance to the Cloud and Grid Computing Platform based on the result. The Cloud and Grid Computing Platform processes the user request and returns the response to the Resource Manager. The Resource Manager sends the result to the Service Broker. Finally, the user can see the result through Cloud Manager.

Virtual Machine Job Manager (VMJM) uses MVC model which makes the maintenance and extension of code easier and reusability better. VMJM executes web service, monitors web service, etc.

The web service can be executed only by authorized users and administrators. The user request is passed to the Resource Manager through the Service Broker. The Resource Manager checks the existence and status of the pre-created virtual machine instances. If the pre-created virtual machine instances exist, they are resumed by the Resource Manager. Otherwise, the Resource Manager creates a virtual machine from the suitable template. When the virtual machine instances are ready, the Service Manager executes the requested service on the virtual machine instances and sends the status information of the executed service to the Service Broker. Finally, the Service Broker sends the information and the user can see it through VMJM.

The operation of the monitoring web service is as follows. The user sends the service request to the Service Broker. The Service Broker passes the request to the Resource Broker. The Resource Broker requests the service list and the service status to the Cloud Computing Platform and returns the result to the Service Broker. The user can see the monitoring information through the Service Broker and the VMJM.

V. RELATED WORKS

The mobile cloud and Grid web service in smart city system, which has the smart city middleware based three tier paradigm, is unique since we cannot find any similar work like us.

Mobile cloud computing seems to be attractive since it can eliminate the constraints of weakness in computing power in mobile devices. H. T. Dinh at el. [20] defines that mobile cloud computing is a combination of mobile computing and cloud computing. Mobile cloud computing can have a variety of service model [21].

C. Doukas et al. [22] proposed @HealthCloud which is a pervasive health care information management system for mobile client system that has electronic healthcare data storage, update function and retrieval function using cloud computing. Its mobile cloud computing does not provide cloud management and job management on virtual machine through mobile devices.

E. E. Marinelli [23] introduced Hyrax that is a mobile-cloud computing platform. It uses Hadoop and runs in Android. It has a mobile cloud computing platform.

J. H. Christensen [24] proposed development methodology for smart mobile applications using Cloud computing and RESTful web service. The convergence of cloud computing and RESTful web service is similar to ours but it is the difference that ours is cooperatively operated with other functions in a smart city middleware in a smart city paradigm.

Eucalyptus [25], Globus Nimbus [26], OpenNebula [6] are open source cloud tools to construct and manage the cloud platform. They provide web interface. They are focused on managing cloud infrastructure and do not provide web services for the job management on the virtual machine. However, we provide the web services. Our mobile cloud and grid web service can execute and monitor smart city services on virtual machines anytime and anywhere through Cloud Manager and Grid Manager.

VI. CONCLUSION AND FUTURE WORK

This paper introduces mobile cloud and grid web services in a smart city paradigm called UTOPIA. UTOPIA has the 3 tier architecture, such as the smart city portal tier, the smart city middleware tier called SOUL and the smart city infrastructure tier.

SOUL has four layers, such as Common Device Interface Layer, Context-aware Computing Layer, Ubiquitous Core Computing Layer and Common Application Interface Layer. And the cloud and Grid computing platform belongs to Ubiquitous Core Computing Layer.

The mobile cloud and grid web services, such as Environment Information Manager, Fire Accident Manager, Infrastructure Manager, Cloud Manager and Grid Manager were explained. We showed that they can be serviced in Android smartphone or tablet personal computers. How the tree tiers of UTOPIA cooperatively support them was described and the internals of the cloud and Grid computing platform as well as the principle and architecture of UTOPIA and those of SOUL were explained. It is shown that the cloud and Grid computing platform, Cloud Manager and Grid Manager process big data in UTOPIA efficiently, easily and conveniently. We have a future plan to enrich the mobile cloud and grid web services in UTOPIA both in variety and quality by adapting new information technologies.

ACKNOWLEDGMENT

We would like to give thanks to Mr. Cheol Sang Yoon, Mr. Kyoung-Gyu Lee, Mr. Eui Dong Hwang and the staffs of Seoul Grid Center and the members of Smart (Ubiquitous) City Consortium for their contribution to this research. This work was supported by the 2011 research fund of the University of Seoul (Yong Woo LEE : the corresponding author). Patents in Korea and in the United State of America are registered and are in pending for the contents of this paper.

REFERENCES

[1] H. S. Jung, C. S. Jeong, Y. W. Lee, and P. D. Hong, "An Intelligent Ubiquitous Middleware for U-City: SmartUM," Journal of Information Science and Engineering, vol. 25, Issue 2, Mar. 2009, pp. 375-388.

[2] S. W. Rho, C. H. Yun, and Y. W. Lee, "Provision of U-city web services using cloud computing," Proc. 13th International Conference on Advanced Communication Technology (ICACT 11), Feb. 13-16 2011, pp. 1545-1549.

[3] Y. W. Lee and S. W. Rho, "U-city portal for smart ubiquitous middleware," Proc. The 12th International Conference on Advanced Communication Technology (ICACT 10), Feb. 7-10 2010, pp. 609-613.

[4] J. W. Park, C. H. Yun, S. W. Rho, Y. W. Lee, and H. S. Jung, "Mobile Cloud Web-Service for U-City," Proc. International Conference on Cloud and Green Computing (CGC 2011), Dec. 12-14 2011, pp. 1161-1065.

[5] N. Fernando, S. W. Loke, and W. Rahay, "Mobile cloud computing: A survey," Future Generation Computer Systems, vol. 29, Issue 1, Jan. 2013, pp. 84-106.

[6] OpenNebula Homepage, [online], Jan. 2014, Available from: http://opennebula.org/.

[7] Y. W. Lee and P. D. Hong, "A Job Management System for a Large Scale Grid System," Proc. IEEE International Conference on Computer and Information Technology (ICCIT 07), Oct. 16-19 2007, pp. 291-294.

[8] Y. W. Lee, "Seoul Grid Portal: A Grid Resource Management System for Seoul Grid Testbed," Proc. Grid and Cooperative Computing (GCC 2004), Oct. 21-24 2004, LNCS, vol. 3251, Sep. 2004, pp. 899-902.

[9] P. D. Hong and Y. W. Lee, "Web Service for Seoul Grid Testbed," Proc. IEEE International Conference on Computer and Information Technology (ICCIT 06), Sept. 2006, pp. 63-68.

[10] J. W. Park, C. H. Yun, S. G. Kim, H. Y. Yeom, and Y. W. Lee, "Cloud computing platform for GIS image processing in U-city," Proc. 13th International Conference on Advanced Communication Technology (ICACT 2011), Feb. 13-16 2011, pp. 1151-1155.

[11] J. W. Park et al., "Cloud Computing for Online Visualization of GIS Applications in Ubiquitous City," Proc. Cloud Computing 2010, Nov. 21-26, 2010, pp. 170-175.

[12] H. K. Park, Y. W. Lee, S. I. Jang, and I. P. Lee, "Online visualization of Urban Noise in Ubiquitous-city Middleware," Proc. The 12th International Conference on Advanced Communication Technology (ICACT 2010), Feb. 7-10, 2010, pp. 268-271.

[13] J. W. Park, C. H. Yun, H. S. Jung, and Y. W. LEE, "Visualization of Urban Air Pollution with Cloud Computing," Proc. IEEE World Congress on Services (SERVICES 2011), July 4-9, 2011, pp. 578-583,.

[14] H. Han et al., "Management of remote facilities through a ubiquitous grid middleware," Proc. The 11th International Conference on Advanced Communication Technology (ICACT 2009), Feb. 15-18, 2009, pp. 2291-2294.

[15] C. H. Yun, H. Han, H. S. Jung, H. Y. Yeom, and Y. W. Lee, "Intelligent Management of Remote Facilities through a Ubiquitous Cloud Middleware," Proc. IEEE 2009 International Conference on Cloud Computing (2009 CLOUD-Ⅱ), Sept. 21-25. 2009, pp. 65-71.

[16] C. H. Yun, Y. W. Lee, and H. S. Jung, "An evaluation of semantic service discovery of a U-city middleware," Proc. The 12th International Conference on Advanced Communication Technology (ICACT 2010), Feb. 7-10, 2010, pp. 600-603.

[17] P. D. Hong and Y. W. Lee, "A Grid portal for grid resource information service," Proc. The 13th International Conference on Advanced Communication Technology (ICACT 2011), Feb. 13-16 2011, pp. 597-602.

[18] The Globus toolkit, [online], Jan. 2014, Available from: http://www.globus.org/tookit/

[19] I. Foster. "Globus Toolkit Version 4: Software for Service-Oriented Systems," Proc. The 2005 IFIP international conference on Network and Parallel Computing (NPC 05), Nov. 30-Dec. 3 2005, pp. 2-13.

[20] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," Wiress Communications and Mobile Computing, vol. 13, 2013, pp. 1587-1611.

[21] D. Kovachev, Y. Cao, and R. Klamma, "Mobile Cloud Computing: A Comparison of Application Models," International Journal of Engineering Technology & Management Research, Vol. 1, Issue. 1, Feb. 2013, pp. 20-25.

[22] C. Doukas, T. Pliakas, and I. Maglogiannis, "Mobile healthcare information management unitizing cloud computing and Android OS," Proc. In Annual International Conference of the IEEE on Engineering in Medicine and Biology Society (EMBC 2010), Aug. 31-Sept. 4 2010, pp. 1037-1040.

[23] E. E. Marinelli, "Hyrax: Cloud Computing on Mobile Devices using MapReduce", Master Thesis, Department of Computer Science, Carnegie Mellon University, Sep. 2009.

[24] J. H. Christensen, "Using RESTful Web-Services and Cloud Computing to Create Next Generation Mobile Applications," Proc. the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications (OOPSLA 2009), Oct. 25-29 2009, pp. 627-634.

[25] Eucalyptus: Open Source AWS Compatible Private Clouds, [online], Jan. 2014, Available from: https://www.eucalyptus.com/

[26] Nimbus, [online], Jan. 2014, Available from: www.nimbusproject.org/

On Application Responsiveness and Storage Latency in Virtualized Environments

Alexander Spyridakis, Daniel Raho
Virtual Open Systems
Grenoble - France
Email: {a.spyridakis, s.raho}@virtualopensystems.com

***Abstract*—Preserving responsiveness is an enabling condition for running interactive applications effectively in virtual machines. For this condition to be met, low latency usually needs to be guaranteed to storage-Input/Output operations. In contrast, in this paper we show that in virtualized environments, there is a missing link exactly in the chain of actions performed to guarantee low storage-I/O latency. After describing this problem theoretically, we show its practical consequences through a large set of experiments with real world-applications. For the experiments, we used two Linux production-quality schedulers, both designed to guarantee a low latency, and a publicly available I/O benchmark suite, after extending it to correctly measure throughput and application responsiveness also in a virtualized environment. Finally, as for the experimental testbed, we ran our experiments on the following three devices connected to an ARM embedded system: an ultra-portable rotational disk, a microSDHC (Secure Digital High Capacity) Card and an eMMC (embedded Multimedia card) device. This is an ideal testbed for highlighting latency issues, as it can execute applications with about the same I/O demand as a general-purpose system, but for power-consumption and mobility issues, the storage devices of choice for such a system are the aforementioned ones. Additionally, the lower the speed of a storage device is, the consequences of I/O-latency are more evident.**

Keywords: KVM/ARM, virtualization, responsiveness and soft-real time guarantees, scheduling, embedded systems

I. INTRODUCTION

Virtualization is an increasingly successful solution to achieve both flexibility and efficiency in general-purpose and embedded systems. However, for virtualization to be effective also with interactive applications, the latter must be guaranteed a high, or at least acceptable responsiveness. In other words, it is necessary to guarantee that these applications take a reasonably short time to start, and that the tasks requested by these applications, such as, e.g., opening a file, are completed in a reasonable time.

To guarantee responsiveness to an application, it is necessary to guarantee that both the code of the application and the I/O requests issued by the applications get executed with a low latency. Expectedly, there is interest and active research in preserving a low latency in virtualized environments [1][2][3][4][5], especially in soft and hard real-time contexts. In particular, some virtualization solutions provide more or less sophisticated Quality of Service mechanisms also for storage I/O [4][5]. However, even just a thorough investigation on application responsiveness, as related to storage-I/O latency, seems to be missing. In this paper, we address this issue by providing the following contributions.

A. Contributions of this paper

First, we show, through a concrete example, that in a virtualized environment there is apparently a missing link in the chain of actions performed to guarantee a sufficiently low I/O latency when an application is to be loaded, or, in general, when any interactive task is to be performed. To this purpose, we use as a reference two effective schedulers in guaranteeing a high responsiveness: Budget Fair Queuing [7] and Completely Fair Queuing [9]. They are two production-quality storage-I/O schedulers for Linux.

Then, we report experimental results with real-world applications. These results confirm that, if some applications are competing for the storage device in a host, then the applications running in a virtual machine executed in the same host may become from not much responsive to completely unresponsive. To carry out these experiments, we extended a publicly available I/O benchmark suite for Linux [8], to let it comply also with a virtualized environment.

As an experimental testbed, we opted for an ARM embedded system, based on the following considerations. On one hand, modern embedded systems and consumer-electronics devices can execute applications with about the same I/O demand as general-purpose systems. On the other hand, for mobility and energy-consumption issues, the preferred storage devices in the former systems are (ultra) portable and low-power ones. These devices are necessarily slower than their typical counterparts for general-purpose systems. Being the amount of I/O the same, the lower the speed of a storage device is, the more I/O-latency issues are amplified. Finally, as a virtualization solution we used the pair QEMU/KVM, one of the most popular and efficient solutions in ARM embedded systems.

B. Organization of this paper

In Section II, we describe the schedulers that we use as a reference in this paper. Then, in Section III we show the important I/O-latency problem on which this paper is focused. After that, in Section IV, we describe how we modified the benchmark suite to execute our experiments. Finally, we report our experimental results in Section V.

II. REFERENCE SCHEDULERS

To show the application-responsiveness problem that is the focus of this paper, we use the following two storage-I/O schedulers as a reference: BFQ [6] and CFQ [9]. We opted for these two schedulers because, they, both guarantee a high throughput and low latency. In particular, BFQ achieves even up to 30% higher throughput than CFQ on hard disks with parallel workloads. Strictly speaking, only the second feature is related to the focus of this paper, but the first feature is however important, because a scheduler achieving only a small fraction of the maximum possible throughput may be, in general, of little practical interest, even if it guarantees a high responsiveness. The second reason why we opted for these schedulers is that up-to-date and production-quality Linux implementations are available for both. In particular, CFQ is the default Linux I/O scheduler, whereas BFQ is being maintained separately [8]. In addition to the extended tests for BFQ and CFQ, we also identified similar behaviour with the Noop and Deadline schedulers. In the next two sections, we briefly describe the main differences between the two schedulers, focusing especially on I/O latency and responsiveness. For brevity, when not otherwise specified, in the rest of this paper we use the generic term *disk* to refer to both a hard disk and a solid-state disk.

A. BFQ

BFQ achieves a high responsiveness basically by providing a high fraction of the disk throughput to an application that is being loaded, or whose tasks must be executed quickly. In this respect, BFQ benefits from the strong fairness guarantees it provides: BFQ distributes the disk throughput (and not just the disk time) as desired to disk-bound applications, with any workload, *independently of* the disk parameters and even if the disk throughput fluctuates. Thanks to this strong fairness property, BFQ does succeed in providing an application requiring a high responsiveness with the needed fraction of the disk throughput in any condition. The ultimate consequence of this fact is that, regardless of the disk background workload, BFQ guarantees to applications about the same responsiveness as if the disk was idle [6].

B. CFQ

CFQ grants disk access to each application for a fixed *time slice*, and schedules slices in a round-robin fashion. Unfortunately, as shown by Valente and Andreolini [6], this service scheme may suffer from both unfairness in throughput distribution and high worst-case delay in request completion time with respect to an ideal, perfectly fair system. In particular, because of these issues and of how the low-latency heuristics work in CFQ, the latter happens to guarantee a worse responsiveness than BFQ [6]. This fact is highlighted also by the results reported in this paper.

III. Missing link for preserving responsiveness

We highlight the problem through a simple example. Consider a system running a guest operating system, say guest G, in a virtual machine, and suppose that either BFQ or CFQ is the default I/O scheduler both in the host and in guest G. Suppose now that a new application, say application A, is being started (loaded) in guest G while other applications are already performing I/O without interruption in the same guest. In these conditions, the cumulative I/O request pattern of guest G, as seen from the host side, may exhibit no special property that allows the BFQ or CFQ scheduler in the host to realize that an application is being loaded in the guest.

Hence, the scheduler in the host may have no reason for privileging the I/O requests coming from guest G. In the end, if also other guests or applications of any other kind are performing I/O in the host—and for the same storage device as guest G—then guest G may receive *no help* to get a high-enough fraction of the disk throughput to start application A quickly. As a conclusion, the start-up time of the application may be high. This is exactly the scenario that we investigate in our experiments. Finally, it is also important to note that our focus has been in local disk/storage, as scheduling of network-based storage systems is not always under the direct control of the Linux scheduling policies.

IV. Extension of the benchmark suite

To implement our experiments we used a publicly available benchmark suite [8] for the Linux operating system. This suite is designed to measure the performance of a disk scheduler with real-world applications. Among the figures of merit measured by the suite, the following two performance indexes are of interest for our experiments:

Aggregate disk throughput. To be of practical interest, a scheduler must guarantee, whenever possible, a high (aggregate) disk throughput. The suite contains a benchmark that allows the disk throughput to be measured while executing workloads made of the reading and/or the writing of multiple files at the same time.

Responsiveness. Another benchmark of the suite measures the *start-up* time of an application—i.e., how long it takes from when an application is launched to when the application is ready for input—with cold caches and in presence of additional heavy workloads. This time is, in general, a measure of the responsiveness that can be guaranteed to applications in the worst conditions.

Being this benchmark suite designed only for non-virtualized environments, we enabled the above two benchmarks to work correctly also inside a virtual machine, by providing them with the following extensions:

Choice of the disk scheduler in the host. Not only the active disk scheduler in a guest operating system, hereafter abbreviated as just guest OS, is relevant for the I/O performance in the guest itself, but, of course, also the active disk scheduler in the host OS. We extended the benchmarks so as to choose also the latter scheduler.

Host-cache flushing. As a further subtlety, even if the disk cache of the guest OS is empty, the throughput may be however extremely high, and latencies may be extremely low, in the guest OS, if the zone of the guest virtual disk interested by the I/O corresponds to a zone of the host disk already cached in the host OS. To address this issue, and avoid deceptive measurements, we extended both benchmarks to flush caches at the beginning of their execution and, for the responsiveness benchmark, also (just before) each time the application at hand is started. In fact the application is started for a configurable number of times, see Section V.

Workload start and stop in the host. Of course, responsiveness results now depend also on the workload in execution in the host. Actually, the scenario where the responsiveness in a Virtual Machine (VM) is to be

TABLE I. Storage devices used in the experiments

Type	Name	Size	Read peak rate
1.8-inch Hard Disk	Toshiba MK6006GAH	60 GB	10.0 MB/s
microSDHC Card	Transcend SDHC Class 6	8 GB	16 MB/s
eMMC	SanDisk SEM16G	16 GB	70 MB/s

carefully evaluated, is exactly the one where the host disk is serving not only the I/O requests arriving from the VM, but also other requests (in fact this is the case that differs most from executing an OS in a non-virtualized environment). We extended the benchmarks to start the desired number of file reads and/or writes also in the host OS. Of course, the benchmarks also automatically shut down the host workload when they finish.

The resulting extended version of the benchmark suite is available here [10]. This new version of the suite also contains the general scripts that we used for executing the experiments reported in this paper (all these experiments can then be repeated easily).

V. Experimental results

We executed our experiments on a Samsung Chromebook, equipped with an ARMv7-A Cortex-A15 (dual-core, 1.7 GHz), 2 GB of RAM and the devices reported in Table I. There was only one VM in execution, hereafter denoted as just *the* VM, emulated using QEMU/KVM. Both the host and the guest OSes were Linux 3.12.

A. Scenarios and measured quantities

We measured, first, the aggregate throughput in the VM while one of the following combinations of workloads was being served.

In the guest. One of the following six workloads, where the tag **type** can be either **seq** or **rand**, with **seq**/**rand** meaning that files are read or written sequentially/at random positions:

1r-type one reader (i.e., one file being read);
5r-type five parallel readers;
2r2w-type two parallel readers, plus two parallel writers.

In the host. One of the following three workloads (in addition to that generated, in the host, by the VM):

no-host_workload no additional workload in the host;
1r-on_host one sequential file reader in the host;
5r-on_host five sequential parallel readers in the host.

We considered only sequential readers as additional workload in the host, because it was enough to cause the important responsiveness problems shown in our results. In addition, for each workload combination, we repeated the experiments with each of the four possible combinations of active schedulers, choosing between BFQ and CFQ, in the host and in the guest.

The main purpose of the throughput experiments was to verify that in a virtualized environment both schedulers achieved a high-enough throughput to be of practical interest. Both schedulers did achieve, in the guest, about the same (good) performance as in the host. For space limitations, we do not report these results, and focus instead on the main quantity of interest for this paper. In this regard, we measured the start-up time of three popular interactive applications of different sizes, inside the VM and while one of the above combinations of workloads was being served.

The applications were, in increasing-size order: *bash*, the Bourne Again shell, *xterm*, the standard terminal emulator for the X Window System, and *konsole*, the terminal emulator for the K Desktop Environment. As shown by Valente and Andreolini [6], these applications allow their start-up time to be easily computed. In particular, to get worst-case start-up times, we dropped caches both in the guest and in the host before each invocation (Section IV). Finally, just before each invocation a timer was started: if more than 60 seconds elapsed before the application start-up was completed, then the experiment was aborted (as 60 seconds is evidently an unbearable waiting time for an interactive application).

We found that the problem that we want to show, i.e., that responsiveness guarantees are violated in a VM, occurs regardless of which scheduler is used in the host. Besides, in presence of file writers, results are dominated by fluctuations and anomalies caused by the Linux write-back mechanism. These anomalies are almost completely out of the control of the disk schedulers, and not related with the problem that we want to highlight. In the end, we report our detailed results **only with file readers**, **only with BFQ** as the active disk scheduler **in the host**, and **for *xterm***.

B. Statistics details

For each workload combination, we started the application at hand five times, and computed the following statistics over the measured start-up times: minimum, maximum, average, standard deviation and 95% confidence interval (actually we measured also several other interesting quantities, but in this paper we focus only on application responsiveness). We denote as a *single run* any of these sequences of five invocations. We repeated each single run ten times, and computed the same five statistics as above also across the average start-up times computed for each repetition. We did not find any relevant outlier, hence, for brevity and ease of presentation, in the next plots we show only averages across runs (i.e., averages of the averages computed in each run).

C. Results

Figure 1 shows our results with the hard disk (Table I). The reference line represents the time needed to start *xterm* if the disk is idle, i.e., the minimum possible time that it takes to start *xterm* (a little less than 2 seconds). Comparing this value with the start-up time guaranteed by BFQ with no host workload, and with any of the first three workloads in the guest (first bar for any of the *1r-seq*, *5r-seq* and *1r-rand* guest workloads), we see that, with all these workloads, BFQ guarantees about the same responsiveness as if the disk was idle. The start-up time guaranteed by BFQ is slightly higher with *5r-rand*, for issues related, mainly, to the slightly coarse time granularity guaranteed to scheduled events in the kernel in an ARM embedded system, and to the fact that the reference time itself may advance haltingly in a QEMU VM.

In contrast, again with no host workload, the start-up time guaranteed by CFQ with *1r-seq* or *1r-rand* on the guest is 3

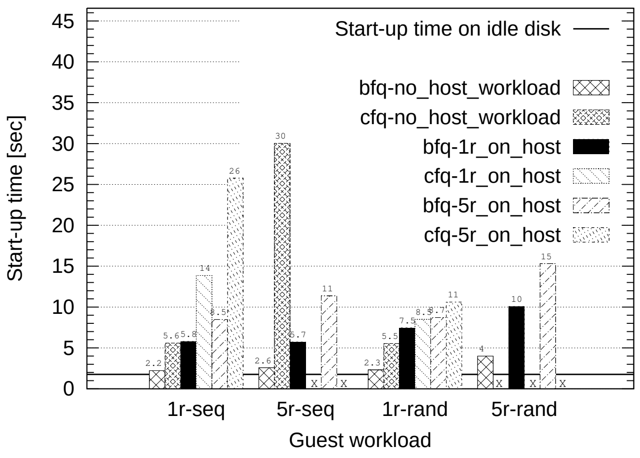


Figure 1. Results with the hard disk (lower is better).

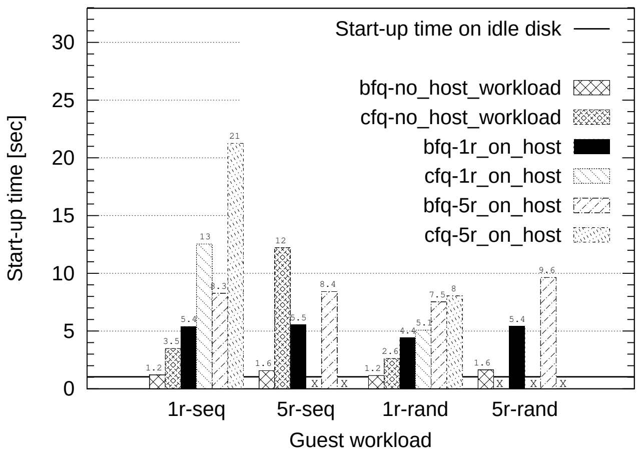


Figure 2. Results with the microSDHC CARD (lower is better).

times as high than on an idle disk, whereas with *5r-seq* the start-up time becomes about 17 times as high. With *5r-seq* the figure reports instead an **X** for the start-up time of CFQ: we use this symbolism to indicate that the experiment failed, i.e., that the application did not succeed at all in starting before the 60-second timeout.

In view of the problem highlighted in Section III, the critical scenarios are however the ones with some additional workload in the host; in particular, *1r_on_host* and *5r_on_host* in our experiments. In these scenarios, both schedulers unavoidably fail to preserve a low start-up time. Even with just *1r_on_host*, the start-up time, with BFQ, ranges from 3 to 5.5 times as high than on an idle disk. The start-up time with CFQ is much higher than with BFQ with *1r_on_host* and *1r-seq* on the guest, and, still with *1r_on_host* (and CFQ), is even higher than 60 seconds with *5r-seq* or *5r-rand* on the guest. With *5r_on_host* the start-up time is instead basically unbearable, or even higher than 60 seconds, with both schedulers. Finally, with *1r-rand* all start-up times are lower and more even than with the other guest workloads, because both schedulers do not privilege much random readers, and the background workload is generated by only one reader.

Figures 2 and 3 show our results with the two flash-based devices. At different scales, the patterns are still about the same as with the hard disk. The most notable differences are related to CFQ: on one side, with no additional host workload, CFQ achieves a slightly better performance than on the hard disk, whereas, on the opposite side, CFQ suffers from a much higher degradation of the performance, again with respect to the hard-disk case, in presence of additional host workloads.

To sum up, our results confirm that, with any of the devices considered, responsiveness guarantees are lost when there is some additional I/O workload in the host.

VI. CONCLUSION AND FUTURE WORK

In this paper, we showed both theoretically and experimentally that responsiveness guarantees, as related to storage I/O, may be violated in virtualized environments. Even with schedulers, which target to achieve low latency through heuristics, the problem of low responsiveness still persists

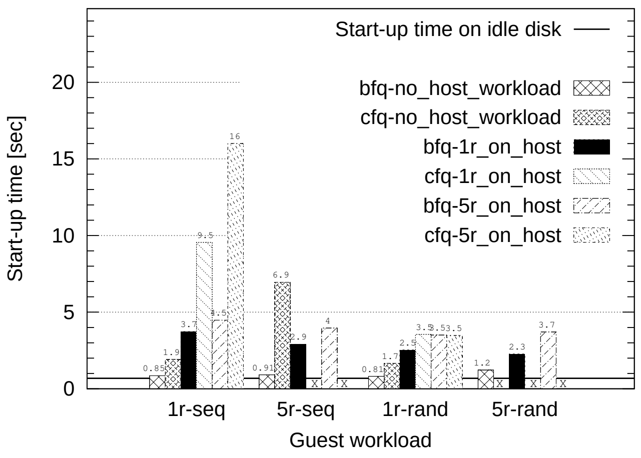


Figure 3. Results with the eMMC (lower is better).

in virtual machines. The host receives a mix of interactive and background workloads from the guest, which can completely contradict per process heuristics by schedulers such as BFQ. We are currently devising a solution for preserving responsiveness also in virtualized environments. The target of this approach is specifically for embedded systems and the KVM on ARM hypervisor, which introduces the concept of coordinated scheduling between the host/guest scheduler and KVM itself. Besides, we also plan to extend our investigation to latency guarantees for soft real-time applications (such as audio and video players), and to consider more complex scenarios, such as more than one VM competing for the storage device.

ACKNOWLEDGMENTS

The authors would like to thank Paolo Valente for providing details about BFQ and his support for the benchmark suite. We also thank the anonymous reviewers for their precious feedback and comments on the manuscript.

References

[1] Jaewoo Lee, et. al., “Realizing Compositional Scheduling through Virtualization”, IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS’12), April 2012, pp. 237-246.

[2] K. Sandstrom, A. Vulgarakis , M. Lindgren, and T. Nolte, “Virtualization technologies in embedded real-time systems”, Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on, Sept. 2013, pp. 1-8.

[3] Z. Gu and Q. Zhao, “A State-of-the-Art Survey on Real-Time Issues in Embedded Systems Virtualization”, Journal of Software Engineering and Applications, Vol. 5 No. 4, 2012, pp. 277-290.

[4] Storage I/O Control Technical Overview [retrieved: April, 2014]. http://www.vmware.com/files/pdf/techpaper/VMW-vSphere41-SIOC.pdf

[5] Virtual disk QoS settings in XenEnterprise [retrieved: April, 2014]. http://docs.vmd.citrix.com/XenServer/4.0.1/reference/ch04s02.html

[6] P. Valente and M. Andreolini, “Improving Application Responsiveness with the BFQ Disk I/O Scheduler”, Proceedings of the 5th Annual International Systems and Storage Conference (SYSTOR ’12), June 2012, p. 6

[7] F. Checconi and P. Valente, “High Throughput Disk Scheduling with Deterministic Guarantees on Bandwidth Distribution”, IEEE Transactions on Computers, vol. 59, no. 9, May 2010.

[8] BFQ homepage [retrieved: April, 2014]. http://algogroup.unimore.it/people/paolo/disk_sched

[9] CFQ I/O Scheduler [retrieved: April, 2014]. http://lca2007.linux.org.au/talk/123.html

[10] Extended version of the benchmark suite for virtualized environments [retrieved: April, 2014]. http://www.virtualopensystems.com/media/bfq/benchmark-suite-vm-ext.tgz

Applying a Resource-pooling Mechanism to MPLS-TP Networks to Achieve Service Agility

Tomoyuki Iijima, Toshiaki Suzuki, and Kenichi Sakamoto
Central Research Laboratory
Hitachi, Ltd.
Kanagawa, Japan
{tomoyuki.iijima.fg, toshiaki.suzuki.cs, kenichi.sakamoto.xj}@hitachi.com

Hidenori Inouchi and Akihiko Takase
Telecommunications & Network Systems Division
Hitachi, Ltd.
Kanagawa, Japan
{hidenori.inouchi.dw, akihiko.takase.wa}@hitachi.com

***Abstract*— A concept called “software-defined networking” (SDN) is applied to carrier networks as one way to add flexibility to those networks. To apply SDN to carrier networks, a resource-pooling mechanism in MPLS-TP networks is proposed. The feasibility of the proposed resource-pooling mechanism applied to Multi-Protocol Label Switching – Transport Profile (MPLS-TP) networks was evaluated in terms of service agility. Moreover, a controller, which utilizes this mechanism to allocate a pooled resource to IP traffic, is prototyped and evaluated. The time required for the controller to allocate pooled resources in MPLS-TP networks to IP traffic is sufficiently short. This result indicates that the proposed mechanism will help to flexibly change carrier networks and reduce manual configurations spanning multiple layers. Consequently, the proposed mechanism will help assure service agility.**

Keywords-Cloud computing; SDN; MPLS-TP; service agility;

I. Introduction

As cloud computing continues to grow, the number of cloud services is increasing dramatically [1]. Cloud computing is a technology that enables users to access “a large pool of data and computational resources” located far afield via the Internet [2]. These resources are mostly deployed in data centers and are provided to users by making full use of “virtualization.” Through these resources, “dynamically composable services” can be deployed through “Web service interfaces [3].” Users can easily and flexibly start their own services using these resources.

For example, online-game providers can use a large amount of computational resources during the launch of their service in order to attract a large number of users. They can thus reduce the amount of investment that they would otherwise have needed if they prepared the resources themselves.

People are now using these resources as if they were located on local computers. A cloud-computing environment is largely supported by the rapidly increasing bandwidth available on the Internet. Even so, it is hard to say that network resources are virtualized enough. Although traffic patterns produced by cloud computing are volatile because of sporadic increases and decreases in the number of Virtual Machines (VMs), networks are not necessarily changed flexibly enough to keep up with such volatility.

In light of this background, Software-Defined Networking (SDN) is getting wide attention. SDN is a concept that separates the control plane of network devices from their data plane and puts the control plane in one place. The controller put in that place then controls the entire network [4]. This centralized controller is expected to have the capability to control network resources virtually. It is also expected to provide flexibility and reliability on behalf of network devices by making full use of virtualization.

These technological trends in networks are often mentioned within the context of data-center networks. However, they are not limited to data-center networks. Carrier networks, i.e., the backbone-network infrastructure managed by telecommunication-service providers, are also affected by the volatile nature of traffic resulting from the increasing number of services provided on the cloud-computing platform. In this regard, it is necessary that carrier networks support SDN; in other words, carrier networks must be virtualized and flexible [5].

There are, however, several issues concerning current carrier networks, and these issues are described in Section II. To address these issues, as described in Section III, “Multi-Protocol Label Switching – Transport Profile” (MPLS-TP) networks have been proposed as a field where SDN is introduced in carrier networks. A resource-pooling mechanism in MPLS-TP networks and collaboration between MPLS-TP and IP networks have also been proposed. The application of the proposed mechanism is described in Section IV, and the proposed mechanism is evaluated in Section V. Finally, future work concerning the mechanism is mentioned in Section VI.

II. Issues concerning Carrier Networks

As mentioned in Section I, network traffic produced by cloud computing is putting a heavy burden on carrier networks. The volatile nature of traffic generated by VMs and various kinds of services, such as video streaming and gaming on-line, that are provided on a cloud-computing platform are also negatively affecting carrier networks. According to Manzalini et al. [6], for example, moving a VM across a Wide-Area Network (WAN) requires at least 622 Mbps of bandwidth throughout the WAN. On top of that requirement, due to accelerating globalization, carrier networks are expected to deal with more-frequent requests to make network changes spanning long distances. In this regard, introducing SDN into carrier networks will help to satisfy these expectations.

However, current carrier networks have at least two issues concerning introducing SDN. As for the first issue, carrier networks must be tightly controlled to provide high reliability. As for the second issue, they must be multi-layered and allow communications between operators of different layers. These issues are described in detail as follows.

A. *Rigidness of carrier networks*

In contrast to the “routed-packet network [7],” which is highly distributed, and therefore, not necessarily expected to provide high reliability, a carrier network is expected to ensure high reliability. For example, transport technologies used by carrier networks, such as Synchronous Optical NETwork/Synchronous Digital Hierarchy (SONET/SDH) [8], are equipped with high-reliability functions such as guaranteeing bandwidth, path protection within 50 milliseconds, and “Operation, Administration, and Maintenance” (OAM). Provisioning in the transport layer is, therefore, rigid and requires significant human intervention [9].

B. *Multi-layers in carrier networks*

Current carrier networks are mostly multi-layered, and their core is made from optical transport networks, consisting of Wavelength-Division Multiplexing (WDM) [10] devices and SONET/SDH devices. Routed-packet networks, namely, IP networks consisting of routers and switches, surround the core of the carrier networks. As of now, each type of device is managed by a different operator. When carriers want to change their networks, operators of different layers thus have to communicate with each other, resulting in a long lead time. Setting up or modifying carrier networks, which involves a man-to-man interface, lasts days or even weeks [11].

These issues are making it difficult for carrier networks to meet current demand and to achieve service agility. It is thus acutely necessary to realize flexibility without undermining the current high reliability of carrier networks.

III. Proposal of Resource Pooling

To address the issues described in the previous section, a packet-transport technology called “MPLS-TP” is applied as one field in which SDN can be introduced. To realize service agility, a resource-pooling mechanism on MPLS-TP networks and collaboration between MPLS-TP and IP networks are proposed. This approach is similar to CloudNet [12] in that a controller assigns a pooled network resource to a user, but it is different in that the pooled resource is in the transport layer of a carrier network.

A. *MPLS-TP as a packet-transport technology*

MPLS-TP is a protocol being standardized by the Internet Engineering Task Force (IETF). Originally, the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) was working on the preceding Transport – Multi-Protocol Label Switching (T-MPLS) protocol in order to emulate the SONET/SDH protocols by developing a whole new range of carrier-class service attributes [13]. However, the IETF took over the T-MPLS standardization as MPLS-TP in order to provide compatibility with IP/MPLS. Accordingly, it has been reported that the stage is set for replacement of SONET/SDH in packet networks [13].

The architecture of MPLS-TP is shown in Figure 1. The controller manages the entire MPLS-TP network [14], in a sense that the control plane and data plane are separated, MPLS-TP networks are managed in the same manner as SDN. The controller “simply sets up one tunnel,” namely, a Label-Switched Path (LSP) and a pseudo wire, “from source to destination [15].” These tunnels, “due to their deterministic nature of bandwidth and delay, provide a carrier-class solution for transport of any payload [15].”

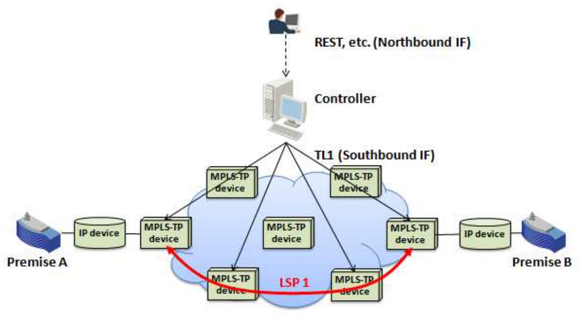


Figure 1. Architecture of MPLS-TP networks.

The functions of MPLS-TP for providing high reliability are well developed. For example, once configured on a MPLS-TP device, a working LSP is required to switch to a standby LSP within 50 milliseconds when it is damaged. The continuity over the LSP is checked before the first packet is sent from the source to destination.

Although one of the advantages of SDN is that the controller ensures reliability on behalf of network devices by making full use of virtualization, it is suggested here that reliability should be ensured by MPLS-TP devices. On top of that suggestion, it is also suggested that flexibility in carrier networks should be attained by introducing a resource-pooling mechanism on MPLS-TP networks.

B. *Resource-pooling mechanism in MPLS-TP networks*

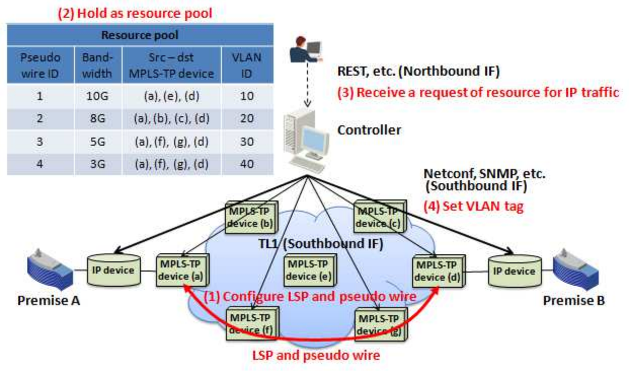
As a method to flexibly change MPLS-TP networks in a short lead time in accordance with certain traffic patterns or user demands, using a LSP and a pseudo wire as a resource pool is proposed in this section.

According to Wischik et al. [16], "resource pooling" is defined as a technology that "helps robustness against failure, load balancing, and flexibility in the face of bursts of traffic while avoiding problems and limitations." Moreover, according to Gmach et al. [17], "many enterprises are beginning to exploit a shared-resource-pool environment to lower their infrastructure and management costs." Under these circumstances, the proposed resource-pooling mechanism on a carrier network therefore targets achieving flexibility in the face of bursts of traffic.

To create a resource pool composed of LSPs and pseudo wires, first, the resources should be set in advance. The status of these resources, such as "used" and "unused," should be held by the controller. When in need of "flexibility in the face of bursts of traffic," the controller activates the pooled and unused LSPs and pseudo wires. It then allocates the resource to a certain traffic pattern or a user demand.

C. *Resource allocation through collaboration*

When a pooled resource of LSPs and pseudo wires is used to transport traffic sent from "routed packet networks," this traffic sent over the pooled resource is IP traffic. In this case, the controller needs to know the attributes of the IP traffic. Accordingly, the controller must manage not only MPLS-TP networks but also IP networks. As shown in Figure 2, the controller then allocates pooled resources in MPLS-TP networks to IP networks.



Resource pool			
Pseudo wire ID	Band-width	Src – dst MPLS-TP device	VLAN ID
1	10G	(a), (e), (d)	10
2	8G	(a), (b), (c), (d)	20
3	5G	(a), (f), (g), (d)	30
4	3G	(a), (f), (g), (d)	40

Figure 2. Collaboration between MPLS-TP and IP networks.

There are several possible scenarios about how to use a resource pool on an MPLS-TP network. One of these scenarios is shown in Figure 2. First, the controller configures LSPs and pseudo wires in advance, such as ones that starts from MPLS-TP device (a) through (e) to (d), and holds these resources as a pooled resource. The controller then associates each pseudo wire with a virtual-LAN identifier (VLAN ID) in order to transport IP traffic over the pseudo wire [18]. For example, in the figure, pseudo wire with ID "1" is associated with VLAN ID "10." Therefore, IP traffic with VLAN ID tag "10" in its layer-2 header, which is sent to MPLS-TP device (a), will be transported over the pseudo wire with ID "1." This traffic will thus be transported from MPLS-TP device (a) through (e) to (d).

When a user of IP traffic demands that its traffic is sent between premises A and B with guaranteed bandwidth of 10 Gbps within a certain period of time, the controller determines that the pseudo wire with ID "1" meets this demand. It then allocates that pseudo wire to that IP traffic. For example, when a user demands that its IP traffic from address "100.100.100.0/24" of premise A to address "100.100.200.0/24" of premise B is sent over carrier networks with guaranteed bandwidth of 10 Gbps within a certain period of time, under the assumption that the routing configuration has already been made on the edge IP devices, the controller assigns VLAN ID tag "10" to the IP traffic on the edge IP devices.

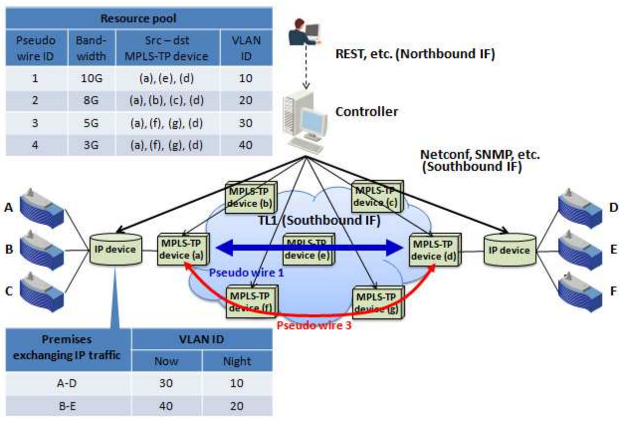
A different collaboration scenario is also possible. The controller can retrieve information about IP networks and change the attributes of pooled resources in MPLS-TP networks accordingly. For example, when a user demands that its IP traffic from address "100.100.100.0/24" of premise A to address "100.100.200.0/24" of premise B (with already assigned VLAN ID "50") is sent over carrier networks with guaranteed bandwidth of 10 Gbps, the controller changes the assignment of the pseudo wire from VLAN ID "10" to "50."

The first scenario mentioned above is focused in this study as a way to achieve service agility under the assumption that configuring an IP device is less time-consuming than configuring an MPLS-TP device.

IV. USE CASE AND EFFECTS

A. *Use case*

A use case in which a resource is shared by multiple users in a short interval, namely, data backup, is depicted in Figure 3.



Resource pool			
Pseudo wire ID	Band-width	Src – dst MPLS-TP device	VLAN ID
1	10G	(a), (e), (d)	10
2	8G	(a), (b), (c), (d)	20
3	5G	(a), (f), (g), (d)	30
4	3G	(a), (f), (g), (d)	40

Premises exchanging IP traffic	VLAN ID	
	Now	Night
A-D	30	10
B-E	40	20

Figure 3. Use case of resource pooling in MPLS-TP networks.

As shown in the figure, IP traffic between premises A and D is already assigned VLAN ID "30," meaning that the IP traffic is transported over the pseudo wire with ID "3" with guaranteed bandwidth of 5 Gbps. In contrast, IP traffic between premises B and E is already assigned VLAN ID "40," meaning that the IP traffic is transported over pseudo wire with ID "4" with guaranteed bandwidth of 3 Gbps.

When a user demands that its IP traffic between premises A and D to be sent over carrier networks with guaranteed bandwidth of 10 Gbps for data backup only at night, the controller acknowledges the demand and assigns VLAN ID tag “10” to this IP traffic at night. Under this configuration, IP traffic between premises A and D is switched from pseudo wire with ID “3,” which has a narrower bandwidth, to pseudo wire with ID “1,” which has a larger bandwidth.

This use case concerns a resource being allocated to a user as a data-backup network in a short lead time. This method achieves flexibility in the face of bursts of traffic.

B. *Expected effects*

The use case described in the previous section is effective in an environment where users do not always use the full potential of guaranteed bandwidth, which is allocated according to the contract between a user and a carrier.

As shown in Figure 4, by enabling the controller to handle a resource pool on MPLS-TP networks and to allocate this resource to IP traffic, resources on MPLS-TP networks will be effectively used by multiple users in a short interval.

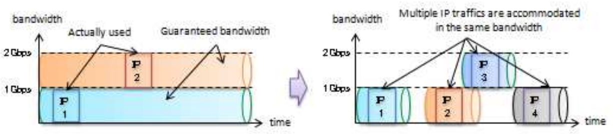


Figure 4. Effects of resource pooling in MPLS-TP networks.

In the left graph, a certain bandwidth is exclusively allocated to one user according to the contract between the user and carrier, and that bandwidth is used by only one type of IP traffic. In this case, bandwidth is underutilized. In contrast, in the right graph, bandwidth is utilized more effectively since it is shared as a resource pool by multiple types of IP traffic in a short interval.

Today, a large amount of lead time is required to change carrier networks. However, thanks to the resource-pooling mechanism and the controller that manages it for IP networks, carrier networks can be flexibly changed in accordance with frequent user demands and volatile traffic patterns resulting from cloud computing. The controller will also reduce the amount of operators’ communications between different layers.

V. PROTOTYPE USING RESOURCE POOLING

A prototype controller, named “Multi-Layer Orchestrator” (MLO), which adopts the resource-pooling mechanism and controls MPLS-TP and IP networks in a collaborative manner, is described as follows.

A. *User interface for configuring MLO*

An overview of the controller, implemented as the MLO, is shown in Figure 5. The MLO provides a REpresentational State Transfer (REST) interface as a northbound interface. Through an application that uses this interface, a user can request a transport network with a guaranteed bandwidth. Manipulating the Graphical User Interface (GUI) of the application, the user can specify a source and destination between which its IP traffic is transported. The user can also request a guaranteed bandwidth. For example, the user can demand that IP traffic from address “100.100.100.0/24” to address “100.100.200.0/24” should be transported in the carrier network with a guaranteed bandwidth of 10 Gbps.

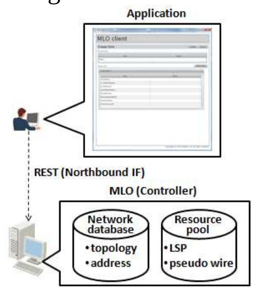


Figure 5. User interface for configuring MLO.

The MLO includes a NetWork DataBase (NWDB) in addition to the resource pool. Topology and address data concerning MPLS-TP and IP networks are held inside this database. When the MLO receives a user demand through the REST interface, it determines which edge IP devices this IP traffic belongs to by referring to the NWDB. It also identifies physical ports of these IP devices through which this IP traffic goes. Then, the MLO identifies MPLS-TP devices that are connected to the physical ports of the edge IP devices. Since the IP traffic will be transported between these MPLS-TP devices, the MLO, finally, searches the resource pool to find the pooled resource that meets the user’s demand.

B. *Data retrieval of MPLS-TP networks through TL1*

For the MLO to hold and manage the resource pool, it needs to obtain the current status of MPLS-TP networks from MPLS-TP devices. Statuses of MPLS-TP networks are retrieved from MPLS-TP devices through Transaction Language 1 (TL1).

If the MLO receives a user demand for guaranteed bandwidth of 10 Gbps for certain IP traffic, it asks the resource pool whether there is a pseudo wire that guarantees a bandwidth of 10 Gbps. If it determines that pseudo wire with ID “1” has the desired bandwidth of 10 Gbps, it searches for the VLAN ID associated with that pseudo wire. If the VLAN ID is “10,” the MLO assigns this VLAN ID to the IP traffic. Then, the MLO configures the IP devices to allocate a VLAN ID to the IP traffic.

C. *Configuration of IP networks through NETCONF*

To assign a VLAN ID to IP traffic, the MLO needs to configure a VLAN ID on IP devices. VLAN ID tags on IP devices are configured through NETCONF [19].

If the MLO concludes that it must assign VLAN ID “10” to a requested IP traffic, it associates VLAN ID tag “10” to the ports of the source and destination IP devices in which the IP traffic is transported. If another VLAN ID is already associated with the IP traffic, the MLO changes the VLAN ID from the previous VLAN ID to “10.”

With this method, the MLO consequently switches the pseudo wire from an old one to a new one over which IP traffic is sent.

VI. EVALUATIONS

To evaluate the feasibility of the MLO in terms of service agility, the configuration time was evaluated as explained below.

A. Evaluation method

The MLO was evaluated in the following environment. Both an application and the MLO were run on a general-purpose computer, whose specification is listed in Table I. The application, through which a user specifies its demand, was implemented with a Java Development Kit (JDK) [20].

The interface, through which the MLO controls MPLS-TP devices, was implemented with JDK. And NETCONF, through which the MLO controls IP devices, was implemented by using the AX – Open Networking – Application Programming Interface (AX-ON-API), which is a Java library [21].

TABLE I. SPECIFICATION OF APPLICATION AND MLO

Specification items	Application	MLO
Operating system	Windows 7 [22]	Ubuntu 12.04 [23]
Processor	2.5 GHz	2.67 GHz
Memory	4 Gbytes	3 Gbytes
Network interface card	1 Gbps	1 Gbps
Runtime environment	Java 7	Java 7
NETCONF implementation	–	AX-ON-API

A testbed composed of an application, the MLO, three MPLS-TP devices, and two IP devices (as shown in Figure 6 with the specification listed in Table II) was constructed. The data plane between MPLS-TP and IP devices was wired by using 10G Ethernet. The control plane between the MLO and all the network devices was wired by using 1G Ethernet.

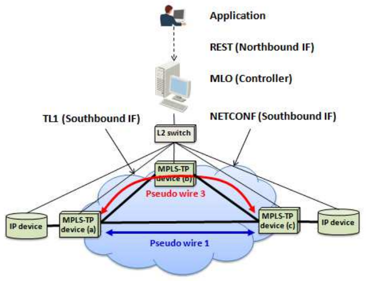


Figure 6. Configuration of testbed.

TABLE II. SPECIFICATION OF MPLS-TP AND IP DEVICES

Type of devices	Product name	Number of devices
MPLS-TP devices	AMN 6400 [24]	3
IP devices	AX 8616 [25]	2

B. Results of evaluation

The time of information retrieval from MPLS-TP devices was evaluated, and the results of the evaluation are listed in Table III. When the MLO retrieved information about a pseudo wire from MPLS-TP devices in order to hold that information as the resource pool, the time needed was 1.5 seconds. In addition, the time to configure the IP devices was evaluated. When the MLO configures the IP devices, the time needed was 13.6 seconds. In total, the time for resource allocation, i.e., the time from the point that the user demanded a network change to the point that a resource of MPLS-TP networks was allocated to the user, was 16.5 seconds.

TABLE III. EVALUATION RESULTS

Evaluation item	Results
Time to retrieve information from MPLS-TP devices	1.5 seconds
Time to configure IP devices	13.6 seconds
MLO's internal processing time	1.4 seconds
Total time to allocate resource to a user	16.5 seconds

On this testbed, the resource pool composed of LSPs and pseudo wires was set in advance. The user then demands that its IP traffic is sent between two MPLS-TP devices with guaranteed bandwidth. After receiving the user demand, the MLO analyzes the demand, searches the resource pool for the demand, and configures IP devices accordingly. Lead time, namely, the time from the point of "user demand," made at "application" in the figure, to the point of "resource allocation," displayed at "application," was evaluated.

The time required for retrieving information from MPLS-TP networks in a carrier network was short. The amount of time is not expected to increase linearly in accordance with the number of MPLS-TP devices since information from each MPLS-TP device is retrieved in parallel. This result indicates that the controller can get timely information from MPLS-TP devices. The resource pool held in the MLO is thus always up-to-date as long as the MLO retrieves information in a short interval.

Currently, it is common to take days or weeks to change the transport technology used in carrier networks. Thus, there is a trend to reduce provisioning time to minutes by placing the service layer on top of the management systems (i.e., controllers) [12]. In this regard, the proposed mechanism aligns with this trend and achieves sufficiently short lead time and service agility.

Consequently, by having a resource-pool mechanism in the MPLS-TP network and making collaboration between MPLS-TP and IP networks, the MPLS-TP network in a carrier network is controlled flexibly. Accordingly, guaranteed bandwidth provided by an LSP and pseudo wire in carrier networks is flexibly allocated to IP traffic according to changes in IP traffic.

VII. CONCLUSIONS AND FUTURE WORK

A growing number of services are provided by cloud computing. The volatile nature of traffic attributed to the behavior of VMs and the increasing number of services provided by cloud computing are, however, negatively affecting carrier networks. To keep up with these trends, carrier networks must be controlled flexibly by SDN.

However, because of a carrier's responsibility to provide high reliability, transport technology in current carrier networks is strictly controlled. Moreover, because carrier networks are multi-layered, communications between operators of different layers are necessary. These issues result in long lead times to change carrier networks.

To control transport technology in carrier networks by SDN, a resource-pooling mechanism on MPLS-TP networks and collaboration between MPLS-TP and IP networks are proposed. More specifically, a controller (named "MLO"), which manages pooled LSPs and pseudo wires and allocates these resources to IP traffic according to user demands, is proposed.

The time to allocate a resource to a user demand was evaluated as 16.5 seconds. From this result, it is concluded that the proposed resource-pooling mechanism on MPLS-TP networks and collaboration between MPLS-TP and IP networks enables SDN in carrier networks in a sense that the proposed mechanism flexibly changes transport technology used in carrier networks. It also reduces communications between operators of different network layers. Consequently, the proposed mechanism can effectively cope with the volatile nature of traffic and thereby achieve service agility. Users utilizing cloud-computing services will thus be able to use network resources in carrier networks on demand.

Future work will be to equip the MLO with a mechanism that makes it work properly with congested networks with few resources, and to equip it with other interfaces of IP devices, such as an "Interface to the Routing System" (I2RS) [26]. By equipping the MLO with these interfaces, it can control not only the VLAN ID tag but also the routing tables of IP devices. This capability will make it possible for the MLO to change routes in layer-3 networks according to resource-pool information of underlying MPLS-TP networks.

ACKNOWLEDGMENT

Part of this research was supported by The Japanese Ministry of Internal Affairs and Communications (MIC) project "Open, Organic, and Optima (O_3)."

REFERENCES

[1] S. Han, M. Hassan, C. Yoon, and E. Huh, "Efficient service recommendation system for cloud computing market," The 2nd International Conference on Interaction Science: Information Technology, Culture and Human, November 2009, pp. 839-845.

[2] D. Nurmi, R. Wolski, and C. Grzegorczyk, "The Eucalyptus Open-source Cloud-computing System," The 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, May 2009, pp. 124-131.

[3] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer System, Vol. 25, Issue 6, June 2009, pp. 599–616.

[4] B. Lants, B. Heller, and N. McKeown, "A Network in a Laptop : Rapid Prototyping for Software-Defined Networks," The 9th ACM SIGCOMM Workshop on Hot Topics in Networks, October 2010, pp. 19:1-19:6.

[5] D. Soldani and R. Saracco, "Future carrier networks," IEEE Communications Magazine, July 2013, pp. 24-26.

[6] A. Manzalini, R. Minerva, F. Callegati, W. Cerroni, and A. Campi, "Clouds of Virtual Machines in Edge Networks," IEEE Communications Magazine, July 2013, pp. 63-70.

[7] C. Janz, "Bringing it all together: Multi-layer software-defined networks and automated operations intelligence," http://www.sdnjapan.org/images/2013spdf/0920_05ciane.pdf, SDN Japan 2013, September 2013 [retrieved: March 2014].

[8] ITU-T, "Characteristics of synchronous digital hierarchy (SDH) equipment functional blocks," http://www.itu.int/rec/T-REC-G.783-200603-I/en, October 2006 [retrieved: May 2014].

[9] L. Velasco, et al., "In-Operation Network Planning," IEEE Communications Magazine, January 2014, pp. 52-60.

[10] A. Salehl, "Optical WDM technology for networking and switching applications," Optical Fiber Communication Conference, February 1992, pp. 199.

[11] R. Giladi and E. Menachi, "ETNA's service layer architecture for automatic provisioning of inter-domain Ethernet trasnport services," GLOBECOM Workshops, November 2009, pp. 1-6.

[12] T. Wood, P. Shenoy, K. Ramakrishnan, and J. Merwe, "CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines,"7th ACM SIGPLAN/SIGOPS international conference on virtual execution environments, Vol. 46, Issue 7, July 2011, pp. 121-132.

[13] R. Vaishampayan, A. Gumaste, S. Rana, and N. Ghani, "Application Driven Comparison of T-MPLS/MPLS-TP and PBB-TE – Driver Choices for Carrier Ethernet," INFOCOM Workshops, April 2009, pp. 1-6.

[14] B. Niven-Jenkins, D. Brungard, M. Betts, N. Sprecher, and S. Ueno, "Requirements of an MPLS Transport Profile," RFC 5654, September 2009.

[15] S. Bryant and P. Pate, "Pseudo wire emulation edge to edge (PWE3) architecture," RFC 3985, March 2005.

[16] D. Wischik, M. Handley, and M. Bagnulo Braun, "The Resource Pooling Principle," ACM SIGCOMM Computer Communication Review, Vol. 38, Issue 5, October 2008, pp. 47–52.

[17] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Resource Pool Management: Reactive versus Proactive or Let's be Friends," Computer Networks, Vol.53, no. 17, December 2009, pp. 2905–2922.

[18] R. Martinotti, et al., "Interworking between MPLS-TP and IP/MPLS," http://tools.ietf.org/html/draft-martinotti-mpls-tp-interworking-02, June 2011 [retrieved: March 2014].

[19] IETF, "Network Configuration (netconf)," http://datatracker.ietf.org/wg/netconf/charter/ [retrieved: March 2014].

[20] Oracle Corp., http://www.oracle.com/technetwork/java/copyright-136087.html [retrieved: March 2014].

[21] T. Iijima, H. Kimura, M. Kitani, and Y. Atarashi, "Development of NETCONF-based Network Management Systems in Web Services Framework," IEICE Trans on Communications, Vol. E92-B, no. 4, April 2009, pp. 1104–1111.

[22] Microsoft Corp., http://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/Usage/Windows.aspx [retrieved: March 2014].

[23] Canonical Ltd., http://www.canonical.com/intellectual-property-rights-policy [retrieved: March 2014].

[24] Hitachi, http://www.hitel.com/solutions/products/optical_transport/6400.html [retrieved: April 2014].

[25] Alaxala, "AX8600R : ALAXALA Networks Corporation," http://www.alaxala.com/en/products/AX8600R/index.html [retrieved: March 2014].

[26] IETF, "Interface to the Routing System (i2rs)," http://datatracker.ietf.org/wg/i2rs/charter/ [retrieved: March 2014].

Analysis of Protection Options for Virtualized Infrastructures in Infrastructure as a Service Cloud

Ramaswamy Chandramouli
Computer Security Division, Information Technology Laboratory
National Institute of Standards & Technology
Gaithersburg, MD, USA
e-mail: mouli@nist.gov

***Abstract*-Infrastructure as a Service (IaaS) is one of the three main cloud service types where the cloud consumer consumes a great variety of resources, such as computing (Virtual Machines or VMs), virtual network, storage and utility programs (DBMS). Any large-scale offering of this service is feasible only through a virtualized infrastructure at the service provider. At the minimum, this infrastructure is made up of resources such as Virtualized hosts together with associated virtual network and hardware/software for data storage An IaaS's consumer's total set of interactions with these resources constitutes the set of use cases for IaaS cloud service. These use cases have associated security requirements and these requirements are met by protection options enabled by available security solutions/technologies. The contribution of this paper is to analyze these protection options. The outcome of this analysis is a realistic characterization of the features, security strengths & architectural foundations of these protection options that will enable an IaaS consumer to choose the appropriate set of security solutions depending upon its deployment context.**

Keywords- Virtualization; Cloud Infrastructure; Virtual Machine; Virtual Network; Infrastructure as a Service

I. INTRODUCTION

Infrastructure as a Service (IaaS) is one of the three main cloud service models that makes available to consumers resources, such as computing (Virtual Machines or VMs), virtual network, storage and utility programs (DBMS). This results in a large virtualized infrastructure at the IaaS cloud service provider's data centers. At the minimum, this infrastructure has to consist of Virtualized hosts together with associated virtual network and hardware/software for data storage. The IaaS consumer's different interactions with these resources constitute the typical set of use cases for the IaaS cloud service. In order that these interactions are secure, certain security requirements should go with each use case.

With increasing demand for IaaS cloud service and competitive nature of the market place, cloud providers and third parties are offering many security solutions. Depending upon the functionality and architecture of these security solutions, they can either be deployed by IaaS cloud consumer (e.g., VM-based Anti-Virus software) or only by IaaS cloud provider (e.g., Hypervisor-based firewall). Also, given the varied feature set and the deployment architecture of these security solutions (let us call them as "protection options" in the rest of this paper), we need an objective way of correlating their security functionality (features) with the security requirements of use cases and determine as to how effectively they (protection options or security solutions) address those security requirements In other words, we need a methodology for performing analysis of the available protection options in the context of the security requirements stemming from a typical set of IaaS cloud service use cases. This is the objective of this paper.

The first step of the methodology, therefore, is to identify the typical set of uses cases encountered by IaaS cloud consumer. Those that we consider in this paper are: (a) Checking out VM Images, (b) Configuring VM instance OS (Guest OS), (c) Configuring Virus/Malware protection for VM instance, (d) Configuring VM instance access protection, (e) Configuring VM instance lifecycle operation protection, (f) Configuring VM instance isolation, and (g) Comprehensive Data protection. The justification for choosing these as use cases comes from the fact that in any IaaS SLA (Service Level Agreement), the responsibility for VM protection & its associated data lies entirely with the consumer. For each use case, we consider the security requirements and then analyze the features/capabilities of the available protection options (provided either by IaaS cloud provider as an integral part of the service or Commercial off-the-shelf (COTS) solutions deployable by IaaS cloud consumer) to meet those requirements. The rest of the paper is organized as follows: Each of the Sections II through VIII provides a brief description of the use case, the security requirements for each use case and an analysis of protection options that are enabled by available security solutions/technologies. The Conclusions and Benefits section summarizes the protection options and findings resulting from analysis for each use case and also outlines the benefits of our approach.

II. CHECKING OUT VM IMAGES

VM images launched on a virtualized host become running VM instances. Each VM image is a self-contained package that contains all constituents needed for running a complete computing stack such as: (a) OS binaries together with other files in the OS distribution, as well as patches, and (b) files containing description of all virtual resources that make up a VM – starting from processor cores, memory size, virtual disks or data stores, etc. Thus, we see that VM images are a set of files which are stored as data and this data forms the foundation for the security profile of production VM instances launched from them. Any VM instance with unsafe security profile can result in jeopardizing the integrity of applications hosted on that VM instance and in some cases may endanger the integrity of other VM instances on the virtualized host. Hence, at the minimum, VM images have the following security requirements:

- Integrity Protection (VM Images can be created/modified only by authorized administrators and its contents should carry this seal of integrity.)
- Authorized use (limiting the administrators who are allowed to check out images from the Image repository and launch VM instances.) This is to prevent a phenomenon known as “VM Sprawl” which may result in proliferation of unauthorized VM instances.

The protection options that can cover the above requirements are [1]:

- The integrity of the VM images can be protected using checksums or digital signatures.
- Robust access control scheme & assignment of names with well-defined semantics to VM images (e.g., association with an enterprise project), are two ways to control VM sprawl. The second option helps to trace the presence of a particular VM instance to a specific corporate effort (Project, Cost Center etc).
- The enterprise should develop a “VM Gold Standard” in terms of OS Distribution Version/Patch #, Resource Profile etc. and any VM image should be checked for conformance to this standard before deployment.
- The file containing metadata information about VM images should be logically separate from VM Images themselves and access to both of them subject to access control.

The above protection options can be realized through COTS crypto modules and Configuration Management Tools. The integrity protection option has to be implemented by the stakeholder who created the VM image repository (IaaS cloud provider or IaaS cloud consumer) while authorized use of VM images for launching has to be implemented by IaaS cloud consumer.

III. CONFIGURING VM INSTANCE OS (GUEST OS)

Securing the OS installed in the leased VM instance (Guest OS) is the first security task of the IaaS cloud consumer. The minimal set of security requirements for this task is:

- The OS modules configured should result in a hardened installation – one that contains the minimal modules from the OS distribution [1] that will provide the functionality for the purpose for which the VM instance is going to be deployed.
- It should contain the latest version of the OS distribution as well as the latest patches.

The protection options available to IaaS cloud consumers for meeting the above requirements are straightforward: (a) Verify whether any of the pre-defined VM images offered by the IaaS cloud provider meets the above security requirements or build their own VM image meeting the above requirements and (b) deploy only those compliant VM images as their VM instances in the cloud provider infrastructure. Being an automated verification process, these options require no further analysis.

IV. CONFIGURING VIRUS/MALWARE PROTECTION FOR VM INSTANCE

Before loading them with applications, VM instances need to be secured with an anti-malware/ anti-virus engine as application execution involves lots of file activity. The security functions expected of these engines are [2]: Monitor file events (e.g., downloads and modifications), periodically scan files resident on the VM, detect viruses and malware using the set of signature files and perform the necessary remediation and/or generate alerts. Remediation may take the form of either deleting or quarantining the malicious files. These requirements can be met using one of the following protection options [2]:

- Run an Anti-virus engine on each VM instance
- Run an Anti-virus engine as a Virtual Security Appliance (VSA) in a specially hardened security VM that uses the hypervisor introspection API to scan and monitor file-level events in all VM instances in that virtualized host

Obviously, the second option holds more advantages than the first because:

- It consumes less resources overall compared to a in-VM solution
- Easier maintenance due to a single copy of anti-virus engine and signature files running in a VSA on a hardened VM
- Uniform application of policies across a set of VM instances since policies are specified centrally at one location
- Ability to add sophisticated logging (generating logs conforming to a standard syslog format) and auditing capabilities since the anti-virus engine is not running in a production VM instance and hence not likely to hog the resources and affect application performance.

However, running an anti-virus engine as a VSA can only be done by IaaS cloud provider as it uses the introspection API of the hypervisor to which individual IaaS cloud consumers cannot be provided access. At the same time, an IaaS cloud consumer cannot hand over the task of providing anti-virus, anti-malware protection through a VSA to the IaaS cloud provider as the latter will gain visibility into all files belonging to the former, thus potentially compromising the confidentiality of enterprise assets.

V. CONFIGURING VM INSTANCE ACCESS PROTECTION

The first security requirement for any server after loading and configuring application is access protection and the VM instance (virtual server) is no exception to this. VM instance access protection requirements can be met through following protection options:

- Establishing a secure session using a secure access protocol such as SSHv2 or TLS/SSL.
- Access to VM instances using multi factor authentication with one of the authentication factors being “what you have” type consisting of a public key certificate [3]
- Enabling privileged access (e.g., using SSH) to VM instances only from IaaS cloud consumer’s corporate network (e.g., specifying the sub network from which SSH access (using port 22) is possible)

The criteria to look for in the above protection options are: (a) Strength of cryptographic keys supported in SSH & TLS solutions and (b) the entropy of authentication secrets.

VI. CONFIGURING VM INSTANCE LIFECYCLE OPERATION PROTECTION

One of the core class of functions that administrative users of IasS cloud consumer perform is Lifecycle operations on their VM instances – Start (Launch), Suspend, and Stop (Terminate). These operations are performed using API calls to the hypervisor management interface. The security requirements for these operations are:

- Ability to restrict the set of administrators who can make these API calls
- Sending the API calls with integrity and in some certain instances in a confidential way

The protection options for meeting these requirements and an analysis of implementation issues are given below:

Restricting the set of API calls a particular IaaS cloud consumer user can invoke can be enforced using conventional access control mechanisms. Very often the Identity & Access Management system provides the ability to create groups or roles to which a set of allowable API calls can be assigned. By assigning an individual IaaS cloud (administrative) consumer to one or more of these groups or roles, that individual’s access rights can be restricted only to the set of permissions assigned to those groups or roles. Further API calls are protected by channeling them through a dedicated management network that is isolated from the network that carries the traffic for applications running on VM instances.

To ensure that lifecycle operations on VM instances have originated from the authorized IaaS cloud (administrative) consumer user and have not been tampered with, while being submitted across the network, commercial IaaS cloud services require that API calls to perform those operations are digitally signed and the interfaces are on a dedicated management network [3]. To obtain this capability, an IaaS cloud (administrative) consumer user has to generate a private cryptographic key and have the corresponding public key vouched for through a Certificate issued by a trusted Certificate Authority (CA). In addition, if the IaaS cloud (administrative) consumer user wants to send the API calls with confidentiality protection, he/she has to establish a SSL session with management interface provided by the IaaS cloud provider.

VII. CONFIGURING VM INSTANCE ISOLATION

The business value for IaaS consumers to lease VM instances from an IaaS cloud provider comes from the ability to architect a multi-tier enterprise application by leasing multiple VM instances. To protect these applications running on different VM instances, IaaS consumers need to have mechanisms for isolating VM instances based on the type of application/application-tier hosted on them. This isolation requirement can be met by the following protection options. A detailed analysis of these protection options follow:

- Isolation through Firewall Configurations
- Isolation using VLAN ID/Portgroup

A. Isolation through Firewall Configurations

Firewalling functions perform monitoring and place restrictions on both inbound and outbound traffic to and from specific VM instances. The argument for placing restrictions on outbound traffic is that if a VM instance belonging to a consumer is compromised, it could be used as a launching pad to attack other VM instances belonging to the same IaaS cloud consumer because of pre-established connections of a multi-tier application.

There are two firewalling architectural options:

- Firewall based at the Virtual Network layer
- Firewall housed on VM instance

Firewalls implemented at the Virtual Network layer restrict inbound and outbound traffic to and from targeted VM instances and have the following architecture & features:

- They generally consists of two components [4]: (a) A Hypervisor kernel module that forwards all or selected (based on a set of rules) packets coming into a virtual network card (vNIC) of every VM in a virtualized host to a firewall that is run as a VSA and (b) A firewall that is run as a VSA on a specially-hardened VM instance that receives packets from the hypervisor kernel module (referred to in (a) above) and enforces traffic restrictions (allow or restrict). These restrictions are enforced based upon traffic filtering rules centrally defined on a virtual infrastructure management server and pushed into this VSA running on each virtualized host.
- They make use of VMI (Virtual Machine Introspection) capability of the hypervisor [5] to gain visibility into the network traffic flowing in and out of VM instances and reside between the physical network interface of the virtualized host and the vNICs of VM instances
- Traffic restriction policies can be enforced at the following level of granularity: (a) Based on TCP 5-tuple (Source IP, Destination IP, Source Port, Destination Port, Protocol type (e.g., TCP/UDP), (b) Application Ports & (c) Administrator-defined Security Groups (Cluster (a group of virtualized hosts), Resource pool (group of VMs) and Port Group/VLAN (that can be defined at the level of virtual switches within a virtualized host))

Since, in a given virtualized host, VM instances belonging to tenants (consumers) run in a typical IaaS cloud service, the virtual network configuration has to be under the control of IaaS cloud provider, and hence, a virtual network-based firewall can only be installed and run by the IaaS cloud provider. However, the cloud provider can provide the capability for selective administration of this firewall to cloud consumers to specify firewall traffic rules pertaining to their own VM instances in that cloud service.

A typical scenario for IaaS cloud consumer to use a virtual network-based firewall provided by the IaaS cloud provider is the following:

- IaaS consumer runs three VM instances one each for three application types – Web Server, Application Server and Database Server. Each of these types can be designated as a Security Group.
- Restrictions on external access to VM instances belonging to each of the security group can be specified [3]. For example, access to VM instances in the Web Server Security Group can be allowed only on ports 80/443 either with no restrictions on the IP source address or restricting it to the corporate IP network of the IaaS cloud consumer. Similarly access to VM instances belonging to Application Server Security group or Database Security Group can be restricted to only designated administrators on the corporate IP network and that too only to ports needed for establishing secure sessions (e.g., SSH on Port 22).
- Restrictions on VM instances from other VM instances run by the same cloud consumer can be specified based on the architecture of the multi-tier application. For example access to VM instances in the Application Server Security Group can be restricted to VM instances from the Web Server Security Group [3]. Similarly access to VM instances in the Database Security Group can be restricted to VM instances in the Application Server Security Group.

In order to obtain enhanced security assurance for applications running on VM instances, the IaaS cloud consumer should also augment the capabilities provided by virtual network-based firewalls with host-based firewalls running on their VM instances, though it may take away some valuable CPU cycles that could otherwise be dedicated to production applications.

B. Isolation using VLAN ID/Portgroup

Another set of network level isolation (protection) options that IaaS consumers can look for in an IaaS cloud provider infrastructure consists of the following:

- Virtual LAN ID (VLANID)-based Isolation
- Portgroup Isolation

It may be difficult in many large scale IaaS cloud provider environments to provide isolation for multi-tenant VMs based on VLANIDs due to the following:

- VLANIDs are complex to configure and the number of IDs are limited (e.g., 4000)
- The security profiles of IaaS cloud consumer VMs are bound to change continuously requiring frequent re-configuration of VLANs.
- In providing isolation using VLANs, the enforcement point is a physical firewall or the physical switch. This requires routing all traffic originating from or coming into a VM to the physical NIC of the virtualized host and on to the trunk port of a physical switch, thus increasing the latency of communication packets with consequent increase in application response times.

Because of the above difficulties in providing isolation through VLAN IDs, IaaS cloud providers could be providing isolation between multi-tenant VMs through a feature called Portgroup Isolation [4]. A portgroup is a software-defined port on the software defined virtual switch on a virtualized host. In the Portgroup isolation, the required isolation between multi-tenant VMs could be provided by assigning the VMs of each tenant to a different portgroup. Isolation between two VMs belonging to two different tenants is obtained by assigning their corresponding VM instances to different portgroups and by installing a gateway software that routes inter-VM traffic based on Portgroup IDs.

VIII. COMPREHENSIVE DATA PROTECTION IN IAAS CLOUD SERVICE

The complete set of data in a typical IaaS cloud service consists of the following: (a) Data generated/used by applications running in the VM instances and (b) Data defining the entire running VM instance. Examples of former type of data are: (a) Data originating from cloud consumer's client software (e.g., Data Input to an application running in a VM instance) and (b) Data originating from a VM instance (e.g., an application in a VM instance that generates data). Let us now look at the security requirements and the available protection options that IaaS consumers have for these two types of data.

A. Data Protection for Data Generated/Used by VM Instances

The storage artifact available to IaaS cloud consumer for associating storage with their VM instances is the concept of "Virtual Disks". However the mapping of these logical storage units (i.e., virtual disks) to physical storage artifacts is entirely under the control of IaaS cloud provider. For example, the virtual disks may map to: (a) local physical disks of the virtualized host (b) remotely located NFS file volumes or (c) remote block storage devices accessed through Internet Small Computer System Interface (iSCSI) or Storage Area Network (SAN) protocols. Irrespective of the storage technology deployed by IaaS cloud provider, protection of data is entirely the responsibility of the IaaS cloud consumer and may span the following requirements:

- Data in Transit protection – This applies to: (a) data in transit between IaaS consumer's client software and VM instance and (b) data travelling between two VM instances of the same IaaS consumer. The protection options for both these classes of in-transit data can be provided through the capability to set up secure sessions (to or between VM instances) using protocols such as SSHv2 or TLS/SS (described under VM instance access protection) as these protocols enable data to be both encrypted and digitally signed.
- Data at Rest protection – This applies to: (a) unstructured data stored under a file system defined over a virtual disk volume and (b) structured data stored by DBMS engine running in VM instances. Regardless of the type of data,it can be protected from unauthorized access/modification through the following: (a) Access Control – Using access control mechanisms available in file systems or DBMS engines, IaaS consumer administrators can define permissions at the appropriate level of granularity for their cloud users, and (b) Encryption. Generally most cloud offerings leave it to the IaaS cloud consumer to encrypt their data. The practical limitation that IaaS consumers encounter while deploying an encryption mechanism to encrypt data going into the virtual disks associated with their VM instances is that the encryption engine and the associated key management engine have to be run in the IaaS cloud provider's infrastructure [6] – most likely in a dedicated VM instance for performance reasons.
- Data Durability/Recoverability protection – This applies to protecting data due to corruption and

loss/theft of the media holding the data. The most common technique applied is a mechanism for robust backup and recovery capability for restore/recovery of data in case of data corruption incidences. This kind of protection again may apply to two types of data. They are: (a) Data generated/used by applications running on VM instances (for which transit/acess/storage protection have been discussed in the previous sections) and (b) Data defining the entire VM instance itself. For data of the first type, the IaaS cloud consumers should have to employ on their own, either a data backup/recovery solution or rely on a Cloud Storage service that may be offered by the same IaaS cloud provider or some other cloud provider. This is due to the fact that such a backup/recovery service is usually not offered as an integral part of IaaS cloud service. Regarding durability/recoverability protection for "data that defines the entire VM instance", there are several options and hence we devote a separate section to discuss this.

B. Data Protection for Files that define the VM instance

Since VM images are a set of data files (refer section II), VM instances that are launched from those images are made up of the same set of files augmented with files that capture the state of the VM instance such as virtual memory swap files and log files. The following backup & recovery solutions are available for backing up files that define the VM instance:

- Image-level backup with Snapshot capability [7]: In this backup mode, the entire contents of the virtual disk defined in the VM instance is backed up as an Image. This backup is done without going through the guest OS of the VM instance. In this type of backup, in order to obtain a transaction-level consistency of data in the various disk blocks, the following procedure is adopted: First the VM instance is subject to quiescing using a special driver that runs inside the guest OS. This action momentarily pauses the running processes on the VM instance and forces the guest OS and applications to write any pending data to disk. Once that is complete, a virtualization-specific process called Snapshot is performed using a tool at the hypervisor layer. The effect of this snapshot process is that any subsequent writes by the running VM instance will be written to a temporary virtual disk file, thus freezing the contents of the original disk file. After the image-level backup of the original (virtual) disk file is completed, the contents of the temporary virtual disk file is merged block by block with the original disk file to bring the contents of the VM instance up to date and the snapshot is also deleted. The advantage of image-level backup is that it not only makes the backup a simple process but also the restore as well since the image-level backup can simply be copied to any other storage device attached to any other virtualized host and the VM restarted in the new virtualized host.
- File-level backup: This backup is done at the level of individual files that constitute a running VM instance and is done through VM instance OS (Guest OS). The downside of this type of backup is that it may take away some valuable CPU cycles allocated to a VM instance which might otherwise be used by applications running on them.

IX. CONCLUSIONS AND BENEFITS

In this paper, we identified the typical set of uses cases for IaaS cloud consumer, the associated security requirements for its safe operation and analyzed the protection options available to meet those requirements based on security solutions/ technologies offered by IaaS cloud providers and third parties. The security requirements, protection options and the focus of analysis of those protection options in terms of feature set/deployment architecture are summarized in Table 1. The primary benefits of the analysis of the protection options are: (a) Provides a realistic picture of security protection options the IaaS cloud consumer can deploy, (b) Provides a realistic assessment of IaaS cloud provider's security capabilities and those that can be demanded and (c) Enables IaaS cloud consumer to choose the most appropriate security configuration for their VM instances depending upon the profile of the applications running in them. It must be mentioned that in spite of the due diligence in analysis of protection options, there are two areas in which obtaining the necessary security assurance is difficult. They are: (a) Device Drivers employed in Hypervisors and (b) Remote Management software [8] deployed to facilitate remote execution of VM Lifecycle operations.

TABLE 1. SUMMARY OF PROTECTION OPTIONS ANALYSIS FOR IAAS USE CASES

Section – Use Case	**Security Requirements**	**Protection Options**	**Features, Security Strengths & Architectural Foundation**
Checking out VM Images	(a)Integrity of VM Image files (b) Preventing Unauthorized VM launches	(a) Digitally signed VM Images (b) Name Space Control, Gold Standard Configuration, Separation of Data & Metadata & Access control	(a) Strength of Cryptographic Signing Keys, Secure session with Image Repositories (b) Robust Configuration Mgmt Utilities
Configuring VM Instance OS (Guest OS)	(a) Hardened OS Distribution (b) Latest Patches	Verify that IaaS provider's pre-defined images meet the IaaS consumer's Gold Standard	N/A
Configuring Virus / Malware Protection	(a) Monitor File Events, Scan Files (b) File Remediation	Run Anti-Virus Engine on each VM instance (or) Run one copy as a Security Virtual Appliance	Security Virtual Appliance enables uniform application of policies & Consumes less resources
Configuring VM instance Access Protection	(a) Secure Session (b) Multifactor Authentication	(a) SSHv2 or TLS/SSL (b) PKI-based Authentication or One-time Password Token	Strength of Encryption / Signing Keys & Entropy of Authentication secrets
Configuring VM instance lifecycle operation	(a) Limit API calls to authorized admins (b) Sending API calls with integrity & Confidentiality	(a) Identity & Access Management System (b) Digitally signed API calls & Dedicated Management network	(a) Creation of Admin Groups/Roles (b) Public Cryptographic Keys on Virtualized Host
Configuring VM Instance Isolation	Restricting the type of inbound & outbound traffic between VMs	(a) In-VM or Virtual Network based Firewall (b) Isolation using VLAN ID/ Portgroup	(a) Virtual Network based Firewalls use Hypervisor's Introspection API (b) Portgroup Isolation solutions function as Application Gateway
Comprehensive Data Protection in IaaS cloud service	Confidentiality & Integrity protection for in-transit & stored data (generated by & constituting VM instance)	(a) Secure Session Protocols (in-transit data) (b) Access Control + Encryption (stored data)	(a) Strength of cryptographic session keys (b) Strong Authentication + Strong Encryption Keys

REFERENCES

[1] T. Brooks, C. Caicedo, and J. Park, "Security Challenges and Countermeasures for Trusted Virtualized Computing Environments," World Congress on Internet Security, Jan 2012, pp. 117-122.

[2] J. D. Sherry, "Continuous Monitoring in a Virtual Environment," Nov 2013, http://www.trendmicro.com/cloud-content/us/pdfs/business/reports/rpt_continuous-monitoring-virtual-environment.pdf [Retrieved: March, 2014]

[3] "Amazon Web Services: Overview of Security Processes," March 2013, http://aws.amazon.com/security/ [Retrieved: February, 2014]

[4] "The Technology Foundations of VMware vShield," Oct 2013, http://www.vmware.com/files/pdf/techpaper/vShield-Tech-Foundations-WP.pdf [Retrieved: April, 2014]

[5] Q. Chen, et al, "On State of the Art in Virtual Machine Security" IEEE Southeastcon, 2012, pp. 1-6.

[6] R. Chandramouli, M. Iorga, and S. Chokhani, "Cryptographic Key Management Issues and Challenges in Cloud Services," NIST IR 7956, Sept 2013, http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7956.pdf [Retrieved: April, 2014]

[7] E. Siebert, "The Expert Guide to VMWare Data Protection and Disaster Recovery," July 2012, http://www.veeam.com/wp-vmware-data-protection-disaster-recovery-expert-guide.html [Retrieved: March, 2014]

[8] D. P. Botero, J. Szefer, and R. B. Lee, "Characterizing Hypervisor Vulnerabilities in Cloud Computing Servers," International Workshop on Security in Cloud Computing, May 2013, pp. 3-10.

A Centralized Architecture for Energy-Efficient Job Management in Data Centers

George Perreas, Petros Lampsas
Department of Computer Engineering
Technological Educational Institute of Central Greece
Lamia, Greece
e-mails: {georgeperreas@gmail.com, plam@teilam.gr}

Abstract—**In this work, we present a centralized monitoring entity that attempts to reduce power consumption in Internet Data Centers (IDCs) by employing live Virtual Machine (VM) migrations between blade servers. To perform live VM migrations, usage statistics collected by servers are evaluated and the servers that may be offloaded are selected. VMs that belong to the servers that may be offloaded are scattered to other active servers provided that the user-perceived performance is sustained. Overall, jobs submitted by users should be consolidated to as few servers as possible and the servers that host no job can be put in stand-by or hibernate mode, thus achieving an overall power reduction. Data Center management authorities may take advantage of such a monitoring entity in order to decrease energy consumption attributed to computing, storage and networking elements of data centers.**

Keywords–Data Center Energy Efficiency; Energy efficient Job Management; Virtual Machine Migrations.

I. INTRODUCTION

Data Centers are facilities used to host cloud computing resources comprising computing systems and associated equipment, such as networking, storage, security and environmental control systems. These computing resources can be accessed through Internet. An IDC, usually, deploys hundreds or thousands of blade servers, densely packed to maximize space utilization. It generally includes redundant or backup power supplies, redundant data communications connections, environmental controls (e.g., air conditioning, fire suppression) and security devices. To protect these systems and their vital functions, however, data centers also employ energy-intensive air conditioning systems, fire suppression systems, redundant/backup power supplies, redundant Internet connections, and security systems.

Running services in consolidated servers in IDCs provides customers an alternative to maintaining in-house equipment and personnel that provides services. IDCs achieve economies of scale that amortize the cost of ownership and the cost of system maintenance over a large number of machines. However, with the rapid growth of IDCs in both quantity and scale over the last few years, the energy consumed by IDCs, directly related to the number of hosted servers and their workload, has been skyrocketed [1].

The most commonly used metric to determine the energy efficiency of a data center is Power Usage Effectiveness (PUE). This simple ratio is the total power entering the data center divided by the power used by the information technology equipments. However, according to an Uptime Institute survey [2], only half of the large organizations (over 2000 servers) measure PUE in a detailed fashion, while only 18% of smaller data centers (those with fewer than 500 servers) had any PUE focus. This is an indication that there is a lot of space for optimizations, as far as IDC energy consumption is concerned.

In IDCs, servers, storage and networking systems may get underutilized during daily operation, especially in cases where job population, resource utilization, arrival and completion rates vary significantly over time. This is not a problem from the scheduling algorithms point of view that distribute load by employing as many servers as possible in order to minimize e.g. job completion time. In the general case, despite the efficiency of scheduling and job placement algorithms when new jobs arrive in an IDC, job completion and/or varying resource needs during job lifetime may create the opportunity to consolidate jobs to servers. By consolidating jobs to some selected servers until the rest of them possess no more jobs, these jobless servers may be put in low power consumption mode or even turned off; thus, achieving decreased energy consumption in the DC. Server consolidation is performed up to the point that the servers selected to host the jobs are fully exploited, as far as their computing power is concerned, without violating user perceived performance and Service Level Agreements (SLAs).

Technologies that tackle energy efficiency in IDCs are network power management, chip multiprocessing energy efficiency, power capping and storage power management, to name a few. VM technology can be considered as a software alternative to the approaches that tackle energy efficiency in IDCs. VM technology (such as VMWare [4], Xen [3]) enables multiple OS environments to co-exist on the same physical machine, albeit in strong isolation from each other.

Despite their technical differences, both technologies support migration of virtual machines (i.e., VM transfer across physical computers). There are two types of migration: regular migration and live migration. The former stops a VM, moves a VM from one host to another and then restarts the VM, while the latter transfers the VM without ceasing to offer the service during transition.

VMs may be transferred between physical machines, without user intervention, when certain conditions apply to the physical machine that originates the migration. VM and VM migration technologies exhibit great potential to efficiently manage workload consolidation, and therefore, improve the total IDC energy efficiency.

In this work, we implemented and tested Open Data Center Manager (ODCM), a centralized mechanism that decides VM migrations (and consequently migrations of every job executed in this VM) according to a multi-criteria decision making algorithm and gathered monitoring information concerning computational load incurred in an IDC. VM migrations are decided in such a way that results in server consolidation, i.e. all the jobs submitted to a data center run to as few servers as possible, taking into account Service Level Agreement between data center managers and end users. We conducted an initial evaluation of ODCM in a, relatively small, cluster and initial results depict that ODCM may result in increased energy efficiency through server consolidation by employing live VM migrations.

The rest of the paper is structured as follows. Section II gives an overview of related work. Section III presents the system model and gives a problem formulation. The building blocks of ODCM are also described there. Section IV outlines implementation issues. Finally, Section V concludes the paper.

II. RELATED WORK

Energy efficiency in data centers, as far as computing elements are concerned has been studied in different contexts. Approaches adopted by researchers fall in two broad categories: i) solutions that attempt to minimize power consumption in the hardware elements of the IDC and ii) software solutions that manage IDCs and schedule jobs on servers taking into account not only performance but the minimization of the overall energy consumption.

The first type of solutions can be generally classified under power management approaches. These options include the Dynamic Voltage/Frequency Scaling (DVFS), turning On/Off system components, the Chip-Multiprocessor approach, etc. Orgerie et al. [6] address theoretical and experimental aspects of energy efficiency in large-scale distributed system both in the power management (study of On/Off models) as well as in the virtualization domain.

DVFS is a prominent approach to adjusting the server power states. Horvath et al. [11] have studied how to dynamically adjust server voltages to minimize the total power consumption while at the same time end-to-end delay constraints in a Multi-tier Web Service environment, are met. Barroso and Hölzle [13] studied how to use Chip Multi-Processor (CMP) to achieve energy-proportional designs. Raghavendra et al. [12] suggested coordinating individual approaches in software and hardware power management in order to efficiently manage energy in multiple levels in data center environments.

The second category of solutions involves mainly job assignment to servers as well as VM migrations to achieve energy-efficiency. Liu et al. [5] proposed an architecture that enables comprehensive online-monitoring, live virtual machine migration, and VM placement optimization, in order to reduce power consumption in Data Centres. Wood et al. [7] proposed the CloudNet architecture that builds a pool of geographically distributed data centres through efficient WAN VM migrations. This approach unifies data centre equipment and offers enterprises a seamless and secure application execution environment.

Chaisiri et al. [8] proposed Optimal Virtual Machine Placement algorithm that can be used in renting resources between cloud providers in order to reduce user costs for deploying applications in data centres. Finally, Tarighi et al. [9] adopted and deployed an approach similar to ours in the context of cluster computing which, however, does not aim at decreasing power consumption.

The approach adopted in ODCM falls in the second category. To the best of our knowledge, this is the first attempt to tackle energy efficiency in data centres by a centralized entity that consolidates applications and required data by live migrating VMs between hosts within an IDC.

III. SYSTEM MODEL

In this work, we assume Internet Data Centers that host compute and storage entities. These entities host applications and data associated to the applications. Customers receive a specific Quality of Service (QoS) as far as application execution and/or perceived response times are concerned, according to SLAs signed between the customer and the IDC service provider.

Compute entities that reside in IDCs host VMs, the execution environment for customers' applications. Both data needed for application execution and the application code are stored in the same IDC; however, replicas may be created among IDCs owned by the same service provider. VMs and hosted applications may be migrated among compute entities that reside in the same or different IDCs. We suppose that, in order for an application to run on an IDC, the associated data must reside in the same IDC. We also suppose that, SLA for a submitted job is satisfied if a certain amount of the host's computing power is assigned to the VM that hosts this job.

Applications that arrive in an IDC are placed in the most loaded available server that, after hosting the newcomer application, still operates below a certain, user-defined, threshold and meets the requirements derived from the hosted applications SLAs.

ODCM works in a periodic fashion. After a certain period of time set by the administrator, ODCM is invoked attempting to consolidate applications. The servers that will be attempted to be put in low-power mode or hibernated are selected according to a multi-criteria method, i.e., TOPSIS [10]. When the servers that will act as originators of migrating VMs are selected, a bin packing heuristic is executed that produces a series of live migrations that lead to the applications being run in as few servers as possible.

The next phase comprises implementation of live migrations in order to achieve the result of bin packing heuristic. This step perhaps involves VM placement rearrangement, i.e., migrations between operating servers not selected as originators for migrations, for space creation. Each operating compute element is checked to see whether he can act as a host to a migrating VM. If there is no host that possesses the required resources to run a VM, then compute elements are checked to see whether they can offload some VMs to other operating compute elements in order to create enough free space for the migrating VMs.

A. *Topsis*

The TOPSIS method is a technique for order preference by similarity to ideal solution, proposed by Hwang and Yoon [10]. The ideal solution (also called the positive ideal solution) is a solution that maximizes the benefit criteria and minimizes the cost criteria, whereas the negative ideal solution (also called the anti-ideal solution) maximizes the cost criteria and minimizes the benefit criteria. The so-called benefit criteria are those for maximization, while the cost criteria are those for minimization. The best alternative is the one that is closest to the ideal solution and farthest from the negative ideal solution.

The TOPSIS procedure is divided in five steps that are described below.

Step 1. A table with the data that will be used for decision making is constructed and normalized.

$$R_{ij} = \frac{x_{ij}}{\sum x_{ij}^2} \quad for\ i = 1,...,hosts\ and\ j = 1,..,criteria \qquad (1)$$

The values x_{ij} are the weighted moving averages of the values reported by each server by the monitoring component (e.g., CPU usage, VM usage).

Step 2. Table R is taken as input from step 1 and is weighted using the matrix with the weights that correspond to the criteria being set.

$$V_{ij} = W_j * R_{ij} \qquad (2)$$

In our case, criteria for classifying overloaded servers (in decreasing order of significance) are CPU usage (%), CPU Speed, Free Cores, Total Cores, Total RAM, Free RAM and Total VMs Executing. For deciding the VMs to migrate from overloaded servers, a different set of criteria is introduced: Virtual CPU Usage (%), Virtual RAM Usage and Virtual Cores (used by a VM). Weights, set after experimenting with several potential values, vary from 1 to 9, depending on the significance of each criterion.

Step 3. Ideal solution is the one that is closest to the ideal solution and farthest from the negative ideal solution. The ideal solution can be calculated as follows (Eq. 3a and 3b):

$$A^w = \left\{ \left\langle \max\left(V_{ij} \middle| i = 1,2,...,m\right) \middle| j \in J \right\rangle, \left\langle \min\left(V_{ij} \middle| i = 1,2,...,m\right) \middle| j \in J^{'} \right\rangle \right\}$$

$$\equiv \left\{ V_{wj} = 1,2,...,n \right\} \to A^w = \left\{ V_1^w, V_2^w, ..., V_j^w, ...V_n^w \right\}$$

where w is the worst ideal solution and b the best ideal solution.

The negative ideal solution is:

$$A^b = \left\{ \left\langle \min\left(V_{ij} \middle| i = 1,2,...,m\right) \middle| j \in J \right\rangle, \left\langle \max\left(V_{ij} \middle| i = 1,2,...,m\right) \middle| j \in J^{'} \right\rangle \right\}$$

$$\equiv \left\{ V_{wj} = 1,2,...,n \right\} \to A^b = \left\{ V_1^b, V_2^b, ..., V_j^b, ...V_n^b \right\}$$

$J = \left\{ j = 1,2,...,n \middle| j \right\}$, for criteria with positive impact, and

$J^{'} = \left\{ j = 1,2,...,n \middle| j \right\}$, for criteria with negative impact.

Step 4. Euclidean distance between every solution and the ideal and negative ideal solution respectively is calculated as follows:

$$S_i^w = \sqrt{\sum \left(V_j^w - V_{ij} \right)^2}\ , where\ i = 1,2,...,m \qquad (4a)$$

$$S_i^b = \sqrt{\sum \left(V_j^b - V_{ij} \right)^2}\ , where\ i = 1,2,...,m \qquad (4b)$$

Step 5. The following amount depicts how close a solution is to the ideal solution (the best choice is the one that is closer to 1):

$$C_i^w = \frac{S_i^b}{S_i^w + S_i^b},\ where\ 0 < C_i^w < 1 \qquad (5)$$

B. *Bin Packing*

Note that, in order to consolidate VMs in as few servers as possible, we actually need to implement a heuristic for a variation of the bin packing problem. In its general form, the bin packing problem (a combinatorial NP-hard problem [14]) can be stated as follows: objects of different volumes must be packed into a finite number of bins or containers each of volume V in a way that minimizes the number of bins used.

The analogy in our problem setting is to consider bins as servers (each of different VM hosting capacity) and objects as VMs that must be hosted on as few servers as possible. The bin packing heuristic is invoked when ODCM is executed and at least a server exceeds a (tunable) CPU utilization limit. All servers that exceed this CPU utilization limit are sorted by TOPSIS from the most appropriate to the least appropriate to be offloaded. These are servers that will act as originators for live VM migrations. After this step, TOPSIS is run again to produce a list of servers that can receive VMs in decreasing order of suitability to act as receivers. Finally, VMs that must be migrated are sorted by TOPSIS from the one that imposes the more load to the CPU to the one that imposes the lesser load to the CPU. Each server in the receiver list is checked and if it possesses enough free resources (CPU cores, free memory space) to host VMs that will be offloaded by the originator server, VM migration commences.

If the server that is selected to be a candidate receiver cannot host VMs that are to be migrated, an additional check is performed in order to find out if the candidate receiver can free enough resources by migrating VMs to other candidate receivers. This attempt to free resources in one server will (perhaps) trigger a series of recursive migrations originated from the servers that are selected to get checked if they can receive a migrating VM. If these checks result in freeing enough resources the migration is performed; otherwise ODCM concludes that no migration can be carried out. The cost for each live migration is considered to be negligible, since migrations are performed within an IDC, over a high-speed local area network.

IV. ODCM IMPLEMENTATION

ODCM is implemented as a client-server application using the Java programming language and UDP transport (Fig. 1). Values obtained from the individual servers concerning CPU and VM virtual CPU utilization are stored in a MySQL database after being processed to obtain the weighted moving average. In this way the values stored take into account not only the last value reported by the monitoring component, but all the reported values within a time period. Older values contribute less to the computed value whereas the more recent the obtained value, the greater the contribution to the value computed and stored.

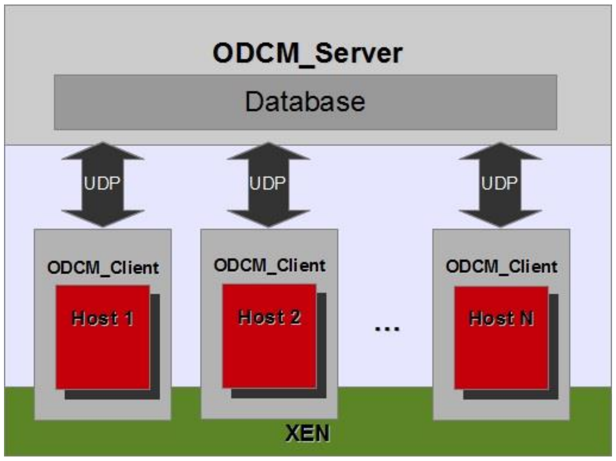


Figure 1. ODCM Architecture

Data management is performed by using the Java Persistence API (JPA). In JPA, there exist persistence entities, i.e., lightweight Java classes, whose states are typically persisted to a table in a relational database.

After the invocation of ODCM, the servers that are loaded above a user defined threshold are selected and the VMs they host are migrated. 20% of these servers (or at least one server) are placed in low power consumption mode in order to be ready to execute new jobs that cannot be hosted to any of the already operating servers. The remaining servers are hibernated, and when one of the servers that are set in low power consumption mode resumes normal operation, one of the hibernated servers is chosen randomly to join the pool of servers in low power consumption mode that are ready to undertake newcomer jobs.

V. CONCLUSION AND FUTURE WORK

ODCM is a periodically executed service that attempts to consolidate applications in as few servers as possible in order to conserve energy. Lab tests to a relatively constrained setting (consisted of 5 servers) revealed that the attempted consolidation (and thus, the resulting power consumption reduction) is achieved and VM live migrations are decided and executed in a timely manner.

ODCM execution could also be event-driven, triggered when specific simple or metrics reach a certain threshold. We plan to evaluate these two approaches, i.e., periodic execution vs. asynchronous event-driven execution and also check which of the metrics are giving best results assuming different workloads.

Since current solutions for VM migration incur service disruption because they slow-down storage I/O operations during migration, we intend to accompany scheduling algorithms with data allocation and replication techniques so that the data required for the computation be as "near" to the computation as possible.

Extensive testing of ODCM using appropriate infrastructure should take place. ODCM will be extended with data consolidation, i.e., migrating data needed for computations to as few storage servers as possible. ODCM could also be extended to minimize energy consumption by migrating tasks and the relevant data among IDCs that belong to the same owner, taking into account time zone job submission statistics. Furthermore, the source of energy provided to the IDC could be taken into account (i.e., it is preferable to migrate jobs to IDCs that are powered using renewable energy sources, as long as user-perceived performance remains within acceptable levels).

ACKNOWLEDGMENT

This work is implemented through the Operational Program "Education and Lifelong Learning" and is co-financed by the European Union (European Social Fund) and Greek national funds.

REFERENCES

[1] Department of Energy (DoE), Data Center Energy Efficiency Program, http://www1.eere.energy.gov/manufacturing/tech_assistance/pdfs/doe_data_centers_presentation.pdf, April 2009 [retrieved: April,2014].

[2] Uptime Institute Survey Results, http://uptimeinstitute.com/2012-survey-results, 2012 [retrieved: April,2014].

[3] Xen User Manual, http://bits.xensource.com/Xen/docs/user.pdf [retrieved: April,2014].

[4] http://www.vmware.com/ [retrieved: April,2014].

[5] L. Liu, H. Wang, X. Liu, X. Jin, W. Bo He, Q. Bo Wang, and Y. Chen, "GreenCloud: a new architecture for green data center," Proc. of the 6th International Conference on Autonomic Computing and Communications industry session (ICAC-INDST 09), 2009, pp. 29-38.

[6] A.-C. Orgerie, L. Lefevre, and J.-P. Gelas, "Demystifying energy consumption in Grids and Clouds," Proc. of the International Conference on Green Computing (GREENCOMP 10), IEEE Computer Society, 2010, pp. 335-342.

[7] T. Wood, K. K. Ramakrishnan, P. Shenoy, and J. van der Merwe, "CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines," Proc. of the 7th ACM SIGPLAN /SIGOPS International Conference on Virtual Execution Environments (VEE 11). ACM, 2011, pp. 121-132.

[8] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal Virtual Machine placement across multiple Cloud providers," Proc. of the Services Computing Conference (APSCC 09), IEEE Asia-Pacific, 2009, pp. 103-110.

[9] M.Tarighi, S.A.Motamedi, and S.Sharifian, "A new model for virtual machine migration in virtualized cluster server based on Fuzzy Decision Making," Journal of Telecommunications, vol. 1, no. 1, Feb 2010, pp. 40-51.

[10] C.L. Hwang and K. Yoon, "Multiple Attribute Decision Making: Methods and Applications," 1981, New York: Springer-Verlag.

[11] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu, "Dynamic voltage scaling in multitier Web servers with end-to-end delay control," IEEE Trans. on Computers, vol. 56, no. 4, Apr. 2007, pp. 444-458.

[12] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No power struggles: coordinated multi-level power management for the data center," Proc. 13th Intl. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 08), ACM, 2008, pp 48-59.

[13] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," IEEE Computer, vol. 40, no. 12, Dec. 2007, pp. 33-37.

[14] Bin Packing, http://en.wikipedia.org/wiki/Bin_packing_problem [retrieved: April,2014].

Securing the Grid using Virtualization
The ViSaG Model

Pierre Kuonen, Valentin Clément, Frédéric Bapst
University of Applied Sciences Western Switzerland, HES-SO//Fribourg
Fribourg, Switzerland
Emails: {pierre.kuonen, frederic.bapst}@hefr.ch, clementval@gmail.com

***Abstract*—Security in large distributed computing infrastructures, peer-to-peer, or clouds, remains an important issue and probably a strong obstacle for a lot of potential users of these types of computing infrastructures. In this paper, we propose an architecture for large scale distributed infrastructures guaranteeing confidentiality and integrity of both the computation and the host computer. Our approach is based on the use of virtualization and we introduce the notion of *confidence link* to safely execute programs. We implemented and tested this approach using the POP-C++ tool, which is a comprehensive object-oriented system to develop applications in large decentralized distributed computing infrastructures.**

Keywords-virtualization; security in large distributed system; grid middleware.

I. Introduction

Today, more and more applications require having punctually access to significant computing power. The purchasing of High Performance Computing (HPC) hardware is really profitable only in case of frequent usage. There are several alternatives to purchasing HPC hardware. The two most popular are Clouds [11] and Grids [12]. Even if these two approaches are not totally identical, they share at least one difficulty, which is the fact that the user has to trust the resources provider. In the case of Grid infrastructures, this problem is complemented by the fact that the resource provider also has to trust the user to be sure that running user’s tasks will not harm his own resources. This paper will focus on how, by using virtualization, we can guarantee confidentiality and integrity of computing and resource for the user and the resources provider in decentralized distributed computing environments, such as Grid systems.

The main questions we intend to answer are:

- How to ensure the integrity of the user's data and the user's calculations?
- How to ensure the integrity of the host machine?
- How to ensure the confidentiality of the communications?
- How to safely use machines belonging to different private networks in the presence of firewalls?
- How to ensure that different users sharing the same computing resource cannot interfere between each other?

Last, but not least, we want to provide these features while minimizing the loss of performances.

We propose an abstract vision of a secure decentralized distributed computing environment. This vision is based on the notion of *confidence links*. It has been implemented in the ViSaG project (ViSaG stands for: Virtual Safe Grid), which is presented in this paper.

The rest of this paper is organized as follows: Section II details the main security issues we want to address with the ViSaG project. Section III presents the ViSaG model and Section IV presents the POP-C++ model, which has been used to implement the ViSaG model. Section V details the implementation of the ViSaG model using the POP-C++ middleware and Section VI presents the results we have obtained. Finally, Section VII concludes this paper.

II. Current security issues in grid computing

As mentioned in the introduction, there are several security issues in current Grid middleware systems that must be addressed. These issues are detailed below.

A. How to ensure the integrity of the user's data and the user's calculations

When a user submits a computation on a remote machine he must be sure that the owner of the remote resource cannot interfere with his computation, or, at least, if there is interference the user must be aware of it.

B. How to ensure the integrity of the host machine

Consider a user of the Grid willing to provide his computing resources to the infrastructure. As this user does not have a strong control on who will execute a job on his resources, he wants that the middleware guarantees him that the executed jobs cannot access unauthorized resources, cannot harm his resources, and cannot use more resources than he agreed to allocate to them.

C. How to ensure the confidentiality of the communications

We want to secure communications between nodes. First, we want to prevent communications from being seen by any other person/system and also we do not want that anyone could intercept and modify the transmitted data.

D. How to safely use machines belonging to different private networks (presence of firewalls)

One of the most difficult security problems when deploying decentralized Grid middleware is due to the presence of private networks protected by firewalls. Indeed, most of the time available resources in an institution, which could be part of a Grid, are resources located on private networks protected by firewalls. The question is: how to make these resources available without creating dangerous security holes in the firewalls.

E. How to ensure that different users using the same computing resource cannot interfere between each other

We also have to ensure that a user of a remote resource cannot harm processes of other users on the same remote resource.

Usage of virtual machines in conjunction with Grids to address security issues has been already proposed in several papers, one of the first being [3]. Santhanam et al. [5] propose four scenarios to deploy virtual machines on Grid. None of these four scenarios exactly corresponds to our approach, even if the fourth is the closest. Smith et al. [6] propose a Grid environment enabling users to safely install and use custom software on demand using an image creation station to create user-specific virtual machines. Keahey et al. [4] focus on creating, configuring and managing execution environments. The authors show how dynamic virtual environments can be modeled as Grid services, thus allowing a client to create, configure and manage remote execution environments. In all these papers, the problem of deploying virtual machines in a Grid is addressed in a general way, although Santhanam et al. [5] have used Condor to test their scenarios. Our approach is different because the model we propose is closely related to an execution scheme based on the paradigm of distributed object-oriented programming. The proposed solution is specifically designed to solve the problems associated with this model such as the creation and destruction of the remote object (process) and passing remote objects as parameters of remote methods.

III. THE VISAG MODEL

Unlike most existing Grid middleware, the approach proposed in the ViSaG project is based on the fundamental assumption that a Grid infrastructure is a fully decentralized system, which, in a sense, is close the peer-to-peer (P2P) network concept. At the hardware level, a computing Grid is composed of an unknown, but large, number of computer owning computing resources. None of the computers in the Grid has a global view of the overall infrastructure. Each computer only knows a very limited number of neighbors to which it is directly connected by two-way confidence links. A confidence link is a bidirectional channel that allows two computers located at both ends of the link to communicate safely at any time. How the confidence links are established is not part of the ViSaG model, but is a hypothesis which defines our vision of a computing Grid. However, we can give as an example of a confidence link, an SSH channel between two computers whose system managers, or users,

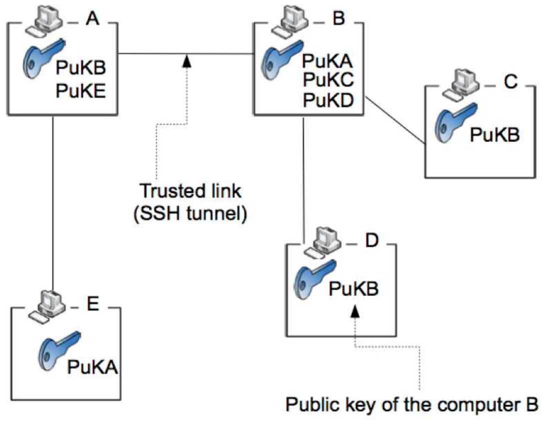


Figure 1. A trusted network using SSH tunnels.

have manually exchanged their public keys. The set of all computers together with all the confidence links form a connected graph, we call it a *trusted network*, whose nodes are computers and edges are the confidence links. In the remainder of this document, when no confusion is possible, we often use the terms nodes and links, respectively, to designate computers and confidence links. Although, usually, computers are volatile resources, we will not address this aspect in this paper, where we made the assumption that, during the execution of a given program, computers participating to this execution do not fail. Finally, we assume that the confidence links are reliable.

Figure 1 illustrates a computing Grid, as defined above, where confidence links have been realized using SSH (Secure Shell [10]) tunnels.

In the ViSaG execution model, computing resources are requested on the fly during execution of the application. Obtainment of requested resources is achieved through the usage of a *resource discovery algorithm* which runs on every node and only uses confidence links. Usually, this algorithm is a variant of the flooding algorithms Details of this algorithm are not part of the model, but is an implementation issue. When a node, we call it the *originator*, needs a new computing resource, it issues a request, which will be handled by the resource discovery algorithm.

When the requested computing resource, provided by a node we will call the *target*, has been found, the originator of the request must contact the target to launch the computation and possibly communicate with it during the computation. For the originator, one possibility would be to communicate with the target by following confidence links. This option is, obviously, very inefficient because it exaggeratedly loads all intermediary nodes which have to route all messages. This is especially true when, during computation, nodes must exchange large data as is often the case in HPC applications. A better solution would be for the originator, to contact the target directly. Unfortunately, it is likely that the originator does not have a direct link (a confidence link) with the target and, in addition, the target does not necessarily desire to create a direct confidence link with the originator. Nevertheless, as the request reached him following confidence links, the target could accept to launch a virtual machine to provide the necessary computing resources for the originator. The virtual machine will act as a sand box for the execution of the remote process. If the virtual machine is not permeable, this will guarantee that the executing node (the target) cannot be damaged by the execution of the remote process and that the computation made by the process cannot be biased by the node which hosts the computation (the target). To summarize, we can make the following statement:

The security of this execution model is only limited by the security offered by the virtual machine and the security offered by the confidence links.

This is the very basic idea of the ViSaG model. The implementation of such a model raises numerous problems that we are going to address in the next sections.

IV. THE POP-C++ EXECUTION MODEL

A Grid computing infrastructure not only consists in hardware but also requires the presence of a middleware which provides services and tools to develop and to run applications on the Grid. Therefore, before presenting how the ViSaG model has been implemented we need to know which Grid middleware our implementation is based on. The ViSaG model, as presented in the previous section, has been implemented in the POP-C++ Grid middleware [1]. In order to achieve this task we had to adapt to the execution model of POP-C++, which is briefly presented below. For more information of the POP-C++ tool please visit the POP-C++ web site: http://gridgroup.hefr.ch/popc.

The POP-C++ tool implements the *POP programming model* first introduced by Dr. Tuan Anh Nguyen in his PhD thesis [2]. The POP programming model is based on the very simple idea that objects are suitable structures to distribute data and executable codes over heterogeneous distributed hardware and to make them interact between each other.

The object oriented paradigm has unified the concept of module and type to create the new concept of class. The next step introduced by the POP model is to unify the concept of class with the concept of task (or process). This is realized by adding to traditional “sequential” classes a new type of classes: *the parallel class*. By instantiating parallel classes we are able to create a new category of objects we call *parallel objects*. Parallel objects are objects that can be remotely executed. They coexist and cooperate with traditional sequential objects during the application execution.

POP-C++ is a comprehensive object-oriented framework implementing the POP model as a minimal extension of the C++ programming language. It consists of a programming suite (language, compiler) and a run-time providing the necessary services to run POP-C++ applications.

In the POP-C++ execution model, when a new parallel object is created, the node which required the creation of the parallel object contacts the POP middleware running locally to ask for a new computing resource for this parallel object. To find this new computing resource, the POP middleware launches the resource discovery service available in the POP-C++ middleware. This service will contact all the neighbors

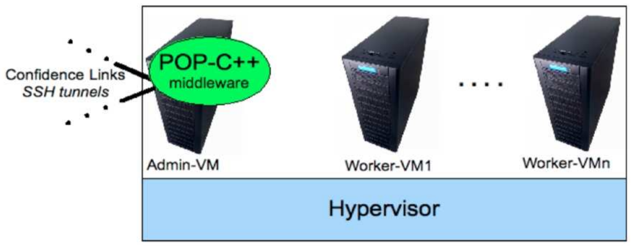


Figure 2. Architecture of a node in the ViSaG model implementation

of the node thanks to its confidence links, to ask for computing resources. Then the request is propagated through the network by following confidence links. When the request reaches a node which is able to provide the requested computing resource, it answers the originator of the request by following back the confidence links. The originator of the request chooses, between all the positive answers it received, the resource it wants to use to create the parallel object and remotely launch the execution of the parallel object inside a virtual machine. In order to be able to use the procedure presented above with the POP-C++ runtime, we had to design a dedicated architecture for the nodes of the Grid. This architecture is presented in the next section.

V. IMPLEMENTATION

A. Node architecture

In the presented implementation, a node is a computer running a virtualization platform, or hypervisor. On this platform, two or more virtual machines are deployed. The first virtual machine, called the *administrator* virtual machine (or in short: *Admin-VM*) is used to run the POP-C++ runtime. The other virtual machines are the *worker* virtual machines (or in short: *Worker-VM*). They are used by the Admin-VM to run parallel objects. The Admin-VM is connected to its direct neighbors in the Grid by the confidence links. The latter are implemented using SSH tunnels. Figure 2 illustrates this architecture.

One of the first questions we have to answer is how many Worker-VMs do we launch on a specific node. In other words, do we launch all parallel objects in the same Worker-VM, or do we launch one Worker-VM for each parallel object allocated to this node? In order to guarantee isolation between applications (see sub-section II.E) we decided to allocate Worker-VM on an application basis: for a given node, parallel objects belonging to the same application (the same POP-C++ program running instance) are executed in the same Worker-VM. This choice implies that we are able to identify applications. For this purpose, we have to generate a unique application identifier, called *AppUID*, for each POP-C++ program launched in the Grid. As we are in a fully decentralized environment, to guarantee unicity of the identifier we have based it on the IP address of the node where the program is launched, the Unix process ID as well as on the clock. This AppUID is added to all requests to allow identifying parallel objects belonging to the same POP-C++ program.

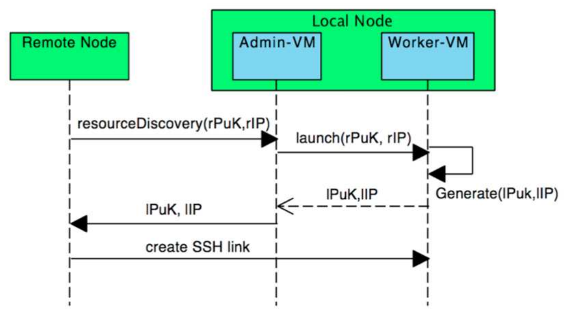


Figure 3. Creation of a parallel object.

When the Admin-VM launches a Worker-VM to provide a computing resource for the execution of a parallel object, it must ensure that the node that originates the request is able to safely contact this Worker-VM, i.e., is able to establish an SSH tunnel. This is realized through a key exchange process which is detailed below.

B. *Key exchange process*

There are two main situations where the POP-C++ middleware needs to exchange keys between virtual machines. The first one is, as mentioned above, when a new Worker-VM is launched, and the second is when the reference of a parallel object is sent to another parallel object. Indeed, as POP-C++ is based on the C++ programming language, it is possible to pass the reference of a parallel object as parameter of a method of another parallel object. As a consequence, these two parallel objects, possibly running on different nodes, must be able to communicate.

Let us first consider the situation where a new Worker-VM is launched by the Admin-VM. This operation is the consequence of a resource discovery request sent by a node that asked for the creation of a new parallel object. This request contains, among other information, the public key (rPuK) and the IP address (rIP) of the node that sent the request. The Admin-VM launches the Worker-VM and passes it the rPuK and the rIP address. The newly launched Worker-VM generates a new pair of private/public keys and sends its lPuK and lIP to the originator of the request along the confidence links. At this stage, both the originator of the request and the newly launched Worker-VM, have both PuKs and therefore are able to establish an SSH tunnel between them. This keys exchange process is illustrated on Figure 3.

The second situation we have to consider is when a parallel object running on the virtual machine VMa sends the reference of a parallel object running on a virtual machine VMb to a third parallel object running on a virtual machine VMc. This situation is illustrated on Figure 4. In such a situation, the POP-C++ middleware must ensure that VMb and VMc can establish an SSH tunnel.

When this situation occurs, we necessarily have an object running on VMa calling a method of an object running on VMc. Thus, the first thing to do is to add the PuK and the IP address of VMb in the message sent by VMa to VMc. This does not increase the number of messages but just slightly increases the size of the message sent to execute a remote method call. Next, along the confidence links, VMc sends its PuK to VMb. Now VMb and VMa can establish an SSH tunnel.

We claim that the proposed infrastructure solves four of the issues mentioned in Section II, namely, issues A, B, C and E.

The integrity of the user’s data and the user’s calculation (issue A) as well as the integrity of the host machine (issue B) are guaranteed by the isolation the virtual machine provides between the host machines and the remotely executed process. The confidentiality of the communications (issue C) is guaranteed by the SSH tunnels. Finally, as each application is executed in a different virtual machine, we guarantee that different users using the same computing resources cannot interfere between each other (issue E).

The last issue to solve (issue D) is to be able to safely use machines belonging to different private networks in presence of firewalls. Indeed the nodes belonging to a same Grid are not necessarily in the same administrative domain and can be

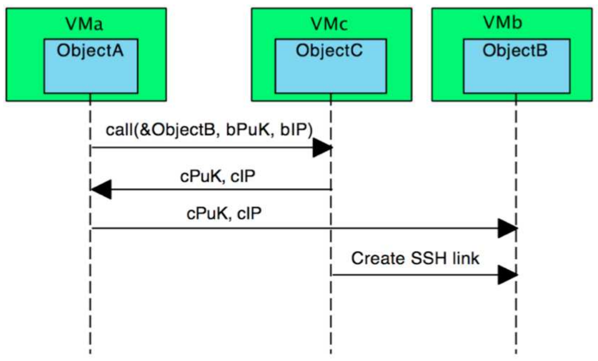


Figure 4. Passing a parallel object reference.

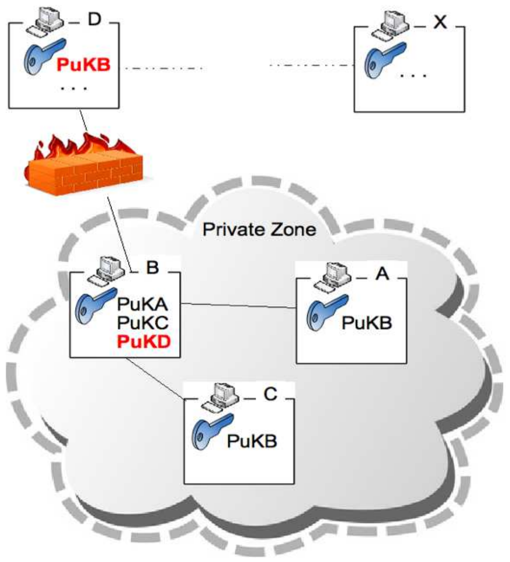


Figure 5. Example of a Grid including a private network protected by a firewall

separated by firewalls managed by different authorities. Our goal is to enable these nodes to belong to the same Grid and to be able to safely run parallel objects without opening security holes in firewalls.

Figure 5 shows a situation where three nodes (A, B and C) are in a private network separated from the rest of the Grid by a firewall. If at least one node of the Grid belonging to the private network (node B on Figure 5) creates at least one confidence link with one node located on the other side of the firewall (node D on Figure 5); then, it is possible to make this private network part of a Grid located outside of the private network. To achieve this goal we have to follow the following procedure. Suppose that a node X, located somewhere in the Grid but outside the private network, launches a request for resources. By following confidence links, this query can reach nodes located in the private network (thanks to the confidence link B-D). If a node inside the private network, let's say node A, agrees to carry out this execution, it must establish a communication with the node X. To achieve this, the Admin-VM of node A will launch a virtual machine (a Worker-VM) and will configure it with the public key of X contained in the request launched by node X. The Worker-VM creates a pair of public/private keys and transmits its public key to its Admin-VM. The latter transmits, by following the confidence links, this public key to the Worker-VM of the node X. Then, node X can establish an SSH tunnel with the Worker-VM started on node A. Now, node X and node A which are not in the same private network can safely directly communicate.

To be able to realize this communication, the firewall must be configured in the following way:

- It must allow the permanent SSH tunnel between nodes B and D.
- It must allow temporary establishment of an SSH tunnel between any node outside the local network and any Worker-VM launched by any Admin-VM inside the local network.

We claim that this configuration of a firewall is perfectly acceptable and does not create security holes if the administrator of the private network follows the following recommendations. First, the node D, which in a sense acts as a bridge toward outside the private network, must be under the control of the administrator of the private network. The Admin-VM running on this node must only run the minimal services required by the POP-C++ middleware. In our case, the SSH services with node B and the few fixed neighbors it will manually establish a confidence link with. Second, when installing the POP-C++ middleware in the private network, the administrator must take the following precaution. As the POP-C++ middleware creates and launches virtual machines, a good policy is to reserve a set of well-defined IP addresses only for this purpose and then to open, in the firewall, the SSH service only for this set of IP addresses. This will guarantee that the nodes external to the private network can only access, through SSH, Worker-VMs handled by the POP-C++ middleware. Of course, we make the assumption that the POP-C++ middleware itself is not malicious.

The middleware must ensure that when the execution of a POP-C++ program terminates, all Worker-VMs allocated to this program are deleted (or reset), to ensure that all links established during the program execution are destroyed.

VI. TESTS

To demonstrate the feasibility of the ViSaG model, we have developed a prototype integrated with the POP-C++ middleware. This prototype uses the VMware ESXi hypervisor [8] to manage virtual machines.

The virtual machine management layer can start, stop, revert, and clone virtual machines. It also allows to exchange SSH PKI and to get the IP address of a virtual machine. All these operations are performed thanks to the *libvirt* library [9] and the proprietary *VMWare VIX API*. As much as possible, we used libvirt to be compatible with different hypervisor virtualization platforms. Unfortunately, not all desired features were available, so we had to partially rely on the proprietary *VIX API* for a few key features such as cloning virtual machines and information gathering.

The SSH tunneling management is independent of any API because it uses the installed version of SSH to initiate and manage SSH tunnel. In our infrastructure, the installed version was OpenSSH running on Ubuntu 10.04 operating system.

To test our model and our prototype, we have deployed a Grid on two different sites. The first site was the "Ecole d'ingénieurs et d'architectes" in the city of Fribourg in Switzerland and the second was "Haute école du paysage, d'ingénierie et d'architecture" in the city of Geneva in Switzerland. These two sites were connected only by Internet and therefore, the security was a key point. More important, the two sites have totally different administrative network management, as required to make the test significant.

We have been able to run several distributed applications written with POP-C++ between the two sites in a transparent way for the users. The performance loss was acceptable; the main slowdown is due to the startup of the virtual machines.

VII. CONCLUSION AND PERSPECTIVES

This paper addressed the security issues in the context of a fully decentralized Grid infrastructure. Grid consumers, system managers, local users of a shared resource, network administrators, etc., are different actors involved in distributed computing, and as such they need security guarantees to accept taking part in a Grid infrastructure. Our solution takes advantage of virtualization as an isolation means, and on public key cryptography.

The existing POP-C++ Grid middleware is taken as the illustration of the decentralized Grid paradigm.

POP-C++ offers "parallel objects" as a programming model that essentially hides the complexity of the Grid aspects (local *vs* remote access, heterogeneous machines, resource discovery, etc.). On top of this architecture, and with no further constraint on the developer, our new implementation adds the wrapping of the parallel objects within virtual machines, as well as secure communications via SSH tunneling. Combining those two features brings a convincing answer to the security issues in decentralized Grids. Two levels of activities in the Grid are distinguished:

- Setup: to join a grid returns to configure and start a dedicated virtual machine (VM-Admin), which manages the POP-C++ services. The setup phase establishes connections to other nodes of the Grid; those confidence links ensure the connectivity of the Grid. The Admin-VM never executes user code itself, but has control over a pool of virtual machines for the user jobs. The setup is considered as a local event (it does not need to stop the Grid), and it typically involves a manual intervention of a user responsible of the Grid installation.

- Grid computing: when a POP-C++ user program is launched, the Admin-VMs communicate to distribute the jobs on the available resources; when it accepts a job, an Admin-VM wraps that job in a virtual machine (Worker-VM) that will be devoted to that running instance of the Grid program. The necessary encrypted connections with other Worker-VMs are automatically established, as our system takes care of conveying the needed public keys from node to node.

Thus launching a program on the Grid causes the start of several virtual machines that will be dedicated to this computation, with the appropriate communication topology. When the distributed program terminates, no trace of its execution remain (the involved virtual machines are reset before being recycled).

Our prototype has been implemented with ESXi virtual machines, but the code relies on libvirt, so that porting to another virtualization technology is greatly simplified.

The value of virtualization as a companion of Grid technology has been shown for many years. In a centralized Grid architecture, using virtual machines instead of physical systems can for instance greatly simplify Grid reconfiguration or load balancing. In the decentralized approach that we advocate, virtual machines are used as an isolation wrapper for pieces of distributed computing, a means to guarantee an appropriate security level.

In the course of our work, we identified several issues that need to be further investigated:

- It would be interesting to bring the current version based on ESXi on another virtualization software (hypervisor). The ideal candidate should provide the same level of isolation, but lightweight VMs management operations (start, stop, resume, revert, clone, etc.).

- A potential issue is about ensuring that the different VMs can benefit from system updates.

- Concerning the VMs installation, it would be worth to define precisely what capabilities have to be included in the OS equipment. In fact this leds to a concept of a "harmless Worker-VM", i.e., a virtual machines that somehow are restricted to compute and communicate with other harmless Worker-VMs only, and that are unable to cause any damage in the hosting environment (in particular no other network traffic).

- In our system, one hypothesis about security is that the POP-C++ installation is safe; we should study how this can be guaranteed and verified by the different Grid nodes.

ACKNOWLEDGMENT

The ViSaG project has been funded by the University of Applied Sciences Western Switzerland (HES-SO), project No.24247.

REFERENCES

[1] T. A. Nguyen and P. Kuonen, “Programming the Grid with POP-C++”, in Future Generation Computer Systems (FGCS), N.H. Elsevier, vol. 23, iss. 1, Jan. 2007, pp. 23-30.

[2] T. A. Nguyen, An object-oriented model for adaptive high-performance computing on the computational Grid. PhD Thesis no 3079, EPFL, Switzerland, 2004.

[3] R. J. Figueiredo, P. A. Dinda, and J. A. B. Fortes, “A Case for Grid Computing on Virtual Machines”, in International Conference on Distributed Computing Systems, May 2003, pp. 550-555.

[4] K. Keahey, K. Doering, and I. Foster, “From Sandbox to Playground: Dynamic Virtual Environments in the Grid”, in Proceedings of the 5th IEEE/ACM international Workshop on Grid Computing, Nov. 2004, pp. 34-42.

[5] S. Santhanam, P. Elango, A. Arpaci-Dusseau, and M. Livny, “Deploying Virtual Machines as Sandboxes for the Grid”, in Conference on Real, Large Distributed Systems - Volume 2, Dec. 2005, San Francisco, CA, pp. 7-12.

[6] M. Smith, M. Schmidt, N. Fallenbeck, T. Dörnemann, C. Schridde and B. Freisleben, “Secure on-demand grid computing” in Future Generation Computer Systems, Volume 25 Issue 3, March, 2009, Pages 315-325.

[7] M. Smith et al., “Secure on-demand grid computing”, in Future Generation Computer Systems archive, vol. 25, no. 3 March 2009, pp. 315-325.

[8] http://www.vmware.com [retrieved: March, 2014].

[9] http://libvirt.org/ [retrieved: March, 2014].

[10] http://en.wikipedia.org/wiki/Secure_Shell [retrieved: March, 2014].

[11] Michael Miller, “Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online”, Que Publishing Company 2008, ISBN:0789738031

[12] Ian Foster, Carl Kesselman, “The Grid 2: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA 2003, ISBN:1558609334

Challenges and Issues Within Cloud Computing Technology

Omar Ali
School of Management and Enterprise
Faculty of Business, Education, Law and Arts
University of Southern Queensland
Toowoomba-Australia
Omar.Ali@usq.edu.au

Jeffrey Soar
School of Management and Enterprise
Faculty of Business, Education, Law and Arts
University of Southern Queensland
Toowoomba-Australia
Jeffrey.Soar@usq.edu.au

Abstract-Cloud computing refers to an emerging computing model where machines in large data centres can be used to deliver services in a scalable manner. It has become popular for corporations in need of inexpensive, large scale computing. Organizations and government agencies increasingly utilise cloud computing architectures, platforms and applications to deliver services and meet the needs of their clients. There are many challenges and issues such as privacy, security and trust that can have major impacts on the information and services supported by this technology. This paper summarises the technology background and discusses challenges and issues that can arise by the use of cloud computing in organizations and government agencies.

Keywords-privacy; security; trust; issues; cloud computing.

I. INTRODUCTION

Information Technology (IT) has been adding significant benefits to various aspects of people's life, either in terms of convenience or comfort or entertainment. One of the latest developments in the IT industry is cloud computing, also known as on-demand computing. This new technology provides higher performance at relatively low cost compared to the existing dedicated infrastructures. Moreover, it can be applied to larger scale with greater reliability. Cloud computing offers a shift in the way organizations invest in their IT resources. The new model removes the need for organization to invest a substantial sum of money for the purchase of limited IT resources that are internally managed. Instead, the organization can outsource its IT resource requirements to a cloud computing service provider and pay per use.

Cloud computing is a computing model that provides a pool of computing resources which users can access through the internet. The basic principle of cloud computing is to shift the computing from a local computer to the network. It offers the capacity to utilize a common collection of resources on request. It proves extremely attractive to cash-strapped IT departments that want to deliver better services under pressure. It can offer access to greater infrastructure resources which include network, server, storage, application, services and other components, as required, without huge investment in purchase, implementation, and maintenance. Cloud computing can be deployed either as: private cloud (where organizations develop their own applications and run their own internal infrastructure), community cloud (where the cloud infrastructure is shared by several organizations and supports a specific community), public cloud (where the cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services), or hybrid cloud (where the cloud infrastructure is an integration and consolidation of two or more clouds which are private, community, or public) [1]. Cloud computing offers many advantages for IT organizations. There are, however issues and challenges that still exist and which must be dealt with. A key concern in adopting cloud computing is data security, privacy and trust.

This paper presented the state-of-the-art of research into cloud computing technology and its characteristics. It highlighted the main important challenges, barriers and risks related to the cloud computing. Also, the paper presented some mitigation steps to overcome the challenges and issues that discussed.

II. CLOUD BACKGROUND

A. Definitions

Formal definitions have been proposed in both academia and industry; however, the one provided by U.S. National Institute of Standards and Technology (NIST) [2] appears to include key common elements widely used in the cloud computing environment:

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service-provider interaction [2].

In order to provide the users with better services, cloud computing maintains proper cloud architecture, offers various deployment strategies depending on the organization structure and provisioning location, and most importantly, takes care of the security issues over a network. The four deployment models along with their characteristics are depicted in the following flow-chart Figure 1.

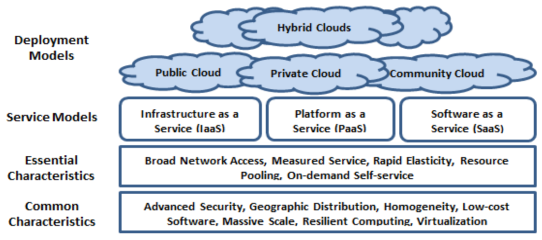

Deployment Models
Hybrid Clouds
Public Cloud
Private Cloud
Community Cloud
Service Models
Infrastructure as a Service (IaaS)
Platform as a Service (PaaS)
Software as a Service (SaaS)
Essential Characteristics
Broad Network Access, Measured Service, Rapid Elasticity, Resource Pooling, On-demand Self-service
Common Characteristics
Advanced Security, Geographic Distribution, Homogeneity, Low-cost Software, Massive Scale, Resilient Computing, Virtualization

Figure 1.The NIST Cloud Definition Framework [2]

- *On-Demand Self-Service*

A consumer with an instantaneous need in a particular timeslot can avail itself of computing resources (e.g., network storage and software use) in an automatic (self-serve) fashion without resorting to human interactions with the providers of these resources.

- *Broad Network Access*

These computing resources are delivered over the network and used by various client applications with heterogeneous platforms (e.g., mobile phones and laptops) situated at a consumer's site.

- *Resource Pooling*

In an effort to serve multiple consumers, a cloud service-provider pools together the computing resources using either the multi-tenancy or the virtualisation model “with different physical and virtual resources dynamically assigned and reassigned according to consumer demand” [2]. The motivation for setting up such a pool-based computing paradigm lies in two important factors: economies of scale and specialisation. The result of a pool-based model is that physical computing resources become ‘invisible’ to consumers, who, in general, do not have control over or knowledge about the location, formation, and origins of these resources (Database).

- *Rapid Elasticity*

For consumers, computing resources become immediate rather than persistent: there are no up-front commitments and contracts as they can use them to scale up whenever they want and release them once they finish scaling down. As there is provision for infinite computing resources on the cloud, there is no limitation in meeting their peak requirement of increased consumption any time.

- *Measured Service*

Even though the resources are pooled and shared by multiple consumers through a multi-tenancy model, the cloud infrastructure is equipped with an appropriate mechanism to monitor the usage of computing resources by individual consumers.

The European Network and Information Security Agency (ENISA) has defined cloud computing as an “on-demand service model for IT provision, often based on virtualisation and distributed computing technologies” [3]. The first academic definition of cloud computing offered by [4] is a bit different from the definition provided by the ENISA. According to Chellappa [4], cloud computing is “A computing paradigm where the boundaries of computing will be determined rationale rather than technical”. According to Buyyaa et al. [5], cloud computing is “A type of parallel and distributed system consisting of a collection of interconnected and virtualised computers that are dynamically provisioned and present as one or more unified computing resources based on the service-level agreements established through negotiation between the service provider and the customer”. While Vaquero et al. [6] proposes the following definition “Cloud is a large pool of easily usable and accessible virtualised resources” (e.g., hardware, development platforms and services). One can reconfigure these resources dynamically to allow the optimum utilisation of resources, which can be exploited by a pay per use model where the infrastructure provider offers a guarantee by means of customised Service Level Agreements (SLAs). These different definitions show the varied understanding of what cloud computing is from the perspectives of different stakeholders such as academics, architects, consumers, developers, engineers and managers.

B. *Service/Delivery Models*

The following three service models are used to categorize cloud services:

- *Software as a Service*

The SaaS service model enables consumers to use the service provider’s applications running on a cloud infrastructure [7]. Consumers can access the applications using various client devices through a thin client interface such as a Web browser (example include, Web-based email) [7]. They do not have the access to manage or control the underlying cloud infrastructure, that is, network, servers, operating systems, storage or even individual application capabilities. Consumers do have access to limited user-specific application configuration settings [2][7][38][39][40][41]. Examples of SaaS include, Salesforce, Netsuite and Google Apps.

- *Platform as a Service*

The PaaS service model enables the consumer to deploy consumer-created or acquired applications onto the cloud infrastructure with the help of programming languages and tools the provider supports [7]. As in the SaaS model, the consumer does not manage or control the underlying cloud infrastructure, but can control the deployed applications and possibly the application-hosting environment configurations [7][41][42][43]. Examples of PaaS include, Microsoft Azure service platform, Salesforce-Force.com, Google App engine Amazon relational database services and rack space cloud sites.

- *Infrastructure as a Service*

The IaaS service model provides the consumers with processing, storage, network and other fundamental computing resources. The consumer can deploy and run arbitrary software, including operating systems and applications. As with the other two models, the consumer cannot manage or control the underlying cloud infrastructure but has control over operating systems, storage and deployed applications and possibly has limited control over select networking components, such as host firewalls [2][44][45]. Examples of IaaS include, Amazon S3 (simple storage service) - EC2 (elastic cloud computing) and rack-space cloud servers.

C. *Deployment Models*

More recently, four cloud deployment models have been defined in the cloud community: [2][3][7][8][9].

- *Public Cloud*

This model enables the cloud infrastructure to be made available to the general public or to a large industry group. The cloud service providers have the full ownership of the public cloud with its own policy, value, and profit, costing, and charging model [8]. Many popular cloud services are public clouds, such as Amazon EC2, Google App Engine and Force.com. In this model, clients can choose the security level they need, and negotiate for SLA [2][8].

- *Private Cloud*

In this model, the cloud infrastructure is deployed solely for a single organization. The organization may itself manage the infrastructure or outsource it to a third party, and the cloud infrastructure may exist in the organization's premises or be based off-premise [8]. The motivation to setup a private cloud within an organization has several aspects. Firstly, in order to optimize the utilization of existing internal resources. Secondly, for security concerns including data privacy and trust which makes private cloud an option for many firms. Thirdly, data transfer cost from local IT infrastructure to a public cloud is even more considerable [10]. Fourthly, organizations always require full control over mission-critical activities that exists behind their firewalls. Lastly, academics often build private cloud for research and teaching purposes [2][7][8].

- *Hybrid Cloud*

In this model, the cloud infrastructure is composed of two or more clouds (private, community or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (example include, cloud bursting for load-balancing between clouds) [2][7]. Organizations use the hybrid cloud model in order to optimize their resources, to increase their core competencies by margining out peripheral business functions onto the cloud, while controlling core activities on premise through private cloud. The concept of the hybrid cloud aims to address the issues of standardization and cloud interoperability [7].

- *Community Cloud*

This model deploys the cloud infrastructure to several organizations at the same time and supports a specific community that shares similar concerns (example include, mission, security requirements, policies and compliance considerations). The cloud community forms into a degree of economic scalability and democratic equilibrium. The cloud infrastructure may be managed by the organizations or by a third party and may exist in the organizations' premise or be based off-premise [8]. Cloud services in government are generally utilized to reach the citizen using various tools such as Internet, Phone, IVR, etc. In particular, the citizen, can access information from various department including centre, state, and local governments such as Tax, Railway, Passport, Immigration and Visa, Central Excise, Company Affairs, National ID [8].

III. CHALLENGES AND ISSUES

A. *Cloud Privacy Issues*

One of the fundamental human rights is that of 'privacy'. From the customer point-of-view in commerce, privacy demands complete protection and the appropriate use of customers' personal information [30]. The practice of privacy in an organization includes abiding by laws, policies, standards and processes in managing the personally identifiable information (PII) of individuals.

'Privacy', in the context of the cloud, is not straight forward and depends on the type of cloud situation. The threat to privacy is minimal in some cloud application areas and services for example, services used for processing public information. However, the threat is more in services that deal with different aspects of data (e.g., collecting, transferring, processing, sharing or storing) relating to personal information. This warrants adequate measures be taken to protect privacy with regard to those services dealing with highly sensitive information, especially, information relating to location, preferences, social networks of individuals and personal health data [30] [47][48][49].

To minimize threat where the potential risk is high, customization of such services by including embedded tracking and profiling with inter-device communication can be implemented [30].

Although the public cloud is the preferred economically viable architecture, it also poses a threat to privacy because customers' data are handled and managed by the Cloud Service Provider (CSP) [30]. In this section researchers consider a number of aspects that illustrate the most important privacy issues in the public cloud: lack of user control, unauthorized secondary usage, trans-border data flow and data proliferation and dynamic provisioning [2][7].

- *Lack of User Control*

In general, cloud computing can be used as three delivery models. As mentioned before: IaaS; PaaS and SaaS [1]. When using SaaS as a delivery module, the responsibility for controlling data belongs entirely to the service-provider [30][50]. Hence, the biggest concern for the customer is: how can it retain its control on the data when information is processed or stored? With this new technology, misuse, theft or illegal use always remain matters of grave concern because a cloud computing system involves the processing of user- sensitive information. Moreover, since the technology has not been secured through a patent initially, CSP can neither impose 'no accessibility' to all PII by a third party nor can it conform to a request to delete an individual's personal data. It can be difficult to get data back from the cloud and avoid vendor lock-in [11][30][47][50]. The problem is even worse when many tenants are hosted on the same physical hardware. Thus, cloud service providers must ensure the customers that their data and applications are secured and the risks are mitigated to an acceptable level. Hence, a legal requirement that becomes important is that both the cloud provider and the customer establish information security systems and trustworthiness for each other [12][30].

- *Unauthorized Secondary Usage*

A threat can occur when information is used illegally but the standard cloud computing business model states that service providers can profit from the authorised secondary use of users' data; In particular, these data are often used to target advertisements [13][30][51].

- *Trans-Border Data Flow and Data Proliferation*

One attribute of cloud computing is data proliferation, which is a process that involves several companies and is not controlled or managed by the data owner. The vendor guarantees the ease of use by facilitating data availability in several data centres. Hence, it is very difficult for the vendor to ensure that duplication of the data or its backups is not stored or processed in a certain authority, and that all these copies of data are deleted if such a request is made [30]. Due to the dynamic nature of this technology and the movement of data, Central Processor (CP) exacerbates the trans-border data flow because it can be extremely difficult to ascertain which specific server or storage device will be used [14][30][52].

- *Dynamic Provision*

Although many of the problems faced by cloud computing are similar to those faced by traditional IT outsourcing, because of the dynamic nature of the cloud, the existing provisions for addressing the problems in more static environments are rendered obsolete or impractical to set up and implement in such a short timescale [30]. It is not clear which party is responsible (statutorily or contractually) for ensuring legal requirements for the protection of personal information, and whether appropriate data handling standards are set and followed [7][30], or if the third-party compliance can be audited effectively with such laws and standards. It is also still unclear if the cloud sub-contractors involved in processing can be properly identified, checked and ascertained as trustworthy, particularly in the dynamic environment of cloud computing. It is also unclear what rights concerning the data will be acquired by data processors and their subcontractors, and whether these are transferable to other third parties upon bankruptcy, takeover or merger [15].

B. *Cloud Security Issues*

In a traditional security model, such as a corporate firewall, for example, there is self-control over computing resources and storing and processing information within a set security perimeter. The network provides transit to other trusted end hosts, which operate in a similar manner. This model has been proven adequate for the original Internet, but not for public and hybrid clouds [30]. Since the confidential information in the cloud may be processed outside the known trusted areas as these computing environments often have unsure boundaries with regard to the location of storing and processing data, the security perimeter becomes blurred. The consumers, on the other hand, need to extend their trust to the cloud service provider, in order to obtain the service, which in turn can give rise to difficulties [30].

In addition to the privacy issues discussed previously, the public cloud has its share of security concerns. A recent user survey [16], indeed rated security as the top challenge of the cloud model. Private clouds can, to a certain extent, guarantee security, but the cost associated with this approach is quite high [50]. In this section, we present problems that are very important for cloud architectures.

The security challenges associated with cloud computing are exacerbated at the network, host and application levels [30]. The main issues relate to defining which parties are responsible for which aspects of security. Such division of responsibility is hampered by the lack of standardization of the cloud Application Program Interface (API). The risk of data loss, the unauthorized collection and usage of data and the CSP not adequately protecting data [52], are some of the security concerns facing customers. The security risks fit into a broader model of cloud-related risks. According to CSA [17], the top threats to cloud computing are abuse and nefarious use of cloud computing, insecure interfaces and APIs, malicious insiders, shared technology issues, data loss or leakage, account or service hijacking and unknown risk profile. There is no consensus on ranking the degree of severity of these risks.

It may be noted that there is scope for outsourcing security to security experts. Therefore, security need not necessarily suffer in moving to the cloud model. In fact, in many cases, greater protection than those available previously, can be obtained. Some security concerns regarding the public cloud are described below:

- *Access*

Access to confidential information by a governments' surveillance over data stored in that country in the cloud can increase the risks. Governments in the countries where the data is processed or stored may even have legal rights to view the data under some circumstances [18][53], and this may not be known by the consumers. Furthermore, as with other computing models, unauthorized access by entities involved in the provider chain with inadequate security mechanisms in place, can exacerbate the risk. There can be the risk of data theft by rogue employees of CSPs or by data thieves breaking into service providers' machines, or even by other customers of the same service if there is inadequate separation of customers' data in a machine that they share in the cloud [52].

The risk to data stored in the cloud for long periods of time is more from malicious behavior than processing in the cloud, because of higher exposure time. However, the problem can be potentially solved using encryption in the cloud storage [30].

- *Control Over Data Lifecycle*

Another important issue for the cloud is to ensure the customer that they have control over their data. In particular, the cloud should ensure that data be deleted and unrecoverable by a cloud service provider. Presently, there is no way to prove this as it relies on trust. The problem is exacerbated in the cloud because there can be many copies of the data, potentially held by different entities [30].

This risk depends more specifically on the cloud service model being used. When using IaaS or PaaS, one or more virtual machines are created in order to run a program. When the task is finished, the virtual machines and the disk spaces are released. However, it may be possible for the next user to recover the previous user's data as the media may not be wiped completely. Users generally do not know what happens to the physical volume supporting their virtual storage. When using the SaaS approach, the customer is one of the users of a multi-tenant application developed by the CSP. The customers' data is stored in the cloud and made accessible to him on his subsequent log in. The data is deleted only at the end of the lifecycle of the data, if the customer wishes to change service provider.

- *Multi-Tenancy*

It is a feature of SaaS that one program can run to multiple machines. CSP uses a multi-tenant application of the cloud to reduce cost by using a virtual machine, but it increases vulnerability [19].

- *Audit*

Internal control can be achieved through an external audit mechanism which permits CSP to monitor data [30]. The cloud computing environment presents new challenges from an audit and compliance perspective, but the existing solutions for outsourcing and audit can be leveraged. For ensuring data integrity and winning the trust of the data owner in the cloud environment, data transaction needs to be appropriate so as to prevent the occurrence of any untraceable action. This provision is still lacking, especially in public models. Additionally, there remains the issue of unclear ownership regarding the transactional data and this makes it hard to anticipate which data need protection [20].

C. *Cloud Trust Issues*

One of the major concerns, particularly with regard to financial and health data is the higher risk to data privacy and security attached to the vendor offerings which actually aim to assist business and encourage the use of cloud computing [30]. Both the financial and health sectors deal with confidential and sensitive information. Therefore, the associated vulnerability of the cloud computing system is the key business inhibitor in such sectors. These domains need control against unauthorized or secondary access or any kind of misuse, and cloud computing systems do not allow such customer control. Sectors, like finance and health, have to rely on mechanisms, such as insurance, court action, or penalties, which provide compensation in case of breach of SLAs.

There is no universally accepted scholarly definition of 'trust', as it is a complex concept; however, according to a number of contemporary cross-disciplinary scholarly writings, the following definition is widely held [21]: "Trust is a psychological state comprising the intention to accept vulnerability based upon positive expectations of the intentions or behavior of another". Yet this definition does not fully capture the dynamic and varied subtleties involved: trust is a complex notion and a multi-level analysis is important in order to try to understand it. There are many different ways in which online trust can be established: security may be one of these, although security, on its own, does not necessarily imply trust [22]. Some would argue that security is not even a component of trust: [23] argues that the level of security does not affect trust. Although some would argue that security is not even a component of trust, it has been found that sometimes increasing security enhances the level of trust among customers. One example of this might be that the assurance of cryptogenic protection of credit cards and personal data may encourage people to participate in e-commerce [24].

Some may consider reputation, another component of online trust, as the most valuable asset of any organization

[23], although the reputation of a CSP may not be justified. In the event of any breach of trust, the brand image suffers the most.

In a relationship, trust goes through different phases. In the beginning, trust is built up until it finally becomes a stable trust relationship. However, another phase may follow after achieving a stable relationship, a phase of declining trust, and trust can easily be lost as Nielsen states [25], “It is hard to build and easy to lose: a single violation of trust can destroy years of slowly accumulated credibility”.

Now, while assessing trust with regard to cloud computing systems, it is important to differentiate between social and technological aspects of providing diligent and dynamic trust as both aspects are necessary [26].

Persistent Trust: Characterized by properties or infrastructure emerging out of a relatively static socio-technological mechanism, persistent trust is established over the long term.

Dynamic Trust: In a context-based socio-technological mechanism, dynamic trust is associated with a specific state or context and specific, short term or variable information.

In a hardware or software component/system, persistent social-based trust is interlinked with technological-based trust, because the former can only be established if confidence in the operation and implementation of the system is gained [30]. A vouching mechanism underlies this connection - it is equally important to know who is vouching and what are they vouching for. Similarly, in a cloud computing system, persistent trust is established when dynamic-technological-based trust is grown in combination with both social and technological mechanisms. In a cloud resource, information can only be trusted if there is trusted vouch for the method that provides and assesses the information. Depending on the context, vouching on a method should come from different entities such as consumer groups, auditors, security experts, regulators, companies with a proven reputation, and established CSPs.

- *Lack of Customer Trust*

Lack of transparency is one of the major reasons for distrust in any system [30]. The same is true for a cloud system where individuals have no clue why, how and by whom their personal information is managed and this result in skepticism which finally leads to mistrust [27]. Customers may not show confidence in using cloud services, especially when personally identifiable information is involved, due to security related concerns as to whether the cloud can adequately protect data [7], and this is particularly related to sectors like finance and health which involve confidential and sensitive information. Before taking any decision regarding a cloud system, customers should also take into account the obligation and compliance assurance from the prospective suppliers promising to address such risks. Therefore, trust is the key element in the adoption of SaaS by the customer. A mechanism that brings transparency into different service provisions might encourage customers to adopt such a system, because when adopting the cloud, there is always trade-offs between factors like security, privacy, compliance, costs, and benefits [27].

- *Weak Trust Relationships*

Trust relationships may be weak at any point in the cloud service delivery chain, but they exist in order that a service can be provided quickly. When a cloud transaction is initiated there is always a risk of loss of control in the transaction of sensitive data to other organizations because of the globalized nature of cloud infrastructure [30]. This may cause significant loss in business due to the lack of data control on the part of the customer who is using the cloud. For example, the parent organization may not know whether the contractors are sub-contracting the key business processes to others. The contract requirements regarding data protection measures may not be propagated down the contracting chain and this further increases the risk.

Ensuring trust at all level of the chain to customer may not be transitive for cloud providers, and particularly, the customer may not trust some of the subcontractors (XaaS providers). Lack of transparency may not even allow the customer to know about the cloud providers in the chain. Particularly, models, like ‘on-demand’ and ‘pay as you go’, based on weak trust relationships, allow third parties in data security practices to, expose data and even delete data which are hard to find. Moreover, addition of new cloud service providers at short notice or in real time does not provide any chance to scrutinize their background.

Trust issues in cloud computing environments can be divided into four sub-categories [28][29][30][31], which include:

- How to define and evaluate trust according to the unique attribute of cloud computing environments?
- How to handle malicious recommend information, which is very important in cloud computing environments, as the trust relationship in the cloud is temporary and dynamic?
- How to consider and provide different security levels of service according to the trust degree?
- How to adjust and really reflect trust relationship dynamic change over time and space?

D. Cloud Interoperability Issues

Currently, each cloud offering has its own way that cloud clients/applications/users interact with the cloud, leading to the ‘hazy cloud’ phenomenon [32][57], the development of the cloud ecosystem is severely hindered because of enforced vendor lock in which prevents users from choosing alternative service providers for optimizing their resources at different levels within an organization. More importantly, proprietary cloud APIs makes it very difficult to integrate cloud services with an organization's own existing systems (e.g., an on-premise data center for highly interactive modelling applications in a pharmaceutical company) [49][57]. The scope of interoperability here refers both to the links among different clouds and the connection between a cloud and an organization's local systems. The primary goal of interoperability is to realize seamless fluid data communication across clouds and between the cloud and local applications [54][57].

There are a number of levels at which interoperability is essential for cloud computing. First, to optimize the IT asset and computing resources, an organization often needs to keep in-house IT assets and capabilities associated with their core competencies while outsourcing marginal functions and activities (e.g., the human resource system) on to the cloud. In this case, frequent communication between cloud services (Human Resources **(**HR) system) and on premise systems (e.g., an Enterprise Resource Planning (ERP) system) becomes crucial and indispensable for running a business. Poor interoperability such as proprietary APIs and overly complex or ambiguous data structures used by an HR cloud SaaS will dramatically increase the integration difficulties, putting the IT department into a difficult situation. Second, more often than not, for the purpose of optimization, an organization may need to outsource a number of marginal functions to cloud services offered by different vendors. For example, it is highly likely that a Small and Medium Enterprises (SME) may use Gmail for the email services and SalesForce.com for the HR service. This means that the many features (e.g., address book, calendar) in the email system must connect to the HR employee directory residing in the HR system [57].

- *Intermediary Layer*

The interoperability issues have been addressed in a number of recent works by providing an intermediary layer between cloud consumers and resources (e.g., VM). For example,[33][57] proposed the notion of virtual infrastructure (open nebula) management to replace native VM API interactions in order to accommodate multiple clouds, private or hybrid for an organization. Open nebula works at the virtualization level, thus providing cloud consumers with a unified view and operation interfaces towards the underlying virtualization implementations of various types. Unlike open nebula, [34][57] developed an abstraction layer at a higher level. This provides a single resource usage model, user authentication model and an API to shield the cloud providers' heterogeneity which can hinder the development of cloud-provider independent applications.

- *Standard*

Standardization appears to be a good solution to address the interoperability issue [57]. However, as cloud computing has only now started to take off, the interoperability problem has not appeared on the agenda of major industry cloud vendors. For example, neither Microsoft nor Amazon supports the Unified Cloud Interface (UCI) project proposed by the Cloud Computing Interoperability Forum (CCIF) [35][57]. The standardization process will be very difficult to progress when these big players do not come forward to reach consensus. A widely used cloud API within academia is the Eucalyptus project [36][57], which mirrors the well-known proprietary Amazon EC2 API for cloud operation. Although an Eucalyptus IaaS cloud consumer can easily connect to the EC2 cloud without substantial redevelopment, it cannot solve the general interoperability issue that requires an open API to be complied with by different types of cloud providers.

IV. MITIGATION STEPS

One observer has correctly admonished IT executives, noting that when it comes to shifting to cloud computing, “Standing pat means being left behind” [55]. Linda Cureton, NASA’s CIO, stated the matter thus:

“I’d like to say it a little more bluntly. If CIOs don’t get ready, manage fears and manage their risk, they will get run over by this disruptive technology. Your organization is doing it anyway – without you! So do something! You don’t have to move your entire enterprise into the cloud, just take the first step and look at some appropriate data sets.This does not have to be an all / none decision” [56].

Some mitigation steps and some solutions to overcome the issues discussed in the previous section are discussed here. In addition, this section also outlines a few recommendations for cloud service providers to develop good strategies which may help in reducing security, trust and privacy issues in a cloud environment. Adopting these issues again raises several issues related to performance as well as the security of the system, issues such as the user’s privilege to control data causing low transaction performance and internet speed affecting performance [37].

Some actions as listed below must take place to mitigate the above problems:

- In order to relocate a cloud environment from its traditional environment, a new policy must be put forward.
- Finding a new solution to avoid or fix a problem is not enough. One has to check the effect of the solution on the system.
- Any new changes made should be scrutinised by providers along with making the customers’ access privileges limited.
- Finding the linked service providers to a particular cloud service provider is necessary for knowing about their right to use data.
- Monitoring system should be excluded.

- Within the duration of services, a customer should be informed by its service provider about managing security policies apart from the provider's own policy.
- Data transfer should be protected and secured by standard security techniques and it must be ensured that data are managed by skilled professionals.

V. CONCLUSIONS

High security is one of the most important problems for opening up the new era of the long dreamed vision of cloud computing as utility services. As the sensitive applications and data are moved into the cloud data centres, run on virtual computing resources in the form of a virtual machine. This unique attribute however, poses many new security challenges, such as: access, control over data lifecycle, availability and back-up, multi-tenancy and audit. With the rapid improvement of cloud computing and the increasing number of cloud users, security, privacy and trust issues will continue to increase. To protect private and sensitive data that are processed in data centres, the cloud user needs to verify the following:

- The real existence of the cloud computing environment in the world.
- The security of information in the cloud.
- The trustworthiness of the systems in the cloud computing environment.

In this paper, researchers primarily aim to highlight the major issues and challenges (security, privacy and trust) on the way towards adopting cloud computing. The interoperability issue was highlighted as well.

REFERENCES

[1] D. Hilley, "Cloud computing: Taxonomy of platform and infrastructure-level offerings", *CERCS Technical Report*, Georgia Institute of Technology, 2009, accessed on September. 2013, available at: http://www.csrc.nist.gov/groups/SNS/cloudcomputing/index.html.

[2] P. Mell and T. Grance, "Draft NIST working definition of cloud computing", vol. 15, 2009, pp. 1-7.

[3] D. Catteddu and G. Hogben, "Cloud computing: Benefits, risks and recommendations for information security", *European Network and Information Security Agency* (ENISA), 2009, pp. 17-25.

[4] R. K. Chellappa, "Intermediaries in Cloud-Computing: A New Computing Paradigm", *INFORMS Annual Meeting*, Dallas, TX, October 26-29, 1997, accessed on December. 2013, available at: http://www.bus.emory.edu/ram/.

[5] R. Buyyaa, C. S. Yeoa, S. Venugopala, J. Broberg, and I. Ivona Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", *Future Generation Computer Systems*, vol. 25, 2009, pp. 599-616.

[6] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition", *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, 2009, pp. 1-6.

[7] CSA, "*Security Guidance for Critical Areas of Focus in Cloud Computing"*, 2009, Cloud Security Alliance.

[8] P. A. Dustin Amrhein, A. De Andrade, E. A. B. Armstrong, J. Bartlett, R. Bruklis, and K. Cameron, "Cloud computing use cases", *White Paper*. Version 3.0 ed., 2010, pp. 1-7.

[9] T. Grance, "The NIST Cloud Definition Framework" 2010, National Institute of Standards and Technology (NIST).

[10] M. Armbrust et al. "Above the clouds: A Berkeley view of cloud computing", *EECS Department, University of California*, Berkeley, Tech. Rep. UCB/EECS, 2009.

[11] S. Pearson, "Taking account of privacy when designing cloud computing services", *ICSE Workshop on Software Engineering Challenges of Cloud Computing*, 2009, p. 44.

[12] Z. Gansen, R. Chunming, L. Jin, Z. Feng, and T. Yong, "Trusted data sharing over untrusted cloud storage providers", *Second International Conference on Cloud Computing Technology and Science*, 2010, p. 97.

[13] P. Kresimir and H. Zeljko, "Cloud computing security issues and challenges", *MIPRO*, 2010, Opatija, Croatia.

[14] K. Popovic and Z. Hocenski, "Cloud computing security issues and challenge", *Proceedings of the 33rd International Convention*, 2010, p. 344.

[15] R. Gellman, "Privacy in the Clouds: Risks to Privacy and Confidentiality from Cloud Computing", *World Privacy Forum*, 2009, accessed on June. 2013, available at: http://www.worldprivacyforum.org/pdf/WPF_Cloud_Privacy_Report.pdf.

[16] IDC, "Enterprise panel", 2009, accessed on July. 2013, available at: http://www.slideshare.net/JorFigOr/cloud-computing-2010-an-idcupdate.

[17] CSA, "Top threats to cloud computing", 2010, Cloud Security Alliance (CSA).

[18] Regulation of Investigatory Powers Act (RIPA), Part II, 2000, UK.

[19] M. Jensen, J. Schwenk, N. Gruschka and L. L. Iacono, "Technical security issues in cloud computing", *IEEE International Conference on Cloud Computing*, 2010, p. 109.

[20] J. A. Hall and S. L. Liedtka, "The sarbanes-oxley act: Implications for large-scale IT outsourcing", *Communications of the ACM*, vol. 50, no. 3, 2007, pp. 95-100.

[21] D. Rousseau, S. Sitkin, R. Burt, and C. Camerer, C. "Not so different after all: A cross-discipline view of trust", *Academy of Management Review*, vol. 23, no. 3, 1998, pp. 393-404.

[22] D. Osterwalder, "Trust through evaluation and certification?", *Social Science Computer Review*, vol. 19, no. 1, 2001, pp. 32-46.

[23] H. Nissenbaum, "Can trust be secured online? A theoretical perspective", *Etica e-Political*, no. 2, 1999.

[24] S. Giff, "*The Influence of Metaphor, Smart Cards and Interface Dialogue on Trust in e-Commerce"*, 2000, University College London.

[25] J. Nielsen, "Trust or bust: Communicating trustworthiness in web design", *Jacob Nielsen's Alert Box*, 1999, accessed on July. 2013, available at: http://www.useit.com/alertbox/990307.html.

[26] S. Pearson, M. Casassa Mont, and S. Crane, "Persistent and dynamic trust: Analysis and the related impact of trusted platforms", *Trust Management*, LNCS 3477, ed: P. Herrmann, V. Issarny, and S. Shiu, 2005, pp. 355-363.

[27] A. Tweney and S. Crane, "Trust guide 2: An exploration of privacy preferences in an online world", *Expanding the Knowledge Economy*, 2007, IOS Press.

[28] S. Paquette, P. T. Jaeger, and S. C. Wilson, "Identifying the security risks associated with governmental use of cloud computing", *Government Information Quarterly*, vol. 27, no 3, 2010, pp. 245-253.

[29] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing", *Journal of Network and Computer Applications*, vol. 34, no. 1, 2010, pp. 1-11.

[30] S. Pearson and A. Benameur, “Privacy, security and trust issues arising from cloud computing”, *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science*, IEEE Press, Nov. 2010, pp. 693-702.

[31] A. Sangroya, S. Kumar, J. Dhok, and V. Varma, “Towards analyzing data security risks in cloud computing environments”, *Communications in Computer and Information Science*, vol. 54, 2010, pp. 255-265.

[32] M. Nelson, “Building an open cloud”, *Science*, vol. 324, 2009, p. 1656.

[33] B. Sotomayor, R. Montero, I. Lorente, and I. Foster, “Virtual infrastructure management in private and hybrid clouds”, *IEEE Internet Computing*, vol. 13, 2009, pp. 14-22.

[34] T. Harmer, P. Wright, C. Cunningham, and R. Perrott, “Provider-independent use of the cloud”, *The 15th International European Conference on Parallel and Distributed Computing*, 2009, p. 465.

[35] Unified Cloud Interface (UCI) 2010, accessed on August. 2013, available at: http://code.google.com/p/unifiedcloud/.

[36] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “The eucalyptus open-source cloud computing system”, *Proceedings of Cloud Computing and Its Applications, 2008.*

[37] P. Mathur and N. Nishchal, “Cloud computing: New challenge to the entire computer industry”, *The1st International Conference on Parallel, Distributed and Grid Computing (PDGC*), 2010, p. 223.

[38] L. Wang, J. Tao, and M. Kunze, “Scientific cloud computing: Early definition and experience”, *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications*, 2008, Dalian, China.

[39] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, “Cloud computing: A perspective study”, *New generation Computing*, vol. 28, 2010, pp. 137-146.

[40] E. Clemons and Y. Chen, “Making the decision to contract for cloud services: Managing the risk of an extreme form of it outsourcing”, *Proceedings of 44th Hawaii International Conference on System Sciences*, Hawaii, 2011, pp. 1-10.

[41] A. Velte, T. Velte, and R. Elsenpeter, R. “Cloud Computing a Practical Approach’, McGraw-Hill, USA, 2010.

[42] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “Cloud computing and Grid computing 360-degree compared”, *Proceedings of Grid Computing Environments Workshop*, Austin, TX, 2008.

[43] T. Dillion, C. Wu, and E. Chang, “Cloud computing: Issues and challenges”, *Proceedings of 24th International Conference on Information Networking and Applications*, 2010, Perth, WA.

[44] Y. Sohan and H. Zeng, “Cloud: A computing infrastructure on demand”, *Proceedings of 2nd International Conference on Computer Engineering and Technology*, 2010, Chengdu, China.

[45] S. Bhardwaj, L. Jain, and S. Jain, “Cloud computing: A study of infrastructure as a service (IaaS)”, *International Journal of Engineering and Information Technology*, vol. 2, no. 1, 2010, pp. 60-63.

[46] J. Hurwitz, R. Bloor, M. Kaufman, and F. Halper, “Cloud computing for dummies”, *Wiley Publishing*, 2010, Indianapolis, Indiana.

[47] S. Alshomrani and S. Qamar, “Cloud based e-government: benefits and challenges”, International Journal of Multidisciplinary Sciences and Engineering, vol. 4, no. 6, 2013, pp. 1-7.

[48] F. A. Alvi, B. S. Choudary, N. Jaffery, and E. A. Pathan, “Review on cloud computing security issues and challenges”, Department of Electronic Engineering, QUEST Nawabshah, Sindh, Pakistan, 2011.

[49] N. Yadav and V. B. Singh, “E-governance: past, present and future in India”, *International Journal of Computer Applications*, vol. 53, no. 7, 2012, pp.36-48.

[50] D. Karunanithi and B. Kiruthika, “Efficient framework for ensuring the effectiveness of information security in cloud computing, *International Conference on Signal, Image Processing and Applications*, workshop of ICEEA, 2011.

[51] D. Catteddu and G. Hogben, “Cloud computing: Benefits, risks and recommendations for information security’, *European Network and Information Security Agency* (ENISA), 2009.

[52] B. Zwattendorfer, K. Stranacher, A. Tauber, and P. Reichstädter, “Electronic government and the information system perspective”, *International Conference on Network and System Security*, 2011, pp.1-6.

[53] T. Hariguna, “Prototype cloud computing for e-government in Indonesia”, *International Journal of Engineering & Technology*, vol. 11, no. 6, 2011.

[54] K. Irion, “Government cloud computing and the policies of data Sovereignty”, *22nd European Regional Conference of the International Telecommunications Society*, Budapest, Economic and Policy Issues, 2011.

[55] H. M. Nezhad, B. Stephenson, and S. Singhal, “Outsourcing business to cloud computing services: Opportunities and challenges”, *IEEE Internet Computing, Special Issues on Cloud Computing*, 2009, pp. 1-10.

[56] M. O’Gara, “Washington itching to take the lead on cloud computing”, 2009, SOA, accessed on December. 2013, available at: http://govit.sys-con.com/node/1055764.

[57] T. Dillon, C. Wu, and E. Chang (2010). Cloud Computing: Issues and Challenges’, 24th IEEE International Conference on Advanced Information Networking and Applications, 2010, pp. 1-7.

Good Performance Metrics for Cloud Service Brokers

John O'Loughlin and Lee Gillam
Department of Computer Science
University of Surrey
Guildford, England
{John.oloughlin,l.gillam}@surrey.ac.uk

Abstract—**The large number of Cloud Infrastructure Providers offering virtual machines in a variety sizes, but without transparent performance descriptions, makes price/performance comparisons difficult. This presents opportunities for Cloud Service Brokers (CSBs). A CSB could offer transparent price/performance comparisons, or performance discovery. This paper explores the kinds of performance measures such CSBs should use. Using three quite different benchmarks, povray, bzip2 and STREAM, we show a 17%, 77% and a 340% increase in performance, respectively, from worst to best in an existing large Cloud. Based on these results, we propose a discovery service for best price/performance for use in aggregation of heterogeneous resources.**

Keywords- Cloud Computing; performance;brokers;metrics

I. INTRODUCTION

In Infrastructure Clouds, users can obtain a wide range of compute resources, predominant amongst which are Virtual Machines (VMs). There are a large (and increasing) number of Cloud providers, all selling a variety of VM types. These VM types are typically defined in terms of amounts of resources provided: number of vCPUs, amount of RAM and an amount of local storage. The lack of transparent performance descriptions for instance types, and the large performance variation of instances of the same type that we [1] and others [14] have previously reported, makes finding the best price/performance a daunting task. It is, therefore, not surprising that Cloud Service Brokers (CSBs), which can act as intermediaries between Cloud users and Cloud providers, are gaining attention. There is a clear opportunity for such CSBs to address the price/performance issue.

For a CSB to address price/performance, they must first clarify how performance is defined, measured and, especially, useful to Cloud users. Provider-specific performance measures, such as Amazon's EC2 Compute Unit (ECU), are of limited value to a user since they do not always correlate well to the performance of the application they wish to run [1].

In this paper, we explore the kinds of performance measures that could be useful to users, and therefore beneficial to profit-seeking CSBs. CSBs will need to know the deliverable performance of various Cloud resources with respect to the applications user wish to run. However, large Clouds are inevitably heterogeneous and resource availability may be unpredictable. And so it is vital to find strategies for obtaining the best resources when it is not possibly simply to request the best.

The rest of this paper is structured as follows: In section II, we explore the types of services a CSB could offer and show how they can help users with the daunting task of finding comparable instances across providers. In section III, we explore performance measurement and identify metrics that are appropriate for Cloud users. In section IV, we explain and present the performance results obtained from approximately 300 m1.small instances on EC2. We show how performance is dependent on both the CPU model backing the instance **and** the application. We use these results to propose a performance discovery service, initially for Amazon EC2; which we explore in detail in section V. In section VI, we present conclusions and future work.

II. OPPORTUNITIES FOR CSBS

Resources in Infrastructure Clouds should be available on demand and with the ability to obtain or release more and/or bigger and smaller VMs in order to scale use as required. Additionally, the user should not have to worry about the provider's ability to meet this demand. Such capability, together with a 'pay for what you use' charging model, makes Infrastructure Clouds particularly attractive for handling workload spikes or ad-hoc computing tasks. Presently, if they hope to obtain best performance for price, each Cloud user needs to understand both the resource requirements of their application and the capabilities and pricing structures offered by each Infrastructure provider. Clearly there are costs and risks associated to such determination, and likely much repetition of effort across users.

At minimum, a CSB could reduce the repetition and associated costs of determining best performance for price by having transparent information about likely performance. Gartner [2] identifies three kinds of service that brokers can provide: ***intermediation***, ***aggregation*** and ***arbitrage***, and each of these kinds of service could benefit from such determination.

An ***intermediation*** service enhances existing services to add value to them. A potential performance service exists simply in being able to find instances whose performance is within a particular range.

In financial markets, ***arbitrage*** refers to the practice of simultaneously buying and selling the same resources in different markets and exploiting the price difference. CSBs could exploit performance variation by reselling better performing instances at a higher price and worse performing instances at a lower price.

CSBs wishing to ***aggregate*** resources across Clouds to find truly optimal performance would need first to understand the deliverable performance of each them, and then to be able to exploit the variations across them.

Such opportunities readily exist: simply finding comparable instance types between providers is not always straightforward. For example, both Amazon's Elastic Compute Cloud (EC2) First Generation Standard Instances [3] and Microsoft's Azure Standard Instances [4] start with a very similar 'small' instance type: 1 vCPU with 1.7GB and 1.75GB of RAM respectively. After this they diverge, with the EC2 extra large instance type having 4vCPU and 15GB RAM whilst the Azure extra large has 8vCPU with 14GB RAM.

Comparing expected performance is not always possible either. EC2 offers a compute rating for their instance types, in the form of the EC2 Compute Unit (ECU), but Microsoft does not do so for Azure. Therefore, price/performance comparison between the respective small instance types requires the customer to conduct benchmarking experiments of both types. Comparing a larger range of instances types, across multiple providers, will quickly become expensive, and quite possibly prohibitively so, for all except the largest users. In addition, as we show later in this paper, it would be a mistake to consider just a few instances in even a single provider as necessarily representative.

For EC2, Amazon define the ECU in terms of equivalent performance to a reference machine, '…an early 2006 Xeon' [5]. How the equivalence is established is not explained. Following EC2 we find both the Google Compute Engine Unit (GCEU) [6] and the HP Cloud Compute Unit (HPCCU) [7] defined in terms of reference machines. It is unclear how these metrics relate to established performance metrics; such as program execution time. Clearly there are multiple opportunities for the CSB in both understanding and providing price/performance information for instance types in terms of performance metrics that are useful to customers. And in the next section we explore what those metrics might be.

III. COMPUTE PERFORMANCE METRICS AND MEASUREMENTS

The question of how compute performance should be defined and measured is surprisingly contentious. There is no commonly accepted unit of work on a computer – and therefore no accepted definition of either how fast a computer is or how much work has been done per unit of time. Some performance metrics involve physical characteristics of the machine, such as CPU clock rates or Theoretical Peak Performance (TPP). Such approaches tend to have common failings: (1) they are only valid when comparing machines of the same micro-architecture (2) they tend to correlate poorly to actual application performance.

Defining machine performance in terms of application performance also leads to some difficulties: should we use actual applications (that are in common use) or applications which are, in some sense, typical of a class of applications? Should we use one application or a suite of applications? If we use a suite of applications how do we best summarise them? We do not discuss these important questions in further detail here but we do note that the trend is for actual applications and not kernels or micro applications.

There are, however, certain characteristics a good performance metric should have [8]. Four of the more important characteristics are:

- A metric is *linear* if, when its value changes by a given ratio, the actual performance (as measured by application performance) of the machine changes by the same ratio. For example, the Amazon ECU to be linear we might expect a 2 ECU machine to run a CPU bound application in half the time of a 1 ECU machine.
- A metric is *reliable* if, whenever it indicates that machine A should outperform machine B, it does.
- A metric has *repeatability* if the same value (within an error bound) is obtained every time we take a measurement.
- Finally, a good metric should be *easy to measure*.

Our previous work [1] has shown that the ECU may be *reliable* but is neither *linear* nor *repeatable*, unless a large error bound is considered. It is also not easy (or indeed possible) to measure since it is defined in terms of equivalent performance to a reference machine without defining how the equivalence is established or what approaches are used to construct it.

A. *'Bad Metrics'*

The following commonly found metrics all fail on at least one of the above characteristics: CPU clock rate, Theoretical Peak Performance, the maximum number of instructions a CPU could in theory execute per second, (TPP), Millions of Instructions per Second (MIPS), BogoMIPS and Floating Point Operations per Second (FLOPS). We discuss these metrics further below:

Clock Rate: A number of providers [12] express expected performance of their instances in terms of a clock rate but do not specify the CPU model. Clock rate is generally not a reliable indicator of application performance. The Pentium 4 range, for example, had higher clock rates than the Pentium 3 models but without a corresponding increase in application performance due to a significant increase in the depth of the CPU pipeline.

TPP: TPP for a multi-core CPU is calculated as the number of cores multiplied by the number of execution units multiplied by the clock rate. It serves only as an upper bound on performance, and assumes that the CPU pipeline is full at all times. However, due to pipeline stalls, branch mis-predictions and memory hierarchy latency, application performance may well differ significantly from the one predicted by the TPP. This is of course an issue with peak metrics in general – they are often unobtainable.

MIPS: MIPS is calculated as the instruction count for an application divided by execution time*10^6. A MIPS rating cannot be used to compare the performance of CPUs with different instruction sets, since the same program may compile to a different total number of instructions. For example, we cannot readily compare a RISC machine to a CISC machine. MIPS, along with FLOPS, suffers from being a throughput (or rate) metric and yet the unit of work being done (execute an instruction or floating point instruction) is not constant. For example, memory access instructions will take longer to execute than instructions which operate on data present in CPU registers. Whilst MIPS are not commonly used by providers, they are the default performance measure available in the well-known Cloud simulation toolkit Cloudsim.

BogoMIPS: BogoMIPS stands for Bogus MIPS and is defined by the number of NOOP (no operation) operations a machine performs per second. It is used in the early stages of Linux Kernel boot process as a calibration tool and was not intended as a performance metric. In spite of this, some have claimed [10] that a machine's BogoMIPS can be related to its performance - without offering any supporting evidence for such a claim.

FLOPS: Similar to MIPS, FLOPS uses an inconsistent unit of work – the FLOP. Different FLOPS may take different amounts of time to execute depending on what they do. However, peak GigaFLOPS (GFLOPS), as measured by High Performance Linpack, is still the measure used to rank systems in the well-known top 500 HPC list. Some supercomputing centres are now moving away from peak performance to sustained performance of applications their users will run. It should also be noted that expressing performance in terms of FLOPS will not inform a user of a system how well an application that contains no floating point operations will run, and so the measurement is domain specific.

B. *'Good Metrics'*

The metrics above are generally defined in terms of machine characteristics; better performance metrics tend to be defined in terms of application performance. We discuss some of these below.

Program Execution Time: This metric is defined by the elapsed wall clock time from program start to finish. Some authors [9] consider this to be the only meaningful and informative metric and suggest that any other metric may be misleading. *Performance* is defined as the inverse of execution time and so faster execution times give higher performance scores.

Throughput: Throughput (CPU bandwidth) is defined as the number of units of work per unit time (usually per second) the CPU can perform. For consistency, the unit of work should be well defined and remain constant. As discussed there is no commonly accepted definition of unit of work, and so this becomes workload dependent.

Work done in a fixed time: In the program execution time metric, the amount of work done is fixed and wall clock time is the variable of interest. In [11], the authors argue that some systems, such as HPC, are purchased in order to allow more work to be done in the same amount of time when compared to older systems. By 'more work' they tend to mean solving a larger problem, not just an increase in throughput. This could be, for example, running a Monte Carlo Simulation at a much greater number of iterations to produce smaller error bounds on estimates.

Response Time: The above metrics are suitable for batch jobs. For interactive applications or websites, response time (also known as application latency) is a good metric. It has been shown [16] that higher response times lead to lower user satisfaction.

In general, the good metrics relate to application performance and not machine characteristics. Given this, ratings that relate equivalent performance to specified physical machines, as currently favoured by large Cloud providers, are unsatisfactory for most purposes. And, as we will show, are also not particularly meaningful even for comparing virtual machines in the same Cloud (provider). Cloud Service Brokerages, then, would add good value by selecting good performance metrics that can clearly relate to the applications that their customers wish to run.

IV. EXPERIMENTS ON EC2

In this section we address a question that we believe will be of interest to a typical Cloud user: Given a number of workloads, where can I obtain best performance for them? Here, we explore this question in one Region of Amazon's EC2 (US-East), which reveals several insights into performance variability.

A. *Experimental Setup and Results*

We consider the following 3 workloads:

1. A bzip2 compression on an Ubuntu 10.04 desktop ISO.
2. A povray ray trace on the benchmark.pov file.
3. STREAM memory bandwidth benchmark using the triad kernel.

Both bzip2 (albeit with different input files) and povray are part of the Standard Performance Evaluation Corporation (SPEC) CPU benchmark suite [15]. They measure different aspects of the CPU: bzip2 primarily uses integer arithmetic whilst povray makes heavy use of floating point operations.

We run into an immediate and interesting difficulty. The EC2 account we are using has access to just 4 of a possible 5 Availability Zones (AZs) [13] in US East: us-east-1b, us-east-1c, us-east-1d and us-east-1e. As we shall see, EC2 AZs have different performance characteristics from each other. It is therefore entirely possible that the AZ this account does not have access to, us-east-1a, provides better performance.

We ran approximately 300 m1.small instances. In each instance the workloads were run sequentially. For bzip2 and povray we recorded the **Programme Execution Time,** whilst STREAM reports memory bandwidth in MB/s. As the unit of work is *consistent*, STREAM is an example of a good throughput metric. In Table 1 below, we record the summary statistics (to the nearest second or MB/s):

TABLE I. SUMMARY STATISTICS

Workload	Mean	Max	Min	SD	CoV
Bzip2 (s)	528	745	421	78	0.15
Povray (s)	636	701	599	33	0.05
STREAM (MB/s)	2853	5860	1328	1239	0.43

We determine the CPU model backing an instance by examining the file /proc/cpuinfo. All of these instances were backed by one of the following Intel Xeon CPU models: E5430, E5-2650, E5645 and E5507. The CPU models found are the same as in our previous work. In Table II below, we present the statistics for each of the workloads broken down by CPU model.

TABLE II. SUMMARY STATISTICS BY CPU MODEL

Workload	Statistic	E5430	E5-2650	E5645	E5507
Bzip2	Mean(s)	439	468	507	621
	Max(s)	467	500	535	745
	Min(s)	421	451	490	567
	SD(s)	11	12	10	31
	CoV	0.025	0.026	0.02	0.05
Povray	Mean(s)	693	614	606	632
	Max(s)	701	624	628	650
	Min(s)	687	606	599	625
	SD(s)	3	5	7	5
	CoV	0.004	0.008	0.011	0.008
STREAM	Mean(MB/s)	1446	5294	3395	2348
	Max(MB/s)	1572	5860	4008	2448
	Min(MB/s)	1328	4935	2995	2078
	SD(MB/s)	66	191	287	104
	CoV	0.045	0.036	0.085	0.044

The coefficient of variation (CoV) is the ratio of the standard deviation relative to the mean and is useful for comparing the amount of variation between two data sets. The CoV here shows that the amount of variation for each workload is greater when considered across all CPU models than for a particular CPU model. For example, across all CPU models the CoV for the bzip2 workload is 0.15 whilst for the individual CPUs the largest CoV we find is 0.05, and the smallest CoV is 0.02, as found on the E5645. We interpret this as follows: There is approximately 8 times the amount of variation in bzip2 results considered across all CPU models than we find on the E5645. We also note that the E5507 has twice the variation as found on the other models. We have similar findings for the povray and STREAM workloads.

From the results we see that performance for all workloads depends on the CPU model backing the instance. As such, we can order the CPUs by how they perform the task. For bzip2 we have (from best to worst): E5430, E5-2650, E5645 and E5507. Interestingly, the orderings for both povray and STREAM are different, for example, for STREAM the ordering would be: E5-2650, E5645, E5507 and E5430. This shows that it is not possible to identify a 'best' CPU for all workloads, providing an opportunity for a CSB to identify which CPUs models provide best performance for specific applications.

These results suggest that the E5-2650 is the most versatile CPU, for these three workloads - it is the second best performing CPU model for both the bzip2 and povray tests and the best for STREAM. In Fig.1, Fig.2, and Fig.3 below we present histograms of the results, broken down by CPU model, which show this more clearly.

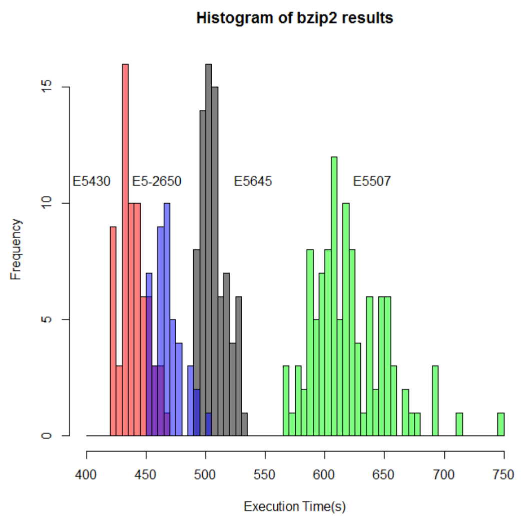


Figure 1. Bizp2 Execution Time(s)

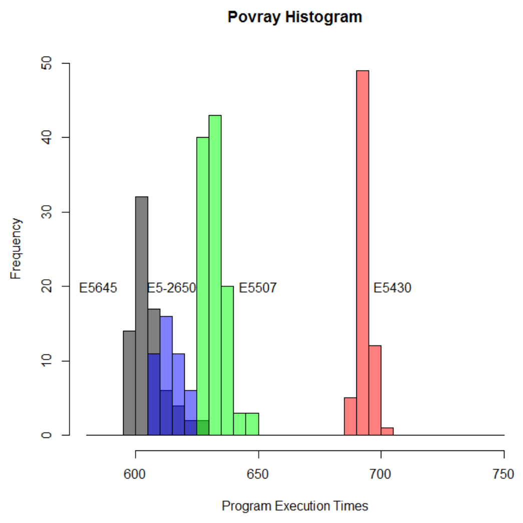


Figure 2. Povray Execution Time(s)

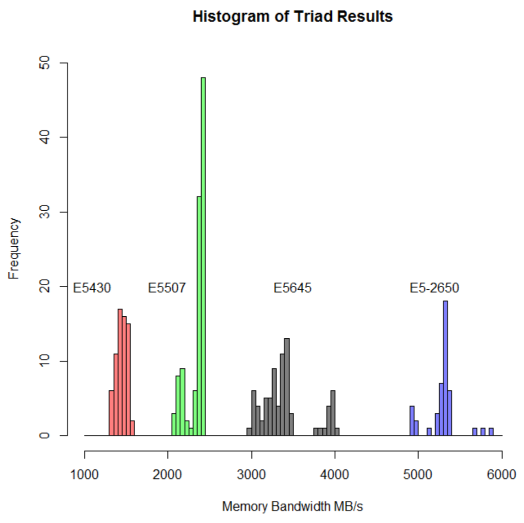


Figure 3. Triad Memory Bandwidth

We can use these results to identify which CPU model is most suitable for a particular workload. The question we now consider is this: How would I obtain instances with this CPU model backing it when I can't specify it in the request? In Table III below, we record the percentages of the CPU models we found in the 4 AZs (note that, due to Amazon's structuring within EC2, different users may well have different mappings):

TABLE III. CPU MODEL DISTRIBUTION BY AZ

	E5430	E5-2650	E5645	E5507
us-east-1b	25%	0	40%	35%
us-east-1c	27%	0	23%	50%
us-east-1d	30%	0	36%	34%
us-east-1e	0%	88%	12%	0%

Considering STREAM, we know that the E5-2650 is the best performing (from Table II), and from Table III, we see that we should choose us-east-1e when running this workload. For povray it is slightly more complicated. The E5645 is, on average, the best for this task but the E5-2650 gives very similar performance. Whilst it may be tempting to run instances in us-east-1b for this workload, we note a high percentage of the E5430, the worst performing CPU for the task. A safer option may still be us-east-1e.

B. Work Done and Price/Performance Considerations

Although performance information for users is useful, arguably more important for them is price/performance – more so when considering systems or workloads requiring multiple instances or multiple instance types. For the bzip2 and povray workloads we consider one completed compression or image rendering as a unit of work. For each CPU model we can calculate the average number of units of work per hour that can be performed, and from this we deduce a price per unit of work. The instance price per hour for an m1.small in the US East Region (as of 01/14) is $0.06. The cost of an instance is the number of wall clock hours elapsed since the instance was launched, with partial hours charged as full hours. For example, an instance started at 20:50 and terminated at 22:05 will be charged for **3** hours. In Table IV below, we record the partial Units of Work (UoW) per hour (which we call the completion rate) together with the price per UoW.

TABLE IV. UNITS OF WORK

Workload	E5430 ($/UoW)	E5-2650 ($/UoW)	E5645 ($/UoW)	E5507 ($/UoW)
Bzip2	8.2 0.0073	7.7 0.0078	7.1 0.0085	5.8 0.001
Povray	5.2 0.0112	5.9 0.0101	5.9 0.0101	5.7 0.0103

Table IV uses mean application execution times, and also includes partial work done. Whilst in some cases this may be useful, in general it is not clear if a user would be interested in partial completion of work. Instead, we can consider a simpler question, for example: What are the best and worse prices for completing my work using m1.small instances? One such piece of work might be 'compress 10 desktop ISO images using bzip2'.

From Table 1, we calculate the best and worst completion rate as 8.6 and 4.8 (min E5430 and max E5507). Based on EC2 wall clock hours, 10 units will see the user charged for at least 2 hours on the E5430 and at least 3 hours on E5507: best price would be $0.12 (potentially $0.18 depending on job start timing) and worst price is $0.18 (potentially $0.24). So, assuming start on the hour, we may see a 50% increase in cost for the same work. And yet, if we compare the actual execution times - 421s to 745s - we find a 77% increase. However, to complete 50 units of work requires 6 hours on the E5430 and 11 hours on the E5507, with respective costs being at least: $0.36 and $0.66, an 83% increase in the cost.

V. EC2 PERFORMANCE DISCOVERY SERVICES

Based on the foregoing, we would envisage one performance service which could be used for EC2 as follows: A user requests performance information for a workload on a range of instance types. We assume the workload would be representative of an application the user wishes to run. For example, a user may have determined on their local systems that their application requires high memory bandwidth, so STREAM workloads, as described in section IV, would be of interest. In general, these workloads could be either well known benchmarks, a dwarf kernel [17] or a self-produced effort. How well the workload and the application correlate is the responsibility of the user not the broker at this point.

The broker will then determine the performance ordering of the CPU models associated to the class. So for example, a user requesting a povray run against the standard benchmark input file on m1.small instances would have the following returned to them: E5645, E5-2650, E5507 and E5430.

In this first step we have identified which CPU models give best price/performance. Next, and based on historical data, the broker will inform the user in which AZ they are **most likely** to find the better performing CPU models, as discussed in section IV sub-section B. The historical data could also be used to provide estimates of the probability of actually obtaining a given model.

Additional services can easily be imagined, such as obtaining instances with given CPU models on behalf of the user. Further, as we have demonstrated, performance variation is greatest amongst different CPU models, and exists in instances backed by the same model. In this case the variation could be a function of resource contention, or simply variation in quality of other system parts, and so is a run time property. Finding best performing instances at run time is another potential performance service.

VI. CONCLUSIONS AND FUTURE WORK

There is an overwhelming variety of instance types on offer currently, in various Infrastructure Clouds, and a lack of transparent performance information. This make choosing instances types that offer best price/performance for given applications difficult. This would seem to be a clear opportunity for CSBs to add performance related services on top of existing Cloud offers. To be successful, we would argue that the notions of performance must be in-line with ones that are relevant to the applications that users wish to run. As discussed in section III, these are most likely to involve one or more of execution times, throughput and work done.

We have shown that performance of an instance depends on both the CPU model backing the instance and the application. And so determining best price/performance requires knowledge of both, as well as an ability to predict where the 'best' CPU models for the application can be found. Based on our work here, and on previous results, we proposed a performance discovery service. As an example, we showed that for the best memory bandwidth performance for m1.small instances (our EC2 account), make requests to us-east-1e.

In future work, we wish to explore these ideas further, and in particular to tackle the problem of performance monitoring with respect to the 'good' metrics described here. This is needed for CSBs who would need to offer Service Level Agreements (SLAs) with performance guarantees for instances obtained on behalf of users. In the same way that providers describe performance in terms of machine characteristics, we find most performance monitoring focuses on system metrics. It is unclear how system metrics relate to the *good* performance metrics as described here, and we hope to address suitable performance monitoring also.

REFERENCES

[1] J. O'Loughlin and L. Gillam, "Towards performance prediction for Public Infrastructure Clouds: an EC2 case study," Proc. IEEE Fifth International Conference on Cloud Computing Technology and Science (CloudCom 2013), Dec2013, pp. 475-480.

[2] "IT Glossary," www.gartner.com. [Online]. Available: http://www.gartner.com/it-glossary/cloud-services-brokerage-csb [Accessed: 27th January 2014].

[3] "Amazon EC2 Instance," aws.amazon.com. [Online]. Available: http://aws.amazon.com/ec2/instance-types/ [Accessed: 27th January 2014].

[4] "Virtual Machine Pricing Details," www.windowsazure.com. [Online]. Available: http://www.windowsazure.com/en-us/pricing/details/virtual-machines/ [Accessed: 27th January 2014].

[5] Amazon EC2 FAQs," aws.amazon.com. [Online]. Available: http://aws.amazon.com/ec2/faqs/#What_is_an_EC2_Compute_Unit_and_why_did_you_introduce_it [Accessed:27th January 2014].

[6] "Google Cloud Platform," cloud.google.com. [Online]. Available: https://cloud.google.com/pricing/compute-engine [Accessed:27th January 2014].

[7] "HP Cloud Pricing," www.hpcloud.com. [Online]. Available: https://www.hpcloud.com/pricing [Accessed:27th January 2014].

[8] D. Lilja, Measuring Computer Performance, New York, Cambridge University Press, 2000.

[9] J. Hennessy and D. Patterson, Computer Architecture a Quantitative Approach, 5th Ed. Waltham, Elsevier, 2012.

[10] I. Goiri, F. Julii, J. Fito, M. Macias and J. Guitart, "Supporting CPU-based guarantees in Cloud SLAs via resource level QoS metrics", Future Generation Computer Systems,Vol 28, pp. 1295-1302, 2012.

[11] Q. Snell and J. Gustafson, "A new way to measure computer performance", Proc. Hawaii International Conference on Systems Science, pp.392-401, 1995.

[12] "CloudLayer Computing", softlayer.com. [Online]. Avaialble: http://www.softlayer.com/cloudlayer/computing [Acessed: 27th January 2014].

[13] Global Infrastructure," aws.amazon.com. [Online]. Available: http://aws.amazon.com/about-aws/globalinfrastructure/ [Accessed: 27th January 2014].

[14] M Armbrust et al, "Above the clouds: a Berkely view of cloud computing". Technical Report EECS-2008-28, EECS Department, University of California, Berkeley.

[15] "SPEC CPU2006," www.spec.org. [Online]. Available: http://www.spec.org/cpu2006/ [Accessed: 27th January 2014].

[16] J. Hoxmeier and C. DiCesare, "System response time and and user satisfatcion: an experimental study of browser based applications", Proc. Of the Association of Information Systems Americas Conference, Long Beach California, pp.140-145, 2000.

[17] Dwarf-Mine," view.eecs.berkeley.edu/wiki. [Online]. Available: http://view.eecs.berkeley.edu/wiki/Dwarfs [Accessed: 27th January 2014]

Towards Makespan Minimization Task Allocation in Data Centers

Kangkang Li, Ziqi Wan, Jie Wu, and Adam Blaisse

Department of Computer and Information Sciences
Temple University
Philadelphia, Pennsylvania, 19122
Emails: {kang.kang.li, ziqi.wan, jiewu, adam.blaisse}@temple.edu

Abstract—**Nowadays, data centers suffer from resource limitations in both the limited bandwidth resources on the links and the computing capability on the servers, which triggers a variety of resource management problems. In this paper, we discuss one classic resource allocation problem: task allocation in data centers. That is, given a set of tasks with different makespans, how to schedule these tasks into the data center to minimize the average makespan. Due to the tradeoff between locality and load balancing, along with the multi-layer topology of data centers, it is extremely time consuming to obtain an optimal result. To deal with the multi-layer topology, we first study a simple case of one-layer cluster and discuss the optimal solution. After that, we propose our hierarchical task allocation algorithm for multi-layer clusters. Evaluation results prove the high efficiency of our algorithm.**

Keywords–Task allocation; Data centers; Makespan.

I. Introduction

Modern data centers comprise tens of thousands of computers interconnected between each other with commodity switches. Nowadays, data centers suffer from resource limitations in both the limited bandwidth resources on the links and the computing capability on the servers, which triggers a variety of resource management problems [1–6]. One classic issue is the task allocation problem, which involve various constraints, including performance, network, and cost.

Generally speaking, a task usually consists of two parts: the computation workloads and the communication traffic. Different tasks running in the data center will compete for computing capability on the servers and bandwidth resources on the links. Obviously, to minimize the duration of communication traffic, locality is one important factor that we need to take into consideration. That is, we need to place the tasks as close to each other as possible, in order to lower the hops of communication between tasks. Furthermore, the tasks running in the same server will not need communication traffic between each other due to the internal communication within the server.

However, the bandwidth capacity of each link is limited. More tasks running under the same link will lead to the lower average bandwidth allocated to each task, which will decrease the communication speed and increase the communication duration. What is worse, the computing capability of a server is limited. If tasks are packed together, it will also reduce the computing capability allocated to each task. With the degraded computation speed, the computation workloads will take more time to complete.

On the other hand, if we apply load-balancing and allocate the tasks evenly to the servers, the computing speed of each task can be maximized [6]. However, the geographically separated tasks need more hops to communicate with each other. With the limited bandwidth resources on the links, load-balancing will considerably lengthen the communication time. Therefore, there is a tradeoff between the locality and load balancing.

The remainder of the paper is organized as follows: In Section II, we introduce some related work. In Section III, we present the task model discussed in this paper. In Section IV, we formulate the task allocation problem in the data center. In Section V, we study the simple case of a one-layer cluster and discuss the optimal solution. Section VI focuses on the multi-layer cluster and gives the hierarchical task allocation algorithm. Section VII conducts the simulations to validate the efficiency of our algorithm. Finally, conclusions are in Section VIII.

II. Related Work

Task allocation has been an open research topic since the traditional distributed computing era. Many task allocation issues are NP-hard, thus, we need to find a good heuristic algorithm to solve the problem, such as the first-fit and best-fit greedy algorithm used by Xu and Fortes [3].

Take the Mapreduce jobs as an example. One Mapreduce job consists of three tasks: map, shuffle and reduce. Many MapReduce schedulers have been proposed to try maximizing the resource utilization in the shared MapReduce clusters. Zaharia et al. [7] introduced delay scheduling that speculatively postpones the scheduling of the head-of-line tasks and ameliorate the locality degradation in the default Hadoop Fair scheduler. In addition, Zaharia et al. [8] also proposed Longest Approximate Time to End (LATE) scheduling policy to mitigate the deficiency of Hadoop scheduler in coping with the heterogeneity across virtual machines in a cloud environment. Ahmad et al. [9] proposed a communication-aware placement and scheduling of MapTasks and predictive load-balancing for ReduceTasks to reduce the network traffic of Hadoop on heterogeneous clusters.

Virtual machine placement is similar to the task allocation problem under the environment of cloud computing. As data centers are becoming the mainframe of cloud services, the virtual machine placement problem in data centers has been an open research area, considering both the network and servers. Piao et al. [1] gave a heuristic algorithm to satisfy both the communication demands and physical resource restrictions. Meng et al. [2] proposes minimizing the traffic cost through virtual machine placement. Their objective is to place virtual machines that have large communication requirements close to each other, so as to reduce network capacity needs in the data center.

Oktopus [4] uses the hose model to abstract the tenant's bandwidth request, including both virtual cluster and over-subscribed virtual clusters. They propose a virtual machine placement algorithm to deal with homogeneous bandwidth demands. The virtual cluster provides tenants with guarantees on the network bandwidth they demand, which, according to Popa et al. [5], can be interpreted as the min-guarantee requirements. However, this min-guarantee fails to consider the potential growth of a tenant's network demand. In order to alleviate this problem, our previous work [6] proposed the concept of elasticity, favoring the on-demand scaling feature of cloud computing.

Our previous work [6] designed a recursive abstraction scheme and hierarchical virtual machine placement algorithm, which is similar to the task allocation scheme in this paper. However, this paper focus on the objective of task makespan minimization and does not consider the environment of cloud computing. Furthermore, we focus on the study of tradeoff between locality and load balancing when doing task allocation. The communication model between tasks is different from the hose model for virtual machines' communications used in [6].

III. Task Model

In our model, each task can be separated into three parts: the pre-computation part, the computation part and the post-computation part, as shown in Figure 1. Here, we study the homogeneous task inputs. That is, all the tasks share the same pre-computation, communication, and post-computation workloads. To normalize these different types of workloads, we consider the situation that there are two tasks running in the data center and they are allocated to two servers under the same switch. Then we define the normalized time for each step as the time those two task go through each step, which is noted as α, 2β, and γ time units, respectively. Here we use 2β for two tasks communicating at the same time. If there is only one task fully using the bandwidth resource during the communication period, then its communication time is β.

Obviously, when $\alpha + \gamma \gg \beta$, the communication time can be neglected. Then the best choice is load-balancing and to evenly divide the input tasks into the servers. On the other hand, when $\alpha + \gamma \ll \beta$, the pre-computation and post-computation time can be neglected. Therefore, the best allocation scheme is to try to put all tasks into one server to minimize the communication cost.

We assume that all the tasks will start their pre-computation part at the same time. After finishing the pre-computation part, a $task_i$ will try to communicate with another $task_j$. However, due to the different completion times of the pre-computation parts of different tasks, $task_j$ might not finish its pre-computation part. Then, $task_i$ must wait for the $task_j$ to complete its pre-computation, and then, they can carry on the communication part with each other. After the communication part is finished, $task_i$ will start to complete its post-computation. Thus, the total makespan of a single $task_i$ is the sum of pre-computation time, waiting time, communication time, and post-computation time.

Pre-Compute
Communication
Post-Compute

Figure 1: Task model

IV. Problem Formulation

In this section, we formulate the average makespan minimization task allocation problem in data centers with the topology of a multi-layer binary tree. The data center configuration is *semi-homogeneous*, as shown in Figure 2. Each server has the same computing capability of C. Also, each link of the same layer has the same bandwidth capacity: L_k (the k^{th} layer link bandwidth capacity). However, the upper layer links have twice the bandwidth capacities than the lower layer links, i.e., $L_1 = 2L_2 = 4L_3$. The links capacities only differ between layers, we refer to this as the *semi-homogeneous* configuration, which is widely used to ease upper-layer link congestion. Our objective is to minimize the average makespan of all the input tasks, which can be expressed below:

$$makespan(i) = pre(i) + wait(i) + commun(i) + post(i) \quad (1)$$

$$\overline{makespan} = \frac{\sum_i^N makespan(i)}{N} \quad (2)$$

In Equation (2), $makespan(i)$ is the makespan of $task_i$, $pre(i)$ is the pre-computation time of $task_i$, $wait(i)$ is the waiting time between pre-computation and communication, $commun(i)$ is the communication time of $task_i$, $post(i)$ is the post-computation time of $task_i$. N is the total number of the tasks running in the system. Therefore, we tried to design a good task allocation scheme to minimize the average makespan of the input tasks.

Considering that the server's computing capability is steady and limited, then the speed of computation is inversely proportional to the number of tasks computed at this time on this server.

Considering that the bandwidth is equally shared by different tasks communicating through the link, then the communication speed is also inversely proportional to the number of tasks communicating between the servers.

In this paper, we study a simple case, in which the bandwidth resources are equally allocated to each task. In that case, given the the link bandwidth capacity B, the bandwidth allocated to each task is $\frac{B}{x}$, where x is the number of tasks go through that link.

Take one server to analyze. Assume there are x tasks in the server. Take $f_1(i, x)$ to be the time it takes $task_i$ to finish its first step in the server, where x is the number of tasks running on this server. Then we have:

$$f_1(1, x) = f_1(2, x) = \cdots = f_1(x, x) = x\alpha \quad (3)$$

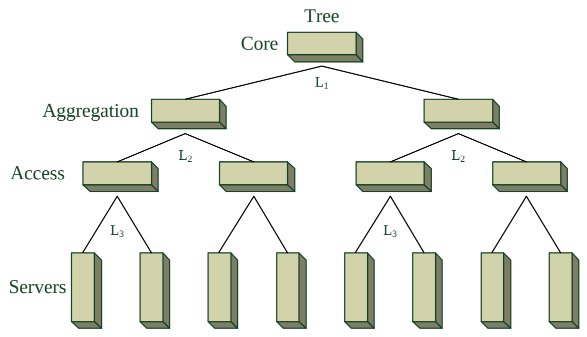


Figure 2: Tree-based network topology

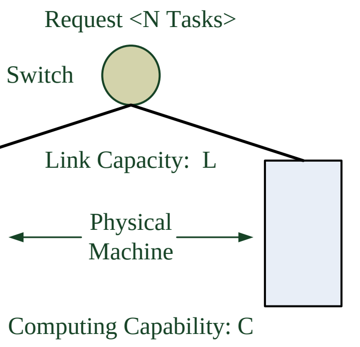


Figure 3: One-layer cluster

Equation (3) indicates that first step finish time of the first task finished in the first step is proportional to the the number of tasks running on the server. This is due to the fact that, tasks on the same server evenly sharing the computation resources. It also shows the finish time of the first step is equal for every task when the pre-computation workload is identical.

Define $f_2(t,x)$ as the number of tasks that have finished step 1 in the server at the time point of t. The we can get

$$f_2(t,x) = \begin{cases} 0 & \text{if } t < \alpha x \\ x & \text{if } t \geq \alpha x \end{cases} \tag{4}$$

t is the time from the the start of the pre-communication.

V. A ONE-LAYER CLUSTER STUDY

With N tasks accepted into the cluster, consider the simple case of two servers with a switch above them, as shown in Figure 3. Since all the tasks are homogeneous, we assume that, without loss of generality, we place x tasks in the left server, and leave $N-x$ tasks to the right server, and $x \geq N-x$. Due to the symmetry of binary-trees, there are, at most, $\lceil \frac{N}{2} \rceil$ different ways to allocate the tasks into two servers as the value of x increase from $\lceil \frac{N}{2} \rceil$ to N.

Let $p(t,x)$ be the probability of a task available for a communication step at time point t, with x tasks in the server.

$$p(t,x) = \frac{f_2(t,x)}{x} \tag{5}$$

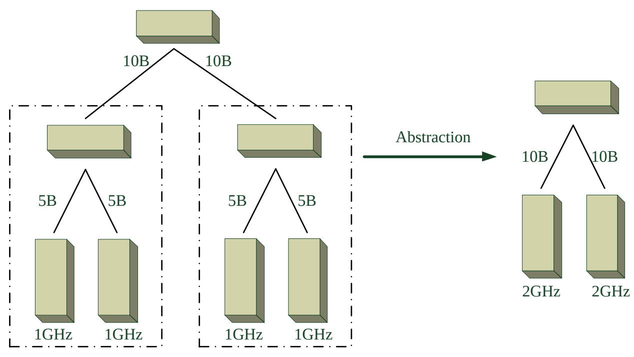


Figure 4: Abstraction process

We assume a task will randomly choose another task to communicate with. Then the probability of $task_i$ and $task_j$, both being available to communicate with each other at time point t is $p_i(t) \times p_j(t)$. In the one-layer binary tree model, there are only 2 possible relationships of $task_i$ and $task_j$, either in the same server, or in the different servers. Let $f_3(t,x)$ be the number of tasks able to start communication at time t in this one-layer cluster, then we can figure out:

$$f_3(t,x) = x \times p(t,x) + (N-x) \times p(t, N-x) \tag{6}$$

The internal traffic of tasks running on the same server will introduce no communication cost. If two tasks communicate in the same server, then their communication time will be ignored. Therefore, our focus is located on the traffic between tasks on different servers. Since the allocated bandwidth of tasks on the two servers might be different, we adopt the minimal one as the bottleneck of communication. However, if there are only two servers in the cluster, all the traffic outside the servers go through both sides of the switch. Then the outbound link bandwidth will be equally shared for both links, when they have the same communication capacities.

Let $f_4(t,x)$ be the number of tasks the have finished the communication part at time t, with x tasks in the left server, and $N-x$ tasks in the right server. The average number of tasks communicating between the two servers at time t is $f_3(t,x) - f_4(t,x)$. For a $task_i$ communicating with a task on the other server, let $t_{sc}(i)$ be the communication starting time of $task_i$, $t_{fc}(i)$ be the communication finishing time of $task_i$. Then we can get the constrains as follow

$$f_3(t_{sc}(i), x) = i \tag{7}$$

$$f_4(t_{fc}(i), x) = i \tag{8}$$

$$\int_{t_{sc}(i)}^{t_{fc}(i)} \frac{1}{f_3(t,x) - f_4(t,x)} dt = \int_0^{\beta} \frac{1}{1} dt = \beta \tag{9}$$

Let $f_5(t,x)$ be the number of tasks that have finished all the three parts (pre-computation, communication and post-computation) at time t. We further assume that $f_{5L}(t,x)$ is the number of tasks that have finished on the left server at time t, and let $f_{5R}(t,x)$ be the number of tasks that have finished on the right server at time t. Similarly, we can split $f_4(t,x)$ into $f_{4L}(t,x)$ and $f_{4R}(t,x)$. Likewise, let $t_{spc}(i)$ be the post-computation starting time of $task_i$, and let $t_{fpc}(i)$ be the post-computation time of $task_i$, and $task_j$ can be either in the left or right server. Then, we can get

$$f_{4L}(t,x) = \frac{f_4(t,x) \times x}{N} \tag{10}$$

$$f_{4R}(t,x) = \frac{f_4(t,x) \times (N-x)}{N} \tag{11}$$

$$\int_{t_{spc}(i)}^{t_{fpc}(i)} \frac{1}{f_{4L}(t,x) - f_{5L}(t,x)} dt = \int_0^{\gamma} \frac{1}{1} dt = \gamma \tag{12}$$

or

$$\int_{t_{spc}(i)}^{t_{fpc}(i)} \frac{1}{f_{4R}(t,x) - f_{5R}(t,x)} dt = \int_0^{\gamma} \frac{1}{1} dt = \gamma \tag{13}$$

$$f_{5L}(t,x) = f_{5L}(t,x) + f_{5R}(t,x) \tag{14}$$

Apparently, by solving $f_5(t,x) = i$, $i = 1, 2, ..., N$, we can get the finish time of each task, say $t_1(x), t_2(x), ..., t_N(x)$. Then the object is to minimize the average makespan of tasks, which is $\frac{t_1(x)+t_2(x)+...+t_N(x)}{N}$. Thus, given a one-layer cluster with 2 servers and N tasks, traverse all the possibilities to partition the N tasks into 2 servers. Since two servers are identical, we just need to choose x from $\lceil \frac{N}{2} \rceil$ to N for one server, and $N - x$ for the other one. Since all the tasks are identical, then the complexity will be $O(N)$. Then, we choose the best number to be put in the left server as x, remaining $N - x$ tasks will be put in the right server in the one-layer cluster.

VI. Multi-layer cluster Study

Given a binary tree multi-layer cluster with M servers and N task requests, we can get the optimal allocation scheme by traversing all the possibilities to partition the N tasks into M servers; however, it will be extremely time-consuming. Due to the NP-hardness of this problem [4], there is no optimal solution in polynomial time. However, based on the optimal results of the one-layer cluster, we can generalize this solution to multi-layer clusters, which has a considerably low time complexity.

For the bottom-layer access switches, we can view them as the root of a one-layer binary cluster, and try to abstract it into a single node. Each server under the access switch shares the same computing capability of C, and the sum of computing capabilities under the access switch is $2C$. Since the upper-layer links have twice the bandwidth resources as the lower layers; therefore, we can view this $2C$ as the accumulative computing capability of the abstraction node.

Based on this abstraction of the bottom-layer switch, we are able to abstract the entire multi-layer cluster to a one-layer cluster in a similar way. For each layer's switch connecting two sub-trees from the bottom to top, we can view it as the root of a one-layer binary cluster, and try to abstract it into a single node. Upon reaching the root switch at the top, the whole multi-layer cluster is abstracted into a one-layer cluster. We can see that our abstraction process misses no information. For abstraction nodes with the same accumulative computing capability, the inner structure of the original sub-trees are the same. We can see that, in Fig. 4, two servers with two lower-layer links are abstracted into a single node of accumulative computing capability of 2GHz. Both abstraction nodes with a computing capability of 2GHz share the same configuration of the original abstracted one-layer sub-tree.

Algorithm 1 Hierarchical Task Allocation Algorithm

Input: The links capacity and servers computing capability; Task requests $\langle N \rangle$

```
1: for layer $i$=1 to N do
2:     for all switches in layer $i$ do
3:         Calculate the accumulative capacity for each switch
           connecting two sub-trees in the layer
4: if input tasks could be accepted then
5:     for layer $j$=N to 1 do
6:         for all switches in layer $j$ do
7:             Optimally allocate the given number of tasks to
               this subtree
```

1) Computing capability Sharing of Multi-layer Clusters: The accumulative computing capability of the abstraction node is equally divided by all tasks allocated in this node.

2) Bandwidth Sharing of Multi-layer Clusters: For multi-layer clusters, the upper-layer link is connecting with a subtree with multiple servers. The link bandwidth would be equally divided by all tasks under that sub-tree.

3) Hierarchical Task Allocation Algorithm: With the abstraction of a multi-layer cluster into a one-layer cluster, we can use the optimal solution for a one-layer cluster. Based on that result, we propose our hierarchical task placement algorithm.

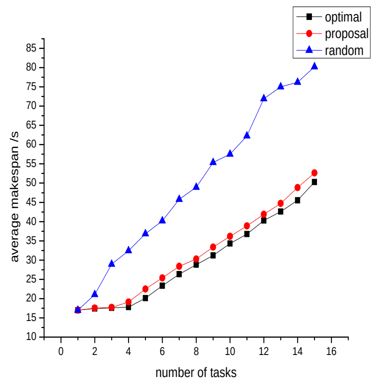
Given N input tasks, our algorithm can be divided into two steps. First, for each switch at each layer from bottom to top, the accumulative computing capability of the abstraction node rooted at that switch is calculated.

Second, for each switch connecting two sub-trees at each layer from top to bottom, recursively allocate the input tasks into the its two sub-trees according to our optimal one-layer solution. Upon finishing the bottom-layer switch (access switch), all the tasks are allocated into the servers. We summarize our algorithm in Algorithm 1.

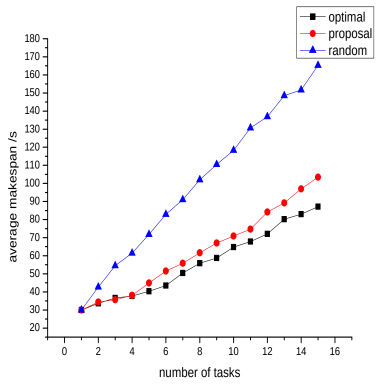
Our algorithm takes two loops. The first is the abstraction from bottom layer to top, and the second is the allocation from top to bottom. For the first loop, each switch at each layer is abstracted. For a K layers cluster, the total number of switch is $\sum_{k=1}^{K-1} k$. Suppose we have M servers at the bottom, then $M = 2^K$. Therefore, it takes $O(M)$ for the first loop of abstraction process. In the second loop, for each layer, our algorithm takes $O(N)$ time to calculate the optimal solution, based on the discussion in Section V. In the K-layer cluster, it takes $O(KN)$ for the allocation process. In sum, the total time complexity of our algorithm is $O(KN + M)$, which is very efficient.

VII. Evaluations

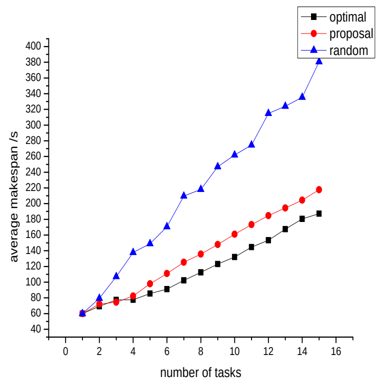
In this section, we evaluate our proposed algorithm in the case of a 2-layer binary tree data center. We made comparisons



(a) $\alpha = 2, \beta = 2, \gamma = 15$

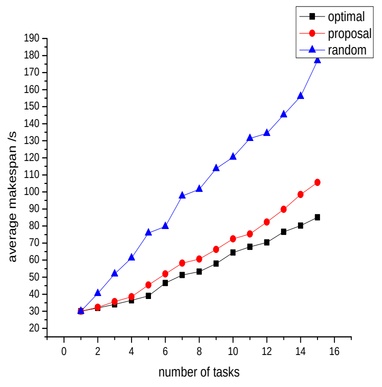


(b) $\alpha = 15, \beta = 2, \gamma = 15$

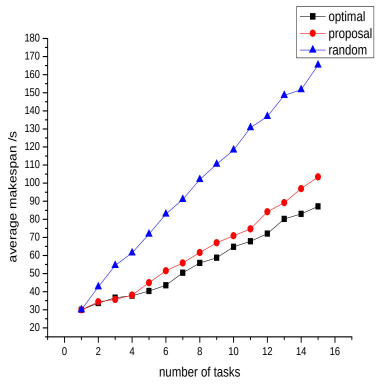


(c) $\alpha = 45, \beta = 2, \gamma = 15$

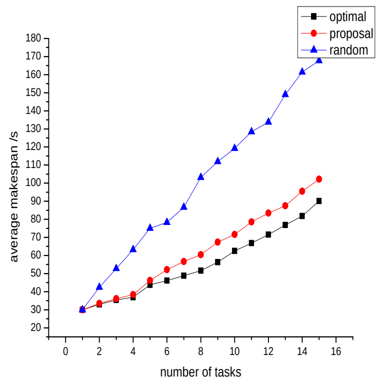
Figure 5: Performance comparisons of average makespan vs. the value of α



(a) $\alpha = 15, \beta = 1, \gamma = 15$



(b) $\alpha = 15, \beta = 2, \gamma = 15$



(c) $\alpha = 15, \beta = 4, \gamma = 15$

Figure 6: Performance comparisons of average makespan vs. the value of β

with our proposed algorithm with the optimal brute force algorithm and the random allocation algorithm. We produce the optimal solution by programs that traverse all the possibilities dividing the N inputs into M servers. The random allocation algorithm is generated by putting random number of tasks into different servers.

We evaluate the average makespan of our algorithm under three groups of simulations on the average total completion time. As shown in Figures 5, 6 and 7, our proposed algorithm is very close to the optimal one, which is much better than the random algorithm.

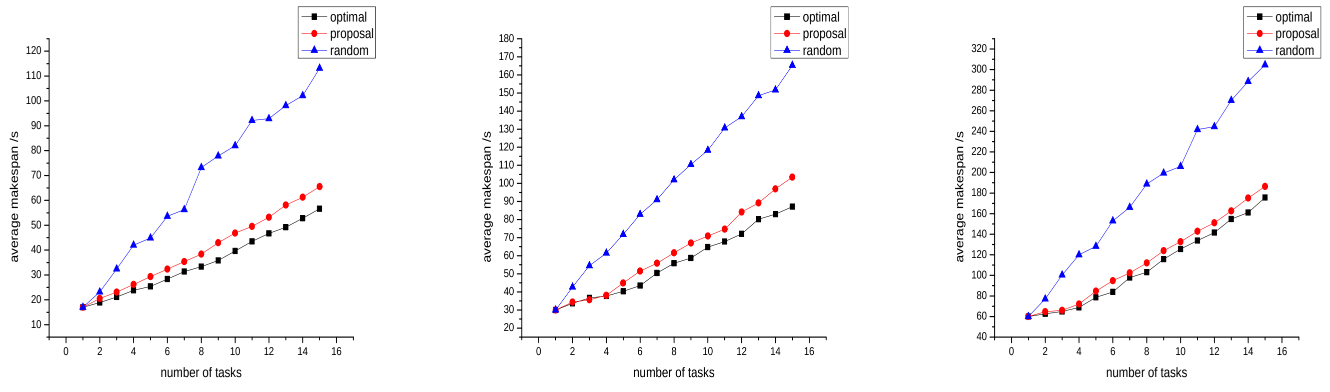
A. Simulation Settings

- Group 1: We select the α to be 2, 15 and 45 separately. And we set β, γ as 2 and 15. The bandwidth of links in each layer is equal. Task numbers go from 1 to 15.
- Group 2: We select the β to be 1, 2 and 4 separately. We also set α, γ as 15 and 15. The bandwidth of links in each layer is equal. Task numbers go from 1 to 15.
- Group 3: We select the γ to be 2, 15 and 45 separately. We also set β, α as 2 and 15. The bandwidth of links in each layer is equal. Task numbers go from 1 to 15.

B. Simulation Results

The results for the three groups of simulations are shown in Figures 5, 6, and 7. From those, we can see that when the number of tasks grows, the proposed algorithm's performance will deviate from that of the optimal one. However, the proposed solution does not deviate far from the optimal solution, and still follows the growing pattern of the optimal solution. Besides that, we still have the following observations:

1) For different pre-computation times (α), as shown in Figure 5, when the pre-computation time grows, the proposed algorithm's performance will deviate farther from that of the optimal one.
2) For different communication times (β), as shown in Figure 6, when the communication time grows, the difference between the proposed algorithm's average makespan and that of the optimal one will grow slowly.
3) For different post-computation times (γ), as shown in Figure 7, when the post-computation time grows, the the proposed algorithm's performance will be closer to that of the optimal one.



(a) $\alpha = 15, \beta = 2, \gamma = 2$ (b) $\alpha = 15, \beta = 2, \gamma = 15$ (c) $\alpha = 15, \beta = 2, \gamma = 45$

Figure 7: Performance comparisons of average makespan vs. the value of γ

As it is shown, the post-computation does not have the same influence as the pre-computation time or communication time does. One reason is that the post-computation is independent with the waiting time. Notably, the waiting time can inevitably increase the makespan of a task. Therefore, it makes sense for the pre-computation time to contribute more deviation than the post-computation time does. The smaller portion of the pre-computation time the better performance will be. The other reason is that our proposed algorithm will loss some information about the bandwidth during the abstraction step. Also, the more congestion in the bandwidth resources, the larger the difference between the proposed solution and the optimal solution will be. However, even the performance varies with the ratio of $\alpha : \beta : \gamma$, our proposed algorithm is still much better than the random one.

VIII. Conclusion and Future Work

In this paper, we study the classic task allocation problem in the data center, which is now suffering from both the limited bandwidth resources and computing capabilities. Compared to the previous work, we focus on the tradeoff between locality and load balancing. To minimize the makespan of input tasks into the multi-layer data center, we first study the one-layer cluster and discuss the optimal solution. After that, we propose our hierarchical task allocation algorithm to deal with the multi-layer cluster. The evaluation results show the high efficiency of our algorithm.

In this paper, we only study the homogeneous task model under the semi-homogeneous data center configuration. In our future work, we will first extend the task model into heterogeneous settings. That is, each task will have different values of α, β, and γ.

Furthermore, the heterogeneous configuration of data centers will also be studied. We will consider two sets of heterogeneous data center configurations. The first is that, we let the servers' computing capabilities heterogeneous, while keeping the links capacities semi-homogeneous as before. The second will be that, we let all the links capacities, along with the servers' computing capabilities heterogeneous. We will evaluate the efficiency of algorithm both theoretically and experimentally.

References

[1] J. T. Piao and J. Yan, "A network-aware virtual machine placement and migration approach in cloud computing," in *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pp. 87–92, Nov 2010.

[2] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–9, March 2010.

[3] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, pp. 179–188, Dec 2010.

[4] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, (New York, NY, USA), pp. 242–253, ACM, 2011.

[5] L. Popa, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "Faircloud: Sharing the network in cloud computing," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, (New York, NY, USA), pp. 22:1–22:6, ACM, 2011.

[6] K. Li, J. Wu, and A. Blaisse, "Elasticity-aware virtual machine placement for cloud datacenters," in *Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on*, pp. 99–107, Nov 2013.

[7] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, (New York, NY, USA), pp. 265–278, ACM, 2010.

[8] Z. Guo and G. Fox, "Improving mapreduce performance in heterogeneous network environments and resource utilization," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pp. 714–716, May 2012.

[9] F. Ahmad, S. T. Chakradhar, A. Raghunathan, and T. N. Vijaykumar, "Tarazu: Optimizing mapreduce on heterogeneous clusters," *SIGARCH Comput. Archit. News*, vol. 40, pp. 61–74, Mar. 2012.

E-Marketplace for Cloud Services

Claudio Giovanoli, Prasad Pulikal, Stella Gatziu Grivas
Institute for Information Systems
University of Applied Science Northwestern Switzerland
Olten, Switzerland
{claudio.giovanoli, stella.gatziugrivas}@fhnw.ch
{prasad.pulikal}@students.fhnw.ch

Abstract— **With the advent of Cloud Computing and the advantages it brings to businesses, there is a keen interest by businesses to adopt these technologies from a strategic advantage viewpoint. This growing interest throws up a challenge to businesses to identify solutions and providers that fits their requirements. It becomes difficult for them to locate and evaluate services provided by multiple Cloud Service Providers. This is exactly where the concept of an Electronic Marketplace for Cloud Services (EMPCS) would fit in. The Cloud Services Brokerage (CSB) or an intermediary is an enhanced type of EMPCS that simplifies the process further by a single-point, aggregated and customized solution. Using research literatures for reviewing existing models of an Electronic Marketplace (EMP) or Electronic Markets (EM) this research tries to identify core components that form part of a "state of the art" intermediary based EMP designed specifically for a Cloud Services.**

Keywords: cloud services marketplace, cloud service broker, cloud services marketplace reference model.

I. INTRODUCTION

Cloud Computing is rapidly changing the way IT services and products are offered. These services are currently evolving and will continue to evolve further with the ever-increasing services and type of products being offered by various cloud service providers. Enterprises are forced to rethink their strategies for building an IT infrastructure and cloud services are helping them by keeping things simple. Organizations can in a way outsource the hassle of building and maintaining their internal IT infrastructure to specialized cloud service providers. It is difficult to find a single source for the varying cloud service requirements of an enterprise. Many cloud service providers, serve very specific cloud areas (e.g. Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) etc.) or type of cloud and they vary a great deal in terms of their specifications, pricing and other detail models. One option is for the enterprises to deal with multiple cloud vendors and service providers. This fragmentation will lead to complexities in terms of integration, maintaining and consuming these services.

An important challenge will be in identifying various service providers who cater to specific needs and requirements of an enterprise [1]. One approach will be to identify providers from a directory or yellow pages catering to these kinds of services. Another approach will be the use of an E-Marketplace catering specifically to this kind of services. Generic EMP's like Amazon and Ebay, deal with a wide variety of products or physical goods. Making a deal on such marketplaces is generally as simple as searching, analyzing and buying. In some cases it is also possible to do some bidding or negotiations before you finalize a deal. An EMPCS works on the same line. Although a simple marketplace where Cloud Users and Cloud Vendors strike deals for Cloud Services still does not address issues related to the fragmentation of such services. Enterprises still have to deal with the integration and maintenance of such services in case of multiple vendors and providers. This brings in the need for an intermediary or broker who would further simplify the consumption of Cloud Services [2]. This kind of a technology partner for enterprises is called the Cloud Services Broker or the Cloud Services Brokerage (CSB). The CSB or an intermediary improves upon the idea of an EMPCS by providing a single-point, aggregated and customized solution and owns the responsibility of security, governance and quality of all the services provided. Early pioneers in the role of CSB's are Jamcracker [13], Parallels [19] and AppDirect [20]. Various researches carried out by Buyya [3] and Gartner Inc. [2], have highlighted some detailed and useful information related to the role and model for a CSB.

The focus within this paper is twofold, firstly to research existing models of EMP's and secondly to identify processes and components that are required to build a state of the art model for EMPCS from a CSB point of view.

The contribution of this research paper is to identify the key activities that go within a CSB and also a proposed modification to the "Reference Model for Electronic Markets" (RM-EM) by Schmid & Lindemann [6] to suit the needs of a CSB. The newly proposed modification is keeping in mind the transactional phases that occur within a CSB model.

Section II, gives an overview on the concept of an EMP. Section III, presents an overview of existing reference models in the same area. Section IV, describes the findings on models that detail the core processes and activities that are part of an EMP model. After describing briefly Cloud Services in Section V, keys difference between an EMP and an EMPCS are described in Section IV. We also discuss our findings related to current CSB models in the same section. A proposal related to the suggested modifications in the RM-EM to suit a "state of the art" CSB model will be presented in Section VII. The paper concludes with Section VIII.

II. Electronic Marketplace

An Electronic Marketplace or E-Marketplace (EMP) refers to an online market environment over the Internet where both sellers and buyers act as market players to exchange goods and services.

EMP's are broadly categorized into different types according to their ownerships or governance [1]. These types are:

a. *Private Marketplace* - owned by an individual company, which focus on connecting buyers and sellers to the marketplace.
b. *Public Marketplace* - described as a free marketplace or Business-to-Business (B2B), is often owned by third party organization.
c. *Consortia Marketplace* - created as a result of competitive companies from same industry and having same set of products and services.
d. *Community Marketplace* - owned by multiple participants with varied backgrounds.

There are three main functions of a marketplace:

a. *Matching of buyers and sellers* - This is the major function within a marketplace. The products have to be aggregated and their features, specifications detailed so it becomes easy for a buyer to search and compare the products according to his needs.
b. *Facilitate the exchange of information, goods, services, and associated payments* - Logistics, payment settlement, communication and tracking fall under this functionality.
c. *Providing an infrastructure that enables the efficient functioning of the market* - Following of legal laws, government rules and regulations to allow the smooth functioning of the marketplace is a feature of this functionality.

III. Reference Model for An E-Marketplace

A Reference model is a framework for understanding significant relationships among the entities of some environment, and for the development of consistent standards or specifications supporting that environment. A reference model is based on a small number of unifying concepts and is an abstraction of the key concepts, their relationships, and their interfaces both to each other and to the external environment may be used as a basis for education and explaining standards to a non-specialist [5]. Further to this a reference model also forms the basis of designing specific models for actual implementation in reality.

The RM-EM defines three phases of a Market Transaction. They are depicted in Figure 1, and defined as follows:

a. Information Phase: is the phase of information gathering or evaluation phase of requirements and products from various sellers. It includes all analysis done to check the suitability of a product to a customers needs. The information phase lasts until products are chosen and an offer is received.
b. Agreement Phase: in this phase the conditions, pricing and other delivery related issues are negotiated. This phase ends with the signing of a legal contract between the customer and the supplier.
c. Settlement Phase: involves the delivery and payment of the final goods according to the agreement. Furthermore, it involves information about payment, logistics (transport, warehousing), Information Transmission, tracking and tracing.

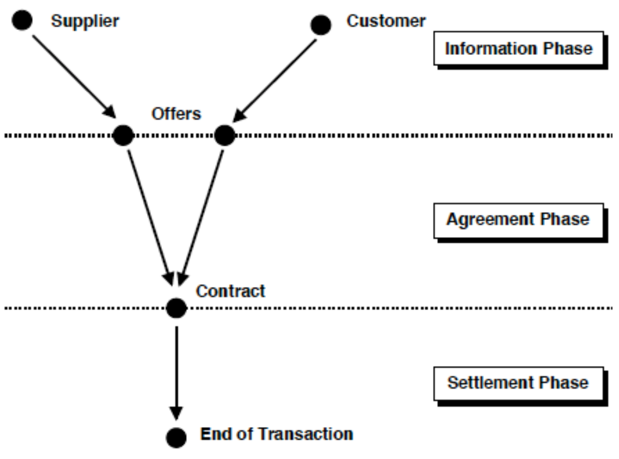


Figure 1: Phase Model of E-Market Transactions [6]

The existing RM-EM proposed by Schmid & Lindemann [6] consists of four layers and three phases. As shown in Figure 2, each layer represents a view for EMs. It consists of two dimensions. The horizontal dimension contains the three phases of market transactions, and the vertical dimension refers to the four layers or views (Business, Process, Transaction and Infrastructure). The upper two views represent the focus on organizational aspects and the two lower views represent the technological aspects of implementing such services.

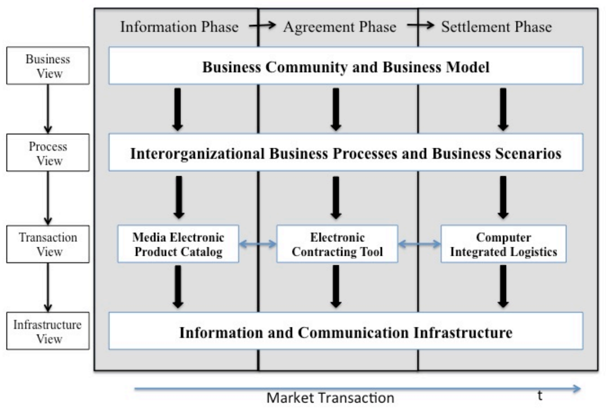

Information Phase
Agreement Phase
Settlement Phase
Business View
Process View
Transaction View
Infrastructure View
Business Community and Business Model
Interorganizational Business Processes and Business Scenarios
Media Electronic Product Catalog
Electronic Contracting Tool
Computer Integrated Logistics
Information and Communication Infrastructure
Market Transaction
t

Figure 2: Reference Model for Electronic Markets [6]

Selz and Schubert [7] define a similar model, which goes on to add an additional phase after the Settlement Phase called the "Communication phase" to phases of an E-Commerce transaction.

IV. MODEL FOR AN E-MARKETPLACE

Taking the RM-EM into account, we first look at the second layer in the model, i.e. the Process View, and we identify processes that are part of this view.

Before discussing models of a generic EMP, various activities or functions that are carried out within an EMP have to be explained.

An EMP system performs the following functions or services [8]:

- Organize the seller product Information
- Organize the buyer needs and preferences
- Facilitate commerce Transactions between the buyers and sellers
- Provide a flexible environment for multiple intermediaries to participate in the flow and monitoring of the commerce transactions.

Although there are many types of EMP's, and each type will have a set of activities associated with it, many of the activities are common across all the EMP's. These common activities can be classified as follows [9]:

- Content publishing tools
- Access management, authentication
- Bid/ask trading
- Auctions, on-sale
- Catalogs, Custom catalogs
- Parametric Search
- Product Configurators
- Aggregation
- Supplier Management
- Order Processing
- Invoicing
- EDI integration
- Event notification
- Fulfillment
- Business Rules facility
- Payment facility
- Workflow
- Reporting
- Integration to Back Office

A simplified list of processes (Marketing, Catalog Management, Order processing, Fulfillment and Settlement) form part of the model from Fingar, Kumar & Sharma [9]. There will be more functional processes required for running an EMP, but the above-defined processes form the core functions of an EMP. Figure 3 shows a graphical representation of an application framework for an EMP with above mentioned processes and their relation to external customers. Table 1 goes a step further and reveals various activities within these processes.

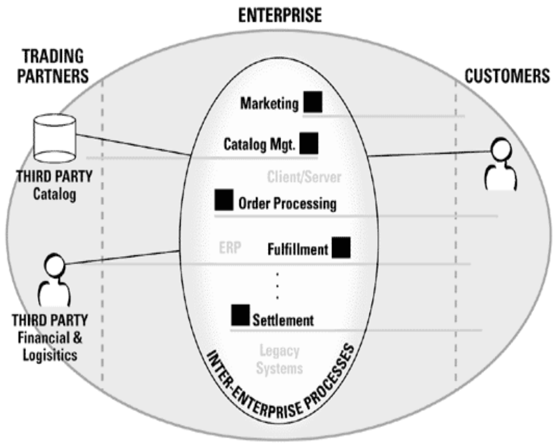


Figure 3: Processes within Electronic Markets [9]

1. Marketing: The Marketing process within an EMP involves all activities like identifying segments, products, building brand awareness and attracting customers. Defining target vendors and customers, promoting awareness; strategies for attracting new customers and customer retention are some functionalities, which play an important role in the marketing process of an EMP. A typical EMP has to identify the areas or specialized segments it prefers to work with, e.g. being clothing, sport, housing, electronics etc. A generic EMP caters to multiple fields and carries a larger number of products. The next stage is identifying and attracting the vendors or sellers who fit the segment description. Additionally, the process

involves identifying customers, segments, business or government agencies that would be interested in such field and attracting them to start using the EMP service. Advertising also forms a highly important activity within this process. As in a real world retailing market, building brand awareness and attracting customers is an important aspect with online markets. Furthermore, an EMP business has to analyze customer-buying behavior, industry trends and adapt its strategy accordingly. E.g. a customer who has recently bought a car from an automotive EMP will definitely be interested in accessories related to his car. A personalized marketing strategy that caters to existing customers and their recent buying behavior is definitely one of the activities that will stand out in a highly competitive industry. An intelligent system that predicts customer behavior will be an attractive prospect for a marketing process within an EMP.

2. Catalog Management: refers to the various aspects of aggregation of different products into a structured and consistent way. A new Vendor who is trying to sell his product through an EMP should have simple interface through which he can select product categories and features. For a customer who is trying to buy a product, it should be very simple to search and find a product according to his need. For segments with complex products and requirements, an ontology based approach for generating product classification is very useful. Shared and agreed ontology provides common, flexible, and extensible definitions of products and requirements for matchmaking and subsequent business processes [12].
3. Order Processing: in this process, users finalize a product that fits their requirement and then go on to order it. This process starts with the product being added into the shopping cart and is followed by functionalities like the customer authentication, gathering the billing and shipping information, calculating the cost involved in shipping and addition of taxes to the cost and more. Finally, the customer proceeds to confirm the order. If online, the billing methods are authenticated and an electronic receipt is sent to the customer via his email; additionally, the receipt can be retrieved via his account within the EMP. Also, part of the process is providing a feature for the customer to track his orders from within his account.
4. Fulfillment or Logistics**:** Right after the order is completed, it moves onto the fulfillment or logistics stage. It involves initiating the logistics related to packing and shipping a product to the customer. It starts with informing the external vendor of the order and shipping details. Once the order is shipped, the vendor updates the status into the EMP's system and it shows up within the customer account and can be tracked regularly.
5. Settlement: The settlement process involves clearing or settling of payments with the merchant banking if credit card transactions are involved. If other payment methods are involved for, e.g. payment on delivery, a receipt confirmation and updating of the payment status into the EMP system are activities that need to be performed. A major activity in this process will be also the settling of dues with the external vendors or supplier. This can be instant settlement once order is shipped or can be done at regular intervals, e.g., monthly or quarterly.

In addition to these processes some EMP models specify an additional process related to customer support, though, in the above model, this activity is included within the settlement process. More and more companies have a strong customer support service, which helps in building a reputation among existing customers. This in turn results in satisfied customers who use the services again.

TABLE 1: PROCESSES AND ACTIVITIES WITHIN ELECTRONIC MARKETPLACES [9]

PROCESS	ACTIVITIES
Marketing	▪ Merchandising ▪ Advertising ▪ Brand & Service Awareness ▪ Promotion Strategy ▪ Personalized Cross & up-selling ▪ Trend Tracking& Analysis
Catalog Management	▪ Ontology / Catalog Classification ▪ Content Publishing ▪ Catalog Management ▪ Catalog searching ▪ Availability checking ▪ Price and feature comparison ▪ Personalization
Order Processing	▪ Shopping Cart ▪ Approval and Authorization ▪ Inventory Update ▪ Tax and Cost Calculation ▪ Shipping Estimates. ▪ E-Receipts ▪ Order status/ tracking
Fulfillment	▪ Warehouse Integration ▪ Vendor Communication ▪ Packaging and Shipping ▪ Inventory Updating ▪ Reporting
Settlement	▪ Invoicing ▪ Payment Processing ▪ BackOffice Integration
Support	▪ Helpdesk ▪ Payment issue resolutions ▪ Returns

V. Cloud Computing Services

Cloud Computing or Cloud Services refers to a delivery mechanism through which on demand computing resources are delivered to end user terminals (Personal Computers, Mobiles, Tablets, etc.), generally over the internet. Cloud Services have some major characteristics or advantages that set them apart from the traditional form of computing. Firstly, the user pays by usage, secondly, it is scalable or elastic, i.e., users can upgrade a service anytime they require more resources and thirdly, it is completely hosted and managed by the Cloud Service Provider (CSP), i.e. the user only requires an internet connection and a computer to access the service. Cloud Services can be broadly classified in three major categories:

1. Software as a Service (SaaS)
2. Platform as a Service (PaaS)
3. Infrastructure as a Service (IaaS)

With technological advances more and more services are moving into the cloud and so the categories of cloud services have been expanding. New forms of emerging Cloud Services are *Database as a Service* (DBaaS), *Communication as a Service (CaaS), Network as a Service (NaaS), Gaming as a Service (GaaS)* etc. Most of these emerging forms of Cloud Services can however be broadly classified into the above-defined categories.

VI. E-Marketplace for Cloud Services and Cloud Service Brokerage

Taking our research further from generic EMP's to specific EMP's tailored for Cloud Services or EMPCS, we identified existing models related to this segment.

Garg, Buyya, and Vecchiola [10] have proposed a design for a market exchange framework called "Mandi" which allows consumers and providers to trade computing resources according to their requirements. The primary aim of Mandi is to provide a marketplace where the consumer's resource requests and the provider's compute resources can be aggregated, and matched using different market models.

The main components defined in the Mandi architecture are:

1. User Services
   - *Registration Service*: a user has to register before accessing the services of the exchange. The user details are stored in the database layer and are used for the purpose of authentication.
   - *Authorization/ Authentication Service*: service to authenticate a user and provide him access to eligible services.
   - *Resource Service*: allows a customer to search and find services that fit his requirement.
   - *Auction Service:* service required in case of bidding marketplace and allows the user to join in any auction and place his bid.
2. Core Services
   - Meta-Broker Service: this service manages every single auction. It conducts the auction and all related activities.
   - Database Service: storage service, which maintains all the information and stores data to every single activity within the system, e.g. product details, user details, user bids, auction details, winner details and more.
   - Reservation Service: reserves the service and confirms to the service provider of the customers interest.
   - Accounting Service: stores the trading information of the user, like bids, successful and failed.

Before going ahead in detailing the models for an EMPCS, some major differences between EMP's and EMPCS's will have to be explained. The one major difference is the type of products that are traded within these marketplaces. On one side through an EMP consumers buy or sell physical goods that will be shipped to us, while within EMPCS there are no physical goods. It is all virtual services that are bought and sold through an EMCPS. This reduces some activities like packing, shipping and logistics involved in delivering these goods. Another major factor of differentiation is that the services traded through EMPCS are not one-time deals. Rather, after buying of services, the services themselves need to be managed, maintained and updated by the trader or by the intermediary managing the EMPCS. So in a way it is long-term relation between the customer and the Services Provider. In case of a one time payment or deal based cloud product, there still has to be a support and other help provided to the user. Another major difference we identified through our research on this subject is that with such a new technology and multiple service providers, it is difficult for the customer to identify all the services he requires. Cloud Services requirement detailing, identification, discovery, integration and maintenance will always remain a major issue for customers due to non-standardized features, description and type of these services. This is exactly where the role of an intermediary or Cloud Services Brokerage (CSB) comes into play.

A CSB service gives the customer a solution for this exact problem with a single point, aggregated and personalized solution that fits well with the requirements of a customer needs. A Cloud Broker allows enterprises to put together services as required from a varied choice of service providers, platforms and cloud types. They also make it easier for the customer because they provide a simple and standardized single point interface.

Plummer et al. [17] define three types of CSB's:

1. Aggregation Brokerage: delivers multiple services to enterprises. No additional feature or functionality is provided from the brokerage. They simply aggregate and provide the services. Simple market places, App Stores, etc., fall under this category. GetApp [18], a cloud marketplace for SaaS applications, is a classic example of such a brokerage.
2. Integration Brokerage: this type of brokerages involves sourcing of varied type of services that are integrated at the broker's end and provided as a packaged service to the customers, enterprises or end-users.
3. Customization Brokerage**:** involves a personalized or customized integrated service, which adds or modifies functionalities or services to enhance its functionality. It also involves consulting and other services related to implementing the cloud strategy of an organization.

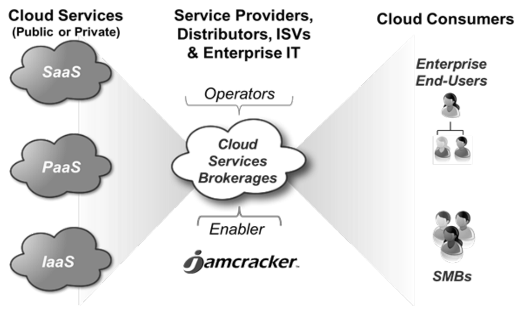


Figure 4: Cloud Services Broker Model [13]

By definition, a cloud broker is a third-party individual or business that acts as an intermediary between the purchaser of a cloud computing service and the sellers of that service. In general, a broker is someone who acts as an intermediary between two or more parties during negotiations. The CSB is an enhanced type of electronic marketplace for Cloud Services, which aggregate multiple service providers in a single location for customers to evaluate and buy Cloud Services. It also simplifies various other processes of maintaining, administration, billing and payment between external vendors and customers. Examples of CSB services are: *Jamcracker* [13] see also Figure 4*, Ensim* [14], *Deutsche Börse Cloud Exchange* [15] etc.

To begin, processes that form part of such frameworks will be listed. Many of the existing models of CSB inherit processes from the earlier model of EMP as described. At a higher level, processes such as *Marketing, Catalog Management, Order Processing, and Fulfillment* (minus logistics) form an integral part of a model for EMPCS. Additional processes like *Self-Servicing, Services Administration, Billing, Metering & Chargeback and Support* are also important from a CSB viewpoint. Figure 5 provides a graphical view of the above-mentioned processes within a CSB platform.

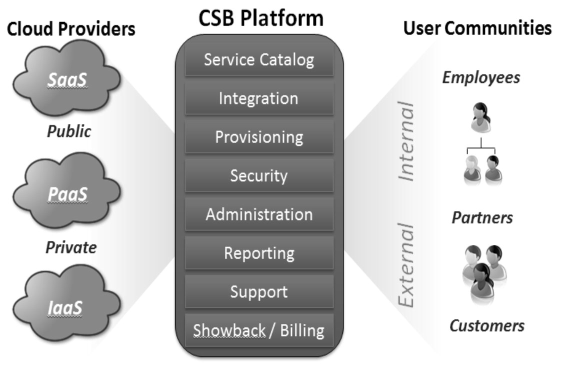


Figure 5: Processes in a CSB platform [16]

The processes and features we discussed above are described in detail in an online article by A. Mauro [16]. Figure 6 provides a graphical representation of the workflow between these processes.

Service Catalog: this is similar to the catalog management feature in generic EMP's. It involves aggregation of content from external vendors or suppliers. This aggregated content is presented in a simplified format, which makes it easier for the customer to search, compare and finalize required services. An ontology-based approach is highly important here as Cloud Services vary in type and distribution so it becomes difficult to classify these Cloud Services into a single catalog. With such a new platform, there are very few standard classifications or ontologies to refer to and hence this presents a new area of research altogether.

Integration: all external Cloud Services have to be integrated into the CSB platform so it is easier for customers to request a service that combines multiple providers. It provides the user with a single point access to combined services as per their needs. Also, billing and administration of services can be centralized into one single console once integrated. From an IT implementation point of view, the Vendors have to add specific API's provided by CSB platforms for billing and administration into their Cloud Services

Provisioning: a central console for controlling services, and user provisioning is an integral part of this process. Also part of this process is the policy based automated provisioning.

Authentication and Authorization: Single Sign-On (SSO) uses a federated identity that provides access to all services with a single password. Roles-based authorization and access control provides granular control of cloud services access based on user role and privileges, password policy enforcement: enables uniform password policy across the enterprise

Administration: administration of users accessing the service is part of this process. All activities related to user management, like new users registration, existing user management, deletion, updating of user details come under this process.

Reporting**:** reporting, notification, and usage statistics are part of this process. Customers can access all these details form the Management Console of their account.

Support: general helpdesk activities including submission of tickets, FAQ, knowledgebase, notifications and alerts are part of this process. Additionally, for a first level support the customer can be assisted directly by the service provider.

Billing: all the billing procedures take place within this process. The customer can access all details regarding the services, usage, components and costs related to these services. This process should accommodate "Pay Per Usage" type of billing. All usage reports can also be accessed to view the billing calculation on the console. Payment and settlement services are also integrated into the platform for billing.

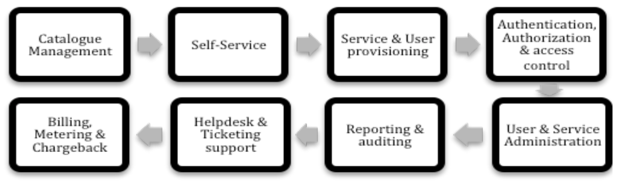

Catalogue Management
Self-Service
Service & User provisioning
Authentication, Authorization & access control
User & Service Administration
Reporting & auditing
Helpdesk & Ticketing support
Billing, Metering & Chargeback

Figure 6: Workflow of a CSB Platform [13]

VII. Proposed Reference Model of a Cloud Services Brokerage

Through the research on the Cloud Services and related EMPCS, it becomes obvious that the basic difference between an EMP and an EMPCS is more related to the final phase from the phase model of transactions in Section III of this paper. In an EMP, the products that are sold or bought are physical products or services and mostly the transactions in such cases end with the final settlement phase. Once the product is delivered and the payment is completed the transaction ends. In case of an EMPCS, there are no physical products that are traded. Rather, these are virtual computing services. Also, in case of an EMPCS, the settlement or the final phase is not the end of the transaction. Rather, it is the beginning of business relationship between the customer and cloud service provider.

Hence, a modification of the RM-EM to suit a CSB service is proposed. According to our proposal, within the Reference Model for Electronic Marketplaces a Relationship Phase can replace the Settlement Phase. This Relationship Phase at the Transaction View includes a centralized console that is responsible for all activities related to service delivery, administration, maintenance subscription, billing, management of the cloud service and support. The Relationship Phase continues as long as there is business relation between the customer and Service Provider. Figure 7 below describes the proposed changes.

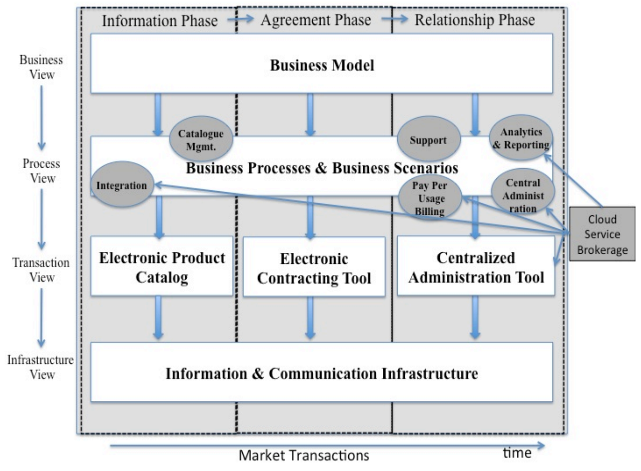


Figure 7: Proposed Reference Model for an EMPCS and CSB

Core activities within a "state of the art" CSB from our research findings are:

- Integration
- Catalogue Management
- Self Service & User Provisioning
- User Authentication and Administration
- Analytics & Reporting
- Billing – Including Pay per Usage.
- Support (Technical helpdesk & Generic support)

VIII. Conclusion

The current research findings have thrown some good insights into how a state of the art on Electronic Marketplace for Cloud Service should be modeled in real world. In general, many of the processes involved in a generic EMP are also part of an EMPCS model. However, to make an EMPCS or a CSB successful in the real world, activities like platform integration, centralized administration, maintenance, reporting, technical helpdesk, standardized billing also have to be included so that the user experience does not suffer because of the varied and complex Cloud Services available in the market. Most of these activities fall within the final phase, i.e. the Relationship Phase that replaces the Settlement Phase from RM-EM proposed by [6]. Some other major improvements we can suggest through our findings and analysis, are firstly, an ontology based approach for classifying the varied Cloud Services which gains a lot of importance within an EMPCS much more that a generic EMP. A standardized ontology for Cloud Services is definitely an important factor for organizations wanting to be successful as Cloud Service Brokers. Much more research is needed into such an approach. Secondly, intelligent systems that simplify the matching of customer needs to available products and search suggestions, also will add a lot to such a service. In terms of performance, such a system could decrease the customer's service evaluation time compared to current EMPCS. Intelligent agents within such systems can also ease out the human intervention required during integration of new services into a CSB platform.

REFERENCES

[1] S. Sundareswaran et al. , "*A Brokerage-Based Approach for Cloud Service Selection*"; IEEE Fifth International Conference on Cloud Computing, 2012, pp. 558-565.

[2] Gartner Inc. , "Three types of Cloud Brokerage will enhance Cloud Services", Gartner Incorporate, Stamford: 2009.

[3] R. Buyya et al. , "Cloudbus Toolkit for Market-Oriented Cloud Computing", Manjrasoft Pty Ltd, Melbourne, Australia, 2012.

[4] C. Standing, P. E. D. Love, R. Stockdale, and D. Gengatharen. "Examining the Relationship Between Electronic Marketplace Strategy and Structure", IEEE Transaction on engineering management, vol. 53, no. 2, 2006, pp. 297.

[5] D. Sawyer and L. Reich, "The Open Archival Information System: A Model for Preserving Digital Information", 2003, Tutorial Presentation: National Aeronautics and Space Administration (NASA).

[6] B. Schmid and M. Lindemann, "Framework for Specifying, Building, and Operating Electronic Markets", in International Journal of Electronic Commerce, Vol. 3, No. 2, Formal Aspects of Digital Commerce, Winter, 1998/1999, pp. 7-21.

[7] D. Selz and P. Schubert, "Web Assessment –A Model for the Evaluation and the Assessment of successful Electronic Commerce Applications", 1998.

[8] J. Sairamesh et al. , "E-Marketplaces: Architecture, Trading Models, and Their Role in Bandwidth Markets." transactions 6, no. 2, 1999, pp. 3-4.

[9] P. Fingar, H. Kumar and T. Sharma, "21st Century Markets: From places to spaces". First Monday (Online) , vol. 4, no. 12, Available: http://firstmonday.org/ojs/index.php/fm/article/view/707/617. 14.04.2014.

[10] S. K. Garg, R. Buyya and C. Vecchiola, "Mandi: a market exchange for trading utility and cloud computing services", 2011.

[11] A. Umar, "Third Generation Distributed Computing Environments", 2004, pp. 2-24.

[12] D. K. W. Chiu et al. , "How Ontologies Can Help in an marketplace." In: ECIS 2005 Proceedings, 2005, Table 1, pp. 2.

[13] Jamcracker: http://www.jamcracker.com/product/jamcracker-platform. 06.12.2013.

[14] Ensim Automation Suite: http://www.ensim.com/. 30.01.2014.

[15] Deutsche Börse Cloud Exchange, Zilch S. , Factsheet "The Worlds first vendor neutral Marketplace for IaaS resources", http://dbcloudexchange.com/media/20140217_DBCE_Factsheet_english.pdf. 17.02.2014.

[16] A. Mauro, "Jamcracker Cloud Broker Enabler", http://vinfrastructure.it/2013/08/jamcracker-cloud-broker-enabler/. 01.08.2013.

[17] C. D. Plummer et al. , "Cloud Services Brokerage Is Dominated by Three Primary Roles", Gartner, November 2011.

[18] GetApp: http://www.getapp.com. 06.12.2013.

[19] Parallels: http://www.parallels.com. 06.12.2013.

[20] AppDirect: http://www.appdirect.com 06.12.2013.

MuTeBench: Turning OLTP-Bench into a Multi-Tenancy Database Benchmark Framework

Andreas Göbel
Institute of Computer Science
Friedrich Schiller University Jena
Jena, Germany
Email: andreas.goebel@uni-jena.de

***Abstract*—Database benchmarks have been used for decades to load test systems and to compare systems or system configurations with each other. However, their methods and assumptions are hardly suitable for multi-tenant cloud database services. Those systems have to provide both performance isolation of a lot of tenants with dynamic workloads and cloud computing features like scalability, elasticity, and reliability. In this article, the open source benchmark framework MuTeBench is presented. It allows the creation of OLTP benchmarks for multi-tenant databases and combines extensibility, portability, and evolved workload support of the underlying OLTP-Bench with flexible scheduling, statistic gathering across tenants, and individual service level agreements.**

Keywords-Benchmarking; Multi-Tenancy; OLTP; Database System; Service Level Agreements.

I. Introduction

Centralization of infrastructure due to cloud computing is an increasingly important attempt of IT enterprises. Multi-tenant architectures enable cloud service providers to share resources and costs across various tenants (organizations, customers, or companies). Particularly, multi-tenancy database management systems (MT-DBMSs) have become an important field of research for academia and industry.

In MT-DBMSs, several or even all tenants share a single DBMS instance and its available resources. Assuming that not all tenants are active simultaneously, high resource utilization can be achieved by avoiding an allocation of resources required for the peak load of each tenant [1]. Consolidation can be implemented by three different approaches [2]. In the *shared machine* approach, each tenant receives a dedicated database resulting in high tenant isolation at the expense of high costs per tenant. In the *shared process* approach, tenants share databases but operate on separate tables or separate schemas. This approach enables partial resource sharing across tenants and allows an appropriate isolation level. The *shared table* approach achieves the highest degree of sharing and efficiency by sharing tables and indexes among tenants. A special column associates each row with the appropriate tenant. Allowing customized database schemas and individual administration for each tenant is very challenging with this approach.

Resource competition of simultaneously active tenants bears a new challenge for DBMSs, in particular with high degree of resource sharing. The performance of tenants must not be affected by resource-intensive activities and volatile workloads of other tenants, for example, in order to meet performance service level agreements (SLAs) of tenants. Moreover, tenant data has to be protected against unauthorized access by other tenants and a MT-DBMS must provide tenant metering, low operating costs and tenant-specific database schemas. These requirements are supplemented by providing general cloud computing features such as zero downtime, elasticity, and scalability. For scalability reasons, a MT-DBMS should run on low cost commodity hardware and scale out to a lot of servers for many customers.

Classical benchmarks are not able to adequately assess MT-DBMSs with respect to the above-mentioned requirements. A new generation of database benchmarks is required, which is suitable for the difficult terrain of clouds and multi-tenancy. In this article, appropriate methods and metrics of MT-DBMS benchmarks are summarized. The purpose, architecture, and configuration of the benchmark framework MuTeBench are presented. By some experiments, its suitability to evaluate MT-DBMSs concerning their major challenges is illustrated.

This article is structured as follows. Section II outlines challenges in benchmarking of MT-DBMSs and compares them against conventional DBMSs. Section III presents MuTeBench, a framework for creating MT-DBMS benchmarks with evolving tenant workloads. After discussing experiments in Section IV and related work in Section V, the article ends with conclusions of our contributions and open issues in Section VI.

II. MT-DBMS Benchmarking

The best known representative of benchmarks for transaction processing systems and databases are benchmarks of the Transaction Processing Performance Council (TPC) [3], which simulate real-world application scenarios. Like most traditional benchmarks, they provide an infrastructure to run a representative workload against a static non changing software system in order to assess its average performance under maximum load. Furthermore, some benchmarks include cost-based metrics. The static setups of those traditional database benchmarks contradicts to MT-DBMSs which have to handle a varying number of active tenants with changeable workload mixes and rates as described in [4]. For this purpose, they may need to allocate additional hardware or save costs by releasing underutilized resources. Therefore, evaluating MT-DBMSs requires benchmarks with the ability to run changing workloads of several tenants in parallel. Such benchmarks can be used by service providers to improve their services.

Under certain circumstances, they may also assist customers in finding an optimal database service.

A. Metrics

Appropriate metrics are needed to assess the impact of other active tenants on the query processing performance of a single tenant. Typical performance metrics, such as request throughput, average latency, or latency percentiles of a single tenant are only partially suited for this purpose. We call them *absolute tenant performance metrics*.

Kiefer et al. [5] proposed a metric called relative execution time. We have adopted this approach and expanded it by relative throughput, relative average latency, and relative latency percentiles, summarized as *relative tenant performance metrics*. For their calculation, the best possible tenant performance in a simulated single-tenant environment has to be determined first. For this purpose, tenants may run workload in an initial baseline run without any resource competitors. Following this, during actual multi-tenancy runs their absolute performance will be gathered. According to formulas 1 and 2, these results are set in proportion to their baseline equivalents relatively, resulting in relative performance values.

$$throughput_{rel} = throughput_{abs}/throughput_{base} \quad (1)$$
$$latency_{rel} = latency_{base}/latency_{abs} \quad (2)$$

Performance SLAs and contract penalties due to non-fulfillment of them are not common in DBMSs and cloud database services yet [6]. They would cause a MT-DBMS to prioritize tenants, statically or even dynamically. The metric *SLA compliance* can be used to determine MT-DBMS capabilities concerning performance reliability and tenant prioritization. It is calculated as total penalty amount.

B. Experiments

The proposed metrics in conjunction with scheduling of evolved tenant workloads enable an evaluation of the main challenges of MT-DBMS by various kinds of experiments.

Scalability tests: These tests refer to the MT-DBMS performance with increasing load. On the one hand, single-tenant scalability can be measured by increasing a tenant's workload rate in a simulated single-tenant environment and gathering its absolute performance. On the other hand, system scalability can be quantified by continuously increasing the number of active tenants with constant workload rates and mixes. On closer consideration of the performance degradations of active tenants, the fairness of resource distribution can be measured.

Performance isolation tests: These tests estimate how well a MT-DBMS isolates tenant performances from each other. This can be achieved by calculating relative performance of a tenant which runs a static workload while one or more other tenants run a dynamic workload in parallel. Dynamic workloads can be achieved either by changing the transaction rate or its mix over time. Tenants may have equal importance or a MT-DBMS prioritizes them by agreed individual SLAs. Relative tenant performance can be used to assess resource allocation fairness of the MT-DBMS, while SLA compliance evaluates its performance reliability.

Further tests may determine database elasticity like warmup times on workload changes. They can also be used to evaluate database robustness on hardware failures or include costs from the perspective of a provider (cost per tenant) or a tenant (cost of delivered performance, etc.). Detailed explanations of these tests are beyond the scope of this article.

III. MuTeBench

MuTeBench [7] is an open-source framework that allows simple creation of highly diverse benchmarks for a variety of multi-tenant DBMSs and cloud database services. We have implemented it with the purpose of running scalability tests and performance isolation tests, but it is not limited to these experiments only. It provides flexible scheduling of various tenant workloads implementing diverse and evolving usage patterns. With the help of fine-grained statistic gathering, all metrics mentioned in Section II-A can be determined.

Instead of developing a new framework, we decided to extend an existing testbed called OLTP-Bench [8]. We will reason why we regard it as an ideal starting point for a MT-DBMS benchmark framework and point out our extensions resulting in MuTeBench.

A. OLTP-Bench

OLTP-Bench is an open-source benchmarking framework for relational databases. Currently it supports data generation and workload execution of 15 online transaction processing (OLTP) benchmarks consisting of classical OLTP benchmarks such as TPC-C [3], modern web benchmarks like Yahoo Cloud Serving Benchmark (YCSB) [9], generated synthetic micro-benchmarks as well as workload traces of real-world applications like Twitter. Due to central SQL dialect management and the use of standard database drivers, each benchmark can be applied to all major relational DBMSs and cloud database services. OLTP-Bench is able to simulate evolving usage patterns by varying its transaction rate and mix. It is determined in a configuration file in conjunction with connection settings and the number of concurrent worker threads. Controlled by a central workload manager, those threads execute requests in parallel and gather transaction latencies. The results of all workers are finally combined and aggregated for a given time window, providing information about average latency, latency percentiles, and throughput. This client-side database performance monitoring can be brought into accordance with server-side monitoring of its resource consumption. However, OLTP-Bench is not suitable for modeling a challenging large-scale multi-tenancy scenario, among other reasons, because it cannot run a benchmark several times in parallel. [8]

B. Architecture

Due to its features mentioned in Section III-A, OLTP-Bench represents an appropriate benchmarking framework for cloud databases. Hence, we decided to purposefully modify and expand it in order to allow benchmarking of MT-DBMSs. Figure 1 illustrates the resulting architecture based on [8], with added components marked in gray. To simplify updates on future OLTP-Bench versions, we changed its components as little as possible. The most impactful change is an added central controller. It schedules benchmark runs for all tenants according to a scenario description file (see Section III-C). For each benchmark run, it controls existing OLTP-Bench components which are necessary for running a workload or modifying data. The most notable changes of these components are related to enabling concurrent benchmark runs,

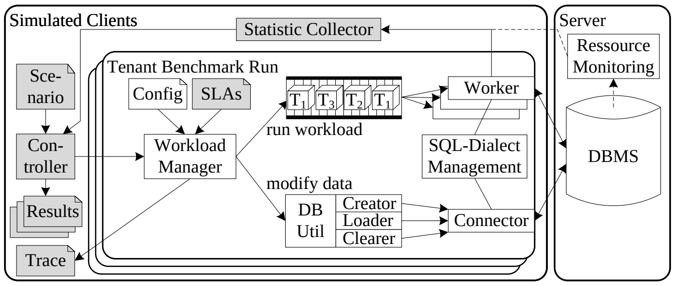


Figure 1: Architecture of MuTeBench

```
1  <event> <!-- Time Format: hh:mm:ss -->
2      <start>00:00:00</start>
3      <repeat>00:05:00</repeat>
4      <stopAfter>00:10:00</stopAfter>
5      <tenantsPerExecution>3</tenantsPerExecution>
6      <firstTenantID>2</firstTenantID>
7      <benchmark>tpcc</benchmark>
8      <actions>create, load, execute</actions>
9      <configFile>tpcc_config.xml</configFile>
10     <slaFile>premium_service.xml</slaFile>
11 </event>
```

Figure 2: Scenario Event Definition

supporting service level agreements, calculating statistics after finishing all benchmark runs, and calculating them both for each tenant and across tenants. Statistics are stored as files and visualized by charts using a graphical user interface (GUI) if requested (see Section III-D).

C. Scenario Description

A scenario description file contains a number of events whose definition is shown in Figure 2. Each event is comparable to an OLTP-Bench run and can be executed by multiple tenants at different times. Event execution times are specified by time of first execution, time of last execution and a repetition time interval (lines 2–4). At any execution time, the benchmark may be performed for several tenants. The associated tenant ID will be incremented for each run, starting from an initial ID (lines 5–6). By Figure 2, benchmark runs for nine tenants will be started. The tenants 2–4 will start immediately, 5–7 after 5 minutes, and 8–10 after 10 minutes. Actions to be executed (table definition, data generation, benchmark execution, script execution, or table truncating) and an universal OLTP-Bench configuration file for the given benchmark have to be defined (lines 7–9). A single configuration file enables tenant specific settings by using wildcards, which will be replaced by the corresponding tenant ID. This enables, for instance, individual workload rates or custom connection settings to support shared machine, shared process, and shared table consolidation (see Section I). Combined with fain-grained workload control of OLTP-Bench, this scheduling flexibility allows running quite diverse large-scale experiments, despite compact definition.

D. Statistics and Service Level Agreements

MuTeBench collects statistics about the absolute performance of each workload run. This is based on OLTP-Bench statistic gathering and includes both performance of each single tenant and overall performance. In addition, relative performance (see Section II-A) may be determined by involving result files of previous tenant baseline runs. Results can be saved as raw data (list of transactions including start times and latencies) and aggregated data by using a given time interval. Furthermore, MuTeBench provides a GUI based on Java Swing and JFreeChart [10] to visualize results as charts.

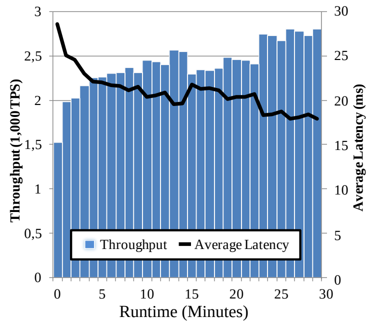
MuTeBench supports tenant-specific SLAs (line 10 in Figure 2). Each agreement is defined for an absolute performance metric (see Section II-A) and a time window like five minutes. Optionally, those agreements can be linked to specific transaction types only. Each agreement may include several service levels, which associate violations of performance targets with a penalty. For example, a service level may predefine a penalty of US$50 if the 99th latency percentile for 'Delivery' transactions is above 50 ms within an interval of five minutes. MuTeBench is able to measure DBMS reliability very conveniently by computing penalty amount for a given tenant or across all tenants. Because of a lack of standards and only marginal DBMS support of performance SLAs, we have developed a SQL extension to forward SLAs to our MT-DBMS prototype for further research purposes.

IV. Experiments

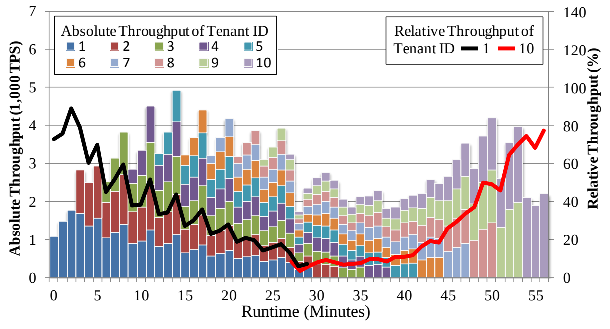
We have performed several experiments to analyze the suitability of MuTeBench for creating various MT-DBMS benchmarks. This article presents the results of two selected experiments using just two machines. Further tests may evaluate database clusters, compare different MT-DBMSs, consider creation, migration, and deletion of tenant data, or evaluate database predictability by using SLA compliance. We used computers of identical construction (Intel Core i7-2600 with 4 cores running at 3.4 GHz, 8 GB of memory, a 7200 rpm hard disk, Debian 4.7.2-4) for client and server, they were interconnected by 1 Gbps Ethernet. For each tenant, 50 parallel worker threads with separate database connections were executed running the YCSB benchmark [9] against MySQL 5.5.31. Due to a shared machine approach, each tenant uses a dedicated database with a size of about 600 MB (YCSB scale factor 500).

During an initial baseline run, a single tenant ran a YCSB workload mix (50% 'ReadRecord' and 10% of each other transaction type) without any rate limitations for 30 minutes. After initial overhead for connection buildup, the performance increased almost steadily due to improved buffer utilization. Figure 3a illustrates the results of this experiment with a maximum performance of about 2,800 transactions per second (TPS) and an average latency of 17 milliseconds at this time.

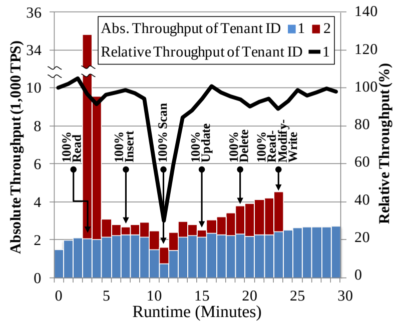
After re-establishing equivalent test conditions, we have performed a system scalability test with 10 tenants, which used the workload profile of the baseline run. They were run one by one at a starting interval of three minutes. The absolute throughput increased up to 4,900 TPS with five active tenants, respectively 250 parallel connections (see Figure 3b). However, the performance decreased with increasing number of connections. The situation was aggravated by exhausting the available memory after 28 minutes. It relaxed again by less active connections near the end of the test. The relative throughput of tenant 1 highlights periodical performance impact by incoming connections of other tenants. Altogether, MySQL distributed resources quite fair among tenants. Performance deviations of active tenants were relatively small at each point in time.



(a) Baseline Run



(b) System Scalability Test



(c) Performance Isolation Test

Figure 3: Experiment results

Another experiment shall give further explanation about tenant performance isolation. Tenant 1 used the workload profile of the baseline run once again, while tenant 2 started an evolving workload mix with unlimited rate after three minutes. At the beginning it only ran the transaction 'ReadRecord' and shifted to the next transaction type by changing the ratio (100:0 $\rightarrow$ 75:25 $\rightarrow$ 50:50 $\rightarrow$ 25:75 $\rightarrow$ 0:100) every minute. This shift was repeated for all YCSB transaction types. Figure 3c shows the tremendous throughput variations of tenant 2. Surprisingly, the achieved 32,758 point queries per second over one minute barely affected the performance of tenant 1. Only range queries resulted in a significant performance impact of tenant 1.

V. Related Work

To the best of our knowledge, *Multe* [5] is the only existing benchmark framework for MT-DBMSs so far. Its purpose is similar to MuTeBench. However, it was not designed for OLTP benchmarks exclusively. It is suitable for real workload simulation just to a limited extent because workload mixes of tenants cannot be changed dynamically and their workload can only be enabled or disabled instead of providing fine-grained rate control. In addition, its statistic gathering is limited and it provides only a sample implementation of a single benchmark for two supported DBMSs so far.

Aulbach et al. [2] presented a MT-DBMS benchmark called *MTD Benchmark*. It simulates an OLTP component of a hosted customer relationship management offering and has been primarily designed to compare the performance of different tenant consolidation approaches by providing schema variability. Hence, its purpose differs from benchmarks built by MuTeBench.

Krebs et al. [11] created a multi-tenancy benchmark to compare multi-tenancy and tenant isolation for dedicated virtual machines on cloud platforms based on TPC-W [3]. Therefore, they address a complete service infrastructure, consisting of web servers, application servers, and database servers.

TPC-VMS [3] requires parallel executions of identical workloads in separate virtual machines, consolidated onto one logical server. It describes an environment with static tenant workloads. By contrast, an intermediate state of TPC-V [12] describes a similar scenario, but with evolving usage patterns and database sizes of tenants. Both benchmarks specify workloads, scenarios, and their environment precisely in order to ensure comparability, while MuTeBench has been designed to perform in a wide range of scenarios.

VI. Conclusion and Future Work

In this article, we summarized our expectations towards capabilities of a MT-DBMS benchmarking framework and reasoned that classical database benchmarks cannot fulfill them. In our opinion, OLTP-Bench represents an ideal basis for such a framework because of its extensibility, portability, statistic gathering, and support of evolving workloads. Our benchmark framework MuTeBench benefits from that and combines it with flexible tenant workload scheduling, SLA support and new metrics to cope with challenges of MT-DBMSs.

However, MuTeBench has some limitations. For instance, compatibility with all OLTP-Bench features is not tested yet and its numerous parallel workers may limit its scalability. Hence, as an extension we plan to properly decompose a given scenario for distributed running a benchmark on several clients and combining their statistics. Transaction-specific SLAs are not yet evaluated and further tests with different database systems and cloud database services are required.

References

[1] D. Jacobs and S. Aulbach, “Ruminations on Multi-Tenant Databases,” in BTW, 2007, pp. 514–521.

[2] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger, “Multi-Tenant Databases for Software as a Service: Schema-Mapping Techniques,” in SIGMOD, 2008, pp. 1195–1206.

[3] Transaction Processing Performance Council, http://www.tpc.org/, retrieved: March 28, 2014.

[4] J. Schaffner and T. Januschowski, “Realistic Tenant Traces for Enterprise DBaaS,” in ICDE Workshops, 2013, pp. 29–35.

[5] T. Kiefer, B. Schlegel, and W. Lehner, “MulTe: A Multi-Tenancy Database Benchmark Framework,” in TPCTC, 2012, pp. 92–107.

[6] W. Lang, S. Shankar, J. M. Patel, and A. Kalhan, “Towards Multi-tenant Performance SLOs,” in ICDE, 2012, pp. 702–713.

[7] MuTeBench, https://github.com/MuTeBench/MuTeBench, retrieved: March 28, 2014.

[8] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudré-Mauroux, “OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases,” PVLDB, vol. 7, no. 4, 2014, pp. 277–288.

[9] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking Cloud Serving Systems with YCSB,” in SoCC, 2010, pp. 143–154.

[10] JFreeChart, http://www.jfree.org/jfreechart/, retrieved: March 28, 2014.

[11] R. Krebs, A. Wert, and S. Kounev, “Multi-Tenancy Performance Benchmark for Web Application Platforms,” in ICWE, 2013, pp. 424–438.

[12] P. Sethuraman and H. R. Taheri, “TPC-V: A Benchmark for Evaluating the Performance of Database Applications in Virtual Environments,” in TPCTC, 2011, pp. 121–135.

Large-Scale Image Processing Research Cloud

Yuzhong Yan, Lei Huang
Department of Computer Science
Prairie View A&M University
Prairie View, TX
Email: yyan@student.pvamu.edu, lhuang@pvamu.edu

Abstract—**We have entered the big data era, where massive data are generated each single day. Most of these new generated big data are images and videos. Besides the fast-increasing data size, the image and video processing algorithms become much more complex, which poses great demands to data storage and computation power. Our image processing cloud project aims to support the image processing research by leveraging the cloud computing and big data analysis technology. In this paper, we present our design for image processing cloud architecture, and big data processing engine based on Hadoop. We also report the performance scalability and analysis on the cloud using several widely used image processing algorithms.**

Keywords–Cloud Computing; Map/Reduce; Hadoop; Image Processing

I. INTRODUCTION

We have entered the so-called big data era, where massive data are generated each single day. Big data are generated by digital processing, social media, Internet, mobile devices, computer systems and a variety of sensors. Most of these new generated big data are images and videos. Big data analytics requires scalable computing power and sophisticated statistics, data mining, pattern recognition, and machine learning capabilities [1]. It is exaggerative in image processing domain since the image and video processing algorithms become more and more complicated, which demands even more power in computation. Some of these image processing requires even real-time processing capability [2]. It is time to rethink if we need to create a domain specific cloud for image processing research in order to meet these challenging requirements.

Image processing research and education are fundamental to support research in many other fields such as medical, oil & gas, and security. It has been widely used in industries. Researchers and students working on the domain are in great need of a high-level programming environment that can utilize the latest, large scale computing resources to speed up their research, since the image data have much higher resolution and the computation are much more sophisticated and intensive than before. The modern computer architectures, however, have evolved to be extraordinarily complex, and frequently becomes a challenge rather than help for general researchers and educators that use image processing technology, which is even equally true for experts in this domain. In order to utilize large scale computing resources to meet the image processing requirements, researchers will face scalability challenges and hybrid parallel programming challenges of creating code for modern computer hardware configurations with multilevel parallelism, e.g., a cluster based on multicore processor nodes. It is not only hard for researchers to implement their algorithms using existing programming environment; but, it is also challenging to them to reuse and share the existing research results since these results are largely dependent on OS, libraries, and underlying architectures.

In order to fill the gap between complicated modern architectures and emerging image processing algorithms for big data, our image processing cloud project aims to produce a high-performance and high-productivity image processing research environment integrated within a cloud computing infrastructure. The cloud will not only provide sufficient storage and computation power to image processing researchers, but also it provides a shared and open environment to share knowledge, research algorithms, and education materials. By leveraging the cloud computing and big data processing technology, our design is to hide the software and hardware complexity from researchers, so that they can focus on designing innovative image processing algorithms, instead of taking care of underlining software and hardware details.

In this paper, we discuss the related work in Section II, and then introduce our image processing cloud architectures in Section III. Further, we describe our experimental image processing applications and their performance analysis in Section IV and Section V, respectively. Last, we will discuss the future work and conclusion in Section VI.

II. RELATED WORK

There are several related work in processing images in parallel using Hadoop platform. The biggest difference between our work and others is that our solution provides a PaaS and supports the multiple languages in implementing image processing algorithms. HIPI [3] is one of them that is similar to our work. In contrast to our work, HIPI [3] creates an interface for combining multiple image files into a single large file in order to overcome the limitation of handling large number of small image files in Hadoop. The input type used in HIPI is referred to as a HipiImageBundle (HIB). A HIB is a set of images combined into one large file along with some metadata describing the layout of the images. HIB is similar with Hadoop sequence file input format, but it is more customizable and mutable [4]. However, users are required to modify the image storage using HIB, which creates additional overhead in programming. In our work, we make the image storage transparent to users, and there is no additional programming overhead for users to handle image storage.

Hadoop Mapreduce for Remote Sensing Image Analysis [5] aims to find an efficient programming method for customized processing within the Hadoop MapReduce framework. It also uses the whole image as InputFormat for Hadoop, which is similar with our solution. However, the work only supports Java so that all mapper codes need to be written in Java. Compared with our solution, he performance is not as good as the ours since we use native C++ implementation for OpenCV.

Parallel Image Database Processing with MapReduce and Performance Evaluation in Pseudo Distributed Mode [6] performs parallel distributed processing of a video database by using the computational resource in a cloud environment. It uses video database to store multiple sequential video frames, and uses Ruby as programming language for Mapper, thus runs on Hadoop with streaming mode same as ours. As a result, our platform is designed to be more flexible and supports multiple languages.

Large-scale Image Processing Using MapReduce [7] try to explore the feasibility of using MapReduce model for doing large scale image processing. It packaged large number of image files into several hundreds of Key-Value collections, and split one huge image into smaller pieces. It uses Java Native Interface(JNI) in Mapper to call OpenCV C++ algorithm. Same with the above work, this work only supports a single programming language with additional overhead from JNI to Mapper.

III. PVAMU Cloud Architecture

The PVAMU (Prairie View A&M University) Cloud Computing infrastructure is built on top of several HPC clusters together. The cloud consists of a virtual machine farm based on Apache CloudStack [8] to provide Infrastructure as a Service (IaaS), and a Hadoop-based high-performance cluster to pvoide Platform as a Service (PaaS) to store and process big data in parallel. Although we describe the entire system in the section, the experiments conducted in the paper were on top of the Hadoop cluster. We integrated the widely-used image processing library OpenCV [9] on the Hadoop cluster to build the image processing cloud. We describe these two major components in the following sections.

Figure 1 shows the Cloud Computing infrastructure developing at PVAMU. The infrastructure consists of three major components: 1) A Cloud center with a large number of Virtual Machines (VM) farm as the cloud computing service portal to all users; 2) A bare-metal high performance cluster to support High Performance Computing (HPC) tasks and big data processing tasks; 3) a shared data storage and archive system to support data access and storage. In this system, the Cloud infrastructure functions as the service provider to meet a variety of users requirements in their research and education. For HPC, the Cloud submits these tasks to the HPC cluster to fulfill their computing power demands. For these high throughput applications, the Cloud will deliver suitable virtual machines from the VM farm to meet their requirements. The Cloud orchestrates all functionalities of the entire system; provide elastic computing capability to effectively share the resources; delivers the infrastructure/platform services to meet users research requirements; supports the big data storage and processing; and builds a bridge between end-users and the complicated modern computer architectures.

A. PVAMU Virtual Machine Farm Cloud

We create a virtual machine farm based on Apache CloudStack on top of an 56 nodes dual-core IBM cluster, and

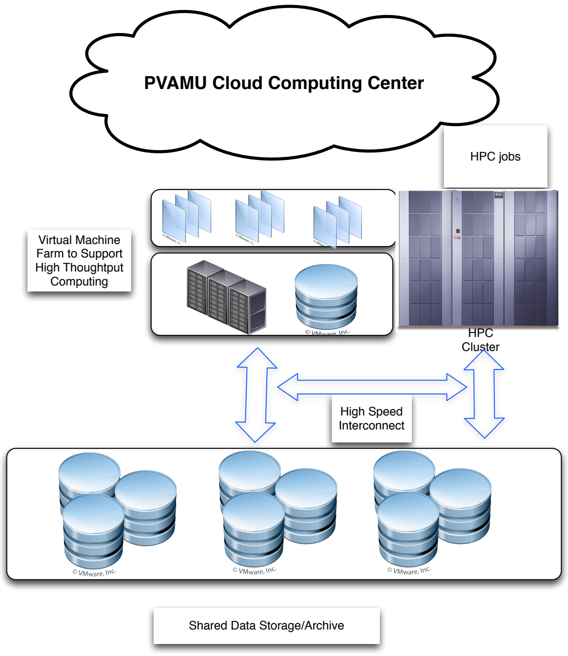

PVAMU Cloud Computing Center
HPC jobs
Virtual Machine Farm to Support High Thoughtput Computing
HPC Cluster
High Speed Interconnect
Shared Data Storage/Archive

Figure 1. PVAMU Cloud and HPC Cluster for Big Data Processing

a new small Dell cluster with three 32 CPU cores servers, and one GPGPU server with 48 CPU cores and 1 NVIDIA Fermi GPGPU. Apache CloudStack is an open source software package that can deploy and manage large number of Virtual Machines to provide highly available, and highly scalable IaaS cloud computing platform. The goals of the PVAMU cloud are to provide IaaS and PaaS with customized services, to share resources, to facilitate teaching, and to allow faculty and students in different groups/institutions to share their research results and enable deeper collaborations. The CloudStack is used to manage users, to handle users requests by creating virtual machines, and allocate resources.

B. Image Processing Cloud

The image processing cloud is built by integrating the image processing library OpenCV with Hadoop platform to deliver PaaS specifically for image processing. The following describes the two major components.

1) Hadoop Cluster: We installed the Hadoop [10] big data processing framework on the bare-metal HPC cluster within PVAMU Cloud to provide PaaS. All experiments presented in the paper are conducted on the Hadoop cluster. The Hadoop cluster consists of one 8-node HP cluster with 16-core and 128GB memory each, and a 24-node IBM GPGPU cluster with 16-core and one Nvidia GPU in each node, and connected with InfiniBand interconnection. We have installed the Intel Hadoop Distribution [11] based on Apache Hadoop [10] software stack, which is a framework that is designed to store and process big data on large-scale distributed systems with simplified parallel programming models. It consists of Hadoop common utilities, Hadoop Distributed File System (HDFS) for high-throughput and fault tolerance data access, Hadoop Yarn for job scheduling and resource management, and Hadoop MapReduce [12] for parallel processing engine based on a simple parallel pattern. Besides its capabilities of storing and processing big data, the built-in fault tolerance feature is also a key to complete big data analytics tasks successfully. The Hadoop cluster is used in our project to handle image and video storage, accessing and processing.

2) OpenCV Image Processing Library: We selected the widely-used OpenCV (Computer Vision) [9] library as the base image processing library integrated with our image processing cloud. OpenCV is an open source library written in C++, and it also has Java and Python interfaces supporting Windows, Linux, Mac OS, iOS and Android. It is optimized and parallelized for multicores and accelerators using OpenCL. We installed the library on the Hadoop cluster to enable image processing capability with MapReduce parallel programming model.

By combining the above two components, we are able to implement a scalable image processing cloud to deliver the capabilities as services to support researchers/faculty/students to conduct their research in image processing domain. In the next section, we present our design and implement of several image processing algorithms in the cloud, and discuss their performance.

IV. Design and Implementation Image Processing Cloud

The goal of our image processing cloud is to deliever PaaS to image processing researchers and developers. It should be able to store large amout of images and videos, as well as be able to process them and meet the performance requirements. Users should be able to work their image processing algorithms using their familiar programming langugaes with very limited knowledge in parallelism. It is a challenge to meet these requirements since image processing researchers use different programming languages in designing and implementing algorithms. The most popular-used programming models include Matlab, Python, C/C++, and Java. In order to meet the multi-language requirement, we cannot rely on native Hadoop Java programming model.

Hadoop platform provides distributed file system (HDFS) that supports large amount of data storage and access. Hadoop MapReduce programming model supports parallel processing data based on the widely-used map-and-reduce parallel execution pattern. In order to support the multiple language requirements in image processing domain, we choose Hadoop streaming programming model by revising standard input and output, and stream data to applications written with different programming languages. Moreover, the streaming model is also easy to debug in a standalone model, which is critical to test and evaluate an algorithm before going to large-scale. To achieve the best performance, we choose C++ in our underlining library implementation to keep the optimizations as much as possible.

The image processing application execution environment with MapReduce on Hadoop is shown in Figure 2. On the left side, a large number of images are stored in HDFS, which are distributed across the cluster with 128MB as one block. These images are split by Hadoop MapReduce engine with customized InputFormat, and are distributed to large number of mappers that execute image processing applications to the assigned images. The results may be merged by the

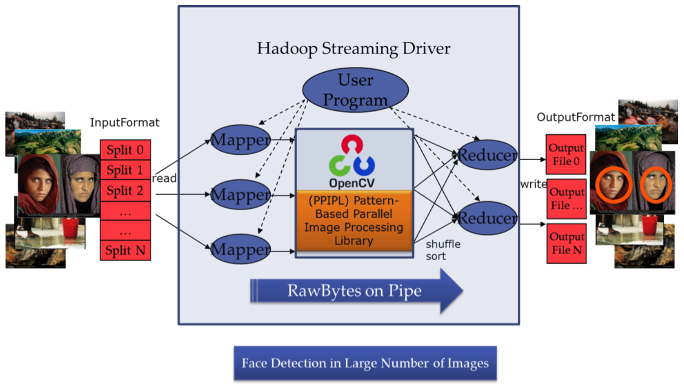


Figure 2. Image Processing Execution Environment with MapReduce

reducer that exports the results to customized OutputFormat class to finally save the outputs. Since large amount raw data are transferred among split, mappers and reducers, it is very important to keep data locality to minimize network traffic. All mappers are launched on the node where the processed images are physically stored.

A. InputFormat

The main challenges of performing image processing on Hadoop are how to split data split and how to implement customized mappers. In Hadoop streaming mode, the input data need to be processed by InputFormat class at first, and then pass to each mapper through the standard input (Stdin). The InputFormat class in Hadoop is used to handle input data for Map/reduce job, which need to be customized for different data formats. The InputFormat class describes the input data format, and define how to split the input data into InputSplits buffer, which will be sent to each mapper. In Hadoop, another class RecordReader is called by mapper to read data from each InputSplit.

Depending on the image or video size, we implemented two different InputFormat classes to handle them. For still image processing with many individual image files, the InputFormat class is straightforward. It simply distributes these images to mappers by each image file since they are smaller than block size of Hadoop system. For the mass individual image files, ImageFileInputFormat extends FileInputFormat, which return false in isSplitable and create ImageFileRecordReader instance in getRecordReader. ImageFileRecordReader will creates Key/Value pair for mapper and read whole content of input image file actually.

For the big video file, it needs to be split and to be sent to the mapper for processing. There are different video file containers; in this project only MPEG transport stream file is considered to simplify split implementation. TSFileInputFormat is used for parsing the MPEG transport stream, and for generating split information including offset in video file and the hostname which will process the related split, and create TSFileRecordReader in the getRecordReader function. TSFileRecordReader will create Key/Value pair for mapper and read the section data from input video file, then pass it to mapper for processing.

B. *Mapper and Reducer*

Most of work for programming in Hadoop is to divide algorithms into Mapper and Reducer, and embed and implement them in them respectively. In Hadoop streaming mode, the main difference with other modes is the I/O processing in Mapper and Reducer. Both Mapper and Reducer could only get Key/Value from Stdin and output results through Stdout. A common I/O class named CommonFileIO was designed to handle different type data sources, including normal local files, Stdin/Stdout and HDFS file on Hadoop. The frequently used file system interfaces were provided, such as open, read/write and close and more. We implement our own Mapper and Reducer as independent image processing applications with input and output handled by Stdin and Stdout. By using Hadoop streaming model, we are able to launch these image processing applications as large number of Mappers or Reducers that execute in parallel.

C. *OutputFormat*

OutputFormat class in Hadoop describes the output-specification for a Map-Reduce job. It sets the output file name and path and creates the RecordWriter instance, which is passed to Map/Reduce framework and writes output results to file. For the image processing with small files, OutputFormat is unnecessary and the intermediate results could to be stored on HDFS directly. But for big video file, different applications will output different results. We have implemented several OutputFormat templates for reducer jobs. For example, to get the Histogram of whole file, it needs to accumulate each result of Reducer in OutputFormat; while for the template matching application, it needs to save each matched result and give a summarization in OutputFormat.

D. *Results*

As a result of our implementation, the image processing cloud is able to handle image processing algorithms written with multiple lanuages, including Matlab, Python, C/C++, and Java, which is the major contribution comparing with other related work. Moreover, the cloud provides scalable performance by keeping the native C++ implementation of OpenCV library internally, and takes the data locality into consideration in the task scheduling strategy. The next section discusses the performance experiments using three typical image processing algorithms.

V. EXPERIMENTS

We choose three widely-used image processing algorithms including Discrete Fourier Transform (DFT) [13], face detection, and template matching to conduct performance and programming experiments on our image processing cloud. Our images are downloaded from Internet public photos, including Google images, National geographic photo gallery, and Flickr public photos. The Fourier Transform algorithm is one of fundamental and most-widely used image processing algorithm that transforms data from spatial domain to frequency domain to facilitate more advanced image processing algorithms. The 2D DFT are frequently applied to digital image files in many image processing algorithms. Face detection on image and video is very useful in many areas, such as surveillance and

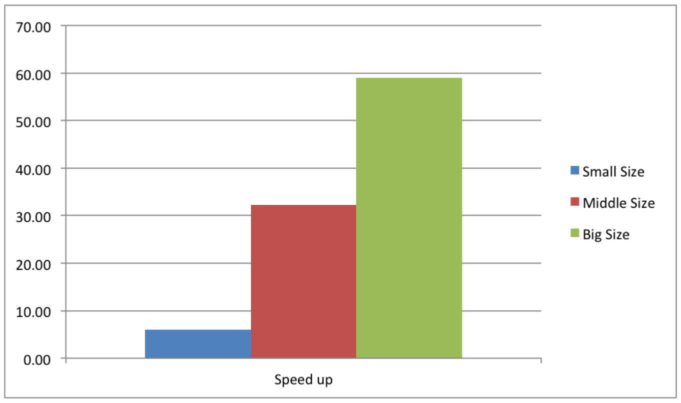


Figure 3. Face Detection Program Speedup on Hadoop

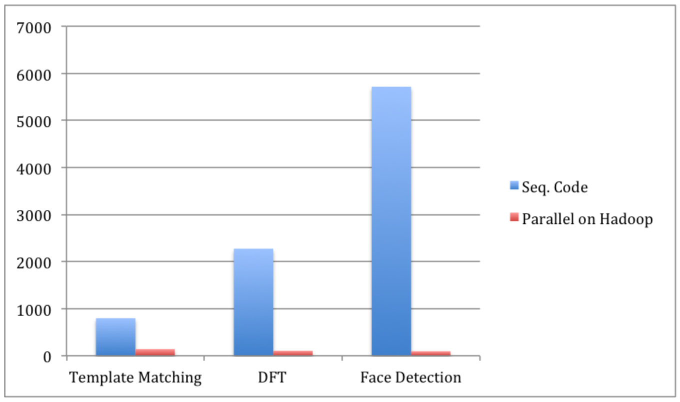


Figure 4. Execution Time of Different Algorithms Apply on Big Size Images

entertainment equipment. The feature-based cascade classifiers provide a practical face detection algorithm; it is popular used but still need much computation for multi-images or video file. Template Matching could find the location of small template image in big image files, which is one core function in machine vision. These three algorithms are implemented on top of OpenCV, and apply them to three groups of images as test pattern. These three groups of images are separated to the small size group with images less than 1MB, the middle size group of images from 1M to 5MB, and the big size group of images from 5MB to 30MB. The experiments are conducted on our small HPC cluster with 8 HP nodes, 16 cores and 128GB memory each. In this cluster one is mater node for jobtracker and the other seven are worker nodes for computation, so we have total 112 cores. Table I shows the face detection program execution time for both sequential and Hadoop MapReduce parallel execution.

TABLE I. FACE DETECTION PROGRAM EXECUTION TIME FOR THREE GROUPS OF IMAGES

	Small Size Images with 5425 Files/594MB	Middle Size Images with 2539 Files/3621MB	Large Size Images with 400 Files/4436MB
Sequential codes	1386.02s	4511.35s	5716.31s
Parallel on Hadoop	228s	140s	97s

Figure 3 shows the face detection speedup of the three groups of images comparing with sequential execution. With

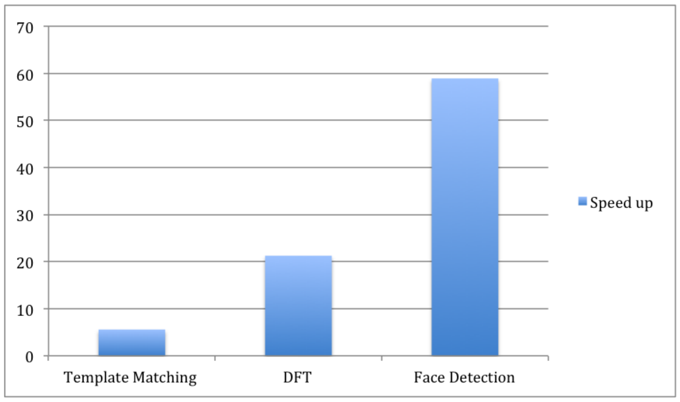


Figure 5. Speed up of Different Algorithms Apply on Big Size Images

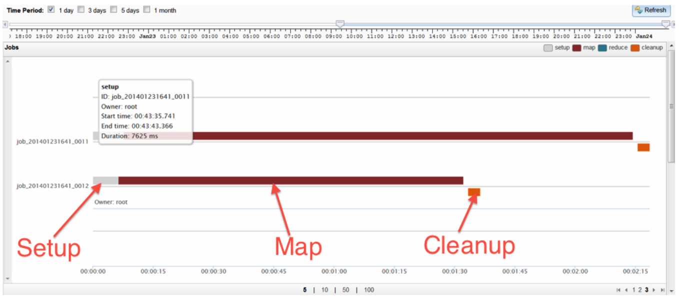


Figure 6. The job running results on Hadoop

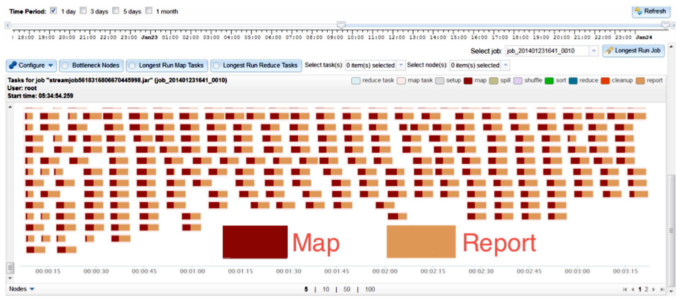


Figure 7. The small image processing job profiling results on Hadoop

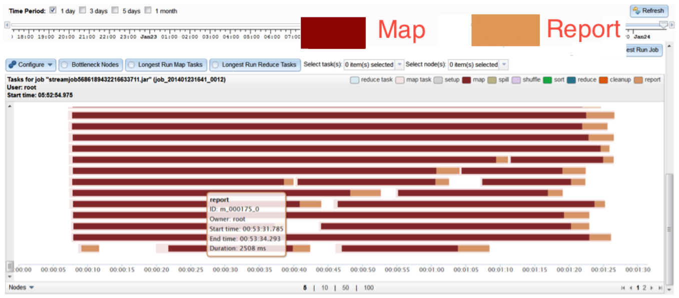


Figure 8. The big image processing job profiling results on Hadoop

8 nodes and 16 cores each, the experiments show a maximum 59 times speedup for big images. Apparently, the image size determines the speedup due to the I/O processing time. In Hadoop big data framework, it is designed to achieve better performance with big files and computation intensive programs. The max number of simultaneously running mappers could reach up to max CPU cores in the cluster. The small images need to be aggregated into big files to improve the performance. Figure 4 shows the execution time of different algorithms applying on big size image files. Figure 5 shows the speed up factors of different algorithms applying on big size image files. The face detection need more time to execute on single process, but could get best speed up on Hadoop platform.

The test program could be divided into three main parts: input and output, decode and encode, image processing algorithm. The total execution time of sequential codes is the sum of all images processing time. We can use the following formula to represent it.

$$Ts = (Ti + Td + Ta) \times N \tag{1}$$

Here, *Ti* is the image reading and writing time; *Td* is the image decoding and encoding time and *Ta* is the algorithm executing time.

While running on Hadoop with only mapper, the total execution time is composed of:

$$Th = Tp + (Tm + Tr) \times (N \div C) + Tc \tag{2}$$

Here, *Tp* is the job preparing/setup time for Hadoop job; *Tm* is the average mapper executing time, which is nearly equal to *(Ti + Td + Ta)*; *Tr* is the average mapper report time and *Tc* is the job cleanup time. In our execution environment, the *Tp* is about 8s and *Tc* is about 3s while running job as shown in Figure 6.

The profiling results of small size image pattern and big size image pattern are shown in Figure 7 and Figure 8. For the small size image pattern, assuming the average *Tr* is 2.5s, and get the average execution time of one image from sequential code execution result, which is 0.2555s, the ideal speed up factor on Hadoop system could be estimated by:

$$S = \frac{1386.02}{8 + (0.2555 + 2.5) \times \lceil \frac{5425}{112} \rceil + 3} = \frac{1386.02}{147.2} = 9.4$$

For the big size image size pattern, assuming the average *Tr* is 2.5s, and get the average execution time of one image from sequential code execution result, which is 14.3s, the ideal speed up factor could be estimated by:

$$S = \frac{5716.31}{8 + (14.3 + 2.5) \times \lceil \frac{400}{112} \rceil + 3} = \frac{5716.31}{87.65} = 65$$

Considering the overhead between mappers, the estimated results is close to our experimental results in the big size of images case, which is the ideal speed-up by considering the data movement, task startup and cleanup overheads. In order to get better performance on Hadoop, we need to reduce these overheads. One possible solution is to further improve the split function to determine a good number of mappers based on the number of available nodes, and reduce overloads of mappers startup and cleanup. The improvement will be explored in the future work.

The Hadoop system has good robustness and scalability. Comparing with the traditional MPI program, MapReduce programs are able to complete jobs even one or more computing nodes in a cluster are down. New nodes could be added into the Hadoop system at runtime to meet dynamic requirements, thus get better performance in most cases and provide elastic computing as needed.

VI. Future Work and Conclusion

At the first stage of the project, our main goal is to explore the feasibility and performance of using Hadoop system to process large number of images, big size of images or videos. From our experimental results, Hadoop is able to handle these problems with scalable performance. However, there are also some issues need to be considered and addressed in future work.

The first issue is the problem of data distribution. As stated in the previous section, Hadoop is good at handling big data. The speedup is not apparent while trying to process many small images scattered across multiple nodes. Even the SequenceFile could not solve this problem efficiently. Our next plan is trying to store image files in HBase [14]. HBase could handle random, realtime reading/writing access of big data. We expect to improve performance and increase the flexibility with new solution on HBase.

The second issue is that Hadoop is not good at handle low-latency requirement. Apache Spark [15] is a fast and general-purpose cluster computing system. Because of the in-memory nature [16] of most Spark computations, Spark programs can better utilize the cluster resources such as CPU, network bandwidth, or memory. It can also handle pipeline, which is frequently used in image processing. In next step, we will try to move to Spark platform, and evaluate the performance of the experimental groups on Spark platform.

Another main goal of this project is to make it easy for users processing image using cloud computing platform. Most of users are not familiar with cloud platform, such as algorithm experts or even common users; they all have requirements of big data processing. In the next stage, a Domain Specific Language (DSL) for image processing and friendly user interface will be provided. Users could utilize the powerful platform with only limited knowledge on Cloud and use DSL to simplify their programming efforts.

Acknowledgment

This project is supported in part by National Science Foundation CRI program award #1205699. Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] J. C. Brian Dolan and J. Cohen, “MAD Skills: New Analysis Practices for Big Data,” in Very Large DAta Bases(VLDB) 09. Lyon, France: ACM, Aug. 2009.

[2] C.-I. C. Hsuan Ren and S.-S. Chiang, “Real-Time Processing Algorithms for Target Detection and Classification in Hyperspectral Imagery,” IEEE Transactions on Geoscience and Remote Sensing, vol. 39, no. 4, 2001, pp. 760–768.

[3] “Hadoop image processing interface,” http://hipi.cs.virginia.edu/, [Retrieved: January, 2014].

[4] L. L. Chris Sweeney and J. L. Sean Arietta, “HIPI: A hadoop image processing interface for image-based map reduce tasks,” pp. 2–3, 2011.

[5] M. H. Almeer, “Hadoop Mapreduce for Remote Sensing Image Analysis,” International Journal of Emerging Technology and Advanced Engineering, vol. 2, 2012, pp. 443–451.

[6] K. K. Muneto Yamamoto, “Parallel Image Database Processing with MapReduce and Performance Evaluation in Pseudo Distributed Mode,” International Journal of Electronic Commerce Studies, vol. 3, no. 2, 2012, pp. 211–228. [Online]. Available: http://www.academic-journals.org/ojs2/index.php/ijecs/article/viewFile/1092/124

[7] K. Potisepp, “Large-scale Image Processing Using MapReduce,” Master’s thesis, Tartu University, 2013.

[8] “Apache CloudStack website,” http://cloudstack.apache.org, [Retrieved: January, 2014].

[9] “Open Source Computer Vision,” http://www.opencv.org/, [Retrieved: January, 2014].

[10] “Hadoop Introduction,” http://hadoop.apache.org/, [Retrieved: January, 2014].

[11] “Intel Distribution of Hadoop,” http://hadoop.intel.com/, [Retrieved: May, 2014].

[12] J. D. S. Ghemawat, “MapReduce: simplified data processing on large clusters,” in Communications of the ACM - 50th anniversary issue: 1958 - 2008, vol. 51. ACM New York, Jan. 2008, pp. 107–113.

[13] C. S. B. Thomas W. Parks, DFT/FFT and Convolution Algorithms: Theory and Implementation. John Wiley & Sons, Inc. NY, USA, 1991.

[14] “Apache Hadoop database, a distributed, scalable, big data store,” http://hbase.apache.org/, [Retrieved: January, 2014].

[15] “Spark Lightning-fast cluster computing,” http://spark.incubator.apache.org/, [Retrieved: January, 2014].

[16] M. Z. Mosharaf Chowdhury and T. Das, “Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing,” in NSDI’12 Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. San Jose, CA: USENIX Association Berkeley, Apr. 2012.

Hybrid Cloud Architecture for Software-as-a-Service Provider to Achieve Higher Privacy and Decrease Security Concerns about Cloud Computing

Paul Reinhold and Wolfgang Benn
Chemnitz University of Technology
Chemnitz, Germany
Email: {paul.reinhold@s2012, wolfgang.benn@informatik}.tu-chemnitz.de

Benjamin Krause and Frank Goetz
Qualitype GmbH
Quality Management Systems
Dresden, Germany
Email: {b.krause,f.goetz}@qualitype.de

Dirk Labudde
Hochschule Mittweida
University of Applied Sciences
Mittweida, Germany
Email: dirk.labudde@hs-mittweida.de

***Abstract*—Security and privacy concerns are still major issues during the adoption of cloud services. Software service developers face new challenges to solve this problem. To establish the software-as-a-service and to address privacy and security concerns of customers, providers can use the suggested hybrid cloud architecture to outsource the data persistence layer. By this approach, neither customer nor provider have to trust an arbitrary public cloud service provider. The approach offers a tradeoff between higher privacy and security for less flexibility and scalability in consideration of costs and therefore its application in practice. Test results of the implemented prototype demonstrate the practical suitability, but show the limitations as well. Besides focusing on functional requirements like privacy and scalability, also non-functional demands such as independency from special software or hardware needs and the minimal migration effort, have been considered.**

***Keywords*-Hybrid Cloud; Privacy; Cloud Security; Architecture; Software-as-a-Service; Key Management**

I. Introduction

The increasing knowledge about cloud computing technology and its publicity leads to a growing number of service offerings over the Internet. Even small and medium sized companies are able to offer services for a large number of consumers through cloud computing concepts. Resources can be obtained easily from public cloud providers like Amazon Web Services [1], without limits regarding scaling and flexibility. Instagram is a typical example for modern cloud based application services [2]. Many of these cloud services are provided to the end users via the Internet in a form of Software-as-a-service (SaaS). SaaS can be understood as web or mobile applications with variable degree of complexity consumer can use on demand. For further information see [3].
The high acceptance of these services suggests that private consumers have a lower privacy demand than business users. The study of [4] indicated that in Europe, and especially in Germany the acceptance of public cloud services for business purposes is low. Typical reasons are security and privacy concerns. Companies do not want their critical business documents or customer data they manage in public clouds. In addition, the study showed that experiences with private cloud computing are, with 83%, mainly positive. It has to be pointed out that the size of the enterprise has a great influence on its experience with cloud computing. For instance, 60% of large companies already have private cloud experience, while small and medium-sized enterprises (SME) are more reserved.
Out of these concerns, we stress aspects of how a provider can develop and run a SaaS in which SaaS consumers get their privacy needs satisfied. Consumers might gain a higher acceptance to cloudbased SaaS so that cloud computing gets more attractive to SME-SaaS providers. Therefore, we propose a hybrid cloud architecture enhanced with an additional architecture layer between business logic and persistence layer. It has a minimal migration effort and reveals no information, except for meta data, about the outsourced data to the public cloud provider. For evaluation purposes, we implement a prototype using the suggested architecture to securely outsource unstructured (files) and structured data (databases) in a public cloud. The results show that our suggested architecture is very practical and an efficient/effective way for SaaS providers to use some of the advantages of public clouds.
The outline for the rest of the paper is as follows. Section II discusses a comparison of three common cloud delivery models with respect to privacy, costs and performance. In Section III, we describe the proposed hybrid cloud architecture. Section IV describes the implemented prototype, performance tests and resulting overheads. Section V provides a critical discussion of the results and gives an overview for future work. Section VI concludes the paper.

II. Cloud Model Comparison

Our work focuses on SME-SaaS providers, which already run SaaS offerings or web applications and take new cloud offerings into account to become more cost-efficient. In addition, SaaS providers probably use own hardware to run their SaaS and plan to develop a new version or new service, which exceeds the current limit of their hardware. Another scenario could be that the provider needs to invest in new hardware to keep its services running and is looking for lower cost alternatives.
The end-user of a cloud service shall be named cloud consumer or simply consumer [5]. In the consumers view, the provider offers SaaS over the Internet. Whether the service offered by provider's hardware or by third party resources is irrelevant for the customer, as long as service supply is ensured. However,

TABLE I. COMPARISON OF DIFFERENT CLOUD MODELS FROM CONSUMERS AND PROVIDERS POINT OF VIEW

View	Criteria	Private Cloud	Public Cloud	Hybrid Cloud
consumer	cost	high	low	medium
	privacy	medium	low	high
	data-at-rest encryption	yes	yes	yes
	key owner	provider	provider	consumer (and provider)
	key management	provider	provider	provider
	compliance	medium	low	medium - high
	governance	medium	low	medium - high
provider	cost	high	low	medium
	availability	medium	very high	high
	backup	medium	very high	very high
	hardware needs	high	very low	medium
	effort to run service	very high	low	medium - high
	achieve cost-efficiency	very difficult	easy	possible
	flexibility	high, but limited	very high	higher, but limited
	scaling	yes, but limited	yes	yes, but limited

the method of providing can be essential for the acceptance of the SaaS on consumers side.

According to the common cloud delivery model by Mell and Grance [6], the provider can run the SaaS in a private, public or hybrid cloud. In the private cloud, the provider runs its own cloud. He owns the hardware and has exclusive access to it. For cost-efficiency reasons the provider runs its hardware in form of a cloud, allowing flexibility and scaling effects. The public cloud is the most flexible and cost-efficient service, since the provider obtains resources and pays by use. The hybrid cloud is the third approach were the provider runs the service both in a private and public cloud.

Table I shows a comparison between the three cloud service models from both consumer's and provider's point of view. The costs factor is comprehensible and hardware expenses are usually passed to consumers. Privacy is low for public and medium for private cloud architecture. Private clouds often have strong authorization and access control concepts, but no special requirement to secure data with encryption against the provider itself [7]. Thus, the cloud provider often has access to customer data and their customers data, respectively. In a public cloud it is very costly or impractical to secure data and to keep them available for processing at the same time, like fully homomorphic encryption [8] can provide.

Another important aspect for consumers is data-at-rest encryption. Encryption is possible in all of the models, but strongly connected with key ownership and management. If the same instance encrypts data and stores the referring key, no trustable security can be guaranteed, because providers can encrypt data without users knowledge or permission. This has been recently documented for economically rational cloud providers [9]. Because of this circumstance the hybrid cloud model suggests a solution where consumers get more control over their data and the possibility for public cloud providers to access unencrypted files is eliminated. Compliance and governance depend directly on this solution.

If consumers care about their data security and privacy, the hybrid cloud model supposed to meet the needs best. Even if a consumer trusts its provider (with a private cloud) and consequently encryption is not needed, the hybrid solution is more economical.

A private cloud cannot provide the high availability, that is guaranteed by a public cloud. The hybrid model benefits from this fact by outsourcing parts of the architecture in a public cloud. Backup security underlays the same principle, in fact the backup process in hybrid model can be outsourced completely. The hardware needs and the effort to run the service are coherent. Lots of own hardware means not only to manage, but also to maintain and have environment settings (buildings, redundant broadband internet access) to run a private cloud. Because of this, it is much easier to create a cost-efficient SaaS in a public cloud than in a private cloud. The great advantages of cloud computing like flexibility and scaling are limited in private and hybrid cloud solutions. As a result of this comparison and questions, our aim is to combine the security of a private cloud with the flexibility, reliability and availability of a public cloud, creating a balanced solution. The hybrid approach offers a trade-off between increased security for decreased efficiency. We want to know if its worth doing this trade-off. In addition, questions we want to answer are:

- Is it possible to set up a practical solution, which does not reveal any information of the data, except for metadata, like size or structure?
- Is a support for both, unstructured data such as files and structured data such as databases, possible?
- Is it possible to do so with very low migration effort, to keep the acceptance high?

III. HYBRID CLOUD ARCHITECTURE

The here proposed privacy-enhanced hybrid cloud architecture is illustrated in Figure 1. It applies typical security concepts in the cloud computing field using tier, logic and data partitioning described by Bohli et al.[10]. In contrast to the study of [10], we do not spread our tiers to various, non-collaborating cloud providers. As mentioned before, our approach is spread over a private and public cloud, performing all critical tasks in the private cloud and outsource only the data tier in public cloud. This makes it unnecessary to label tasks or data as critical in a manual or semi-automatic manner, like described by Zhang et al. [11]. The Figure 1 shows the data flow from the consumer via the SaaS provider to the public cloud. For simplification just ingoing traffic is displayed. The consumer uses a computer with Internet connection to access the SaaS (1). In addition, the consumer has a master key for encryption purposes. The initial login and identification

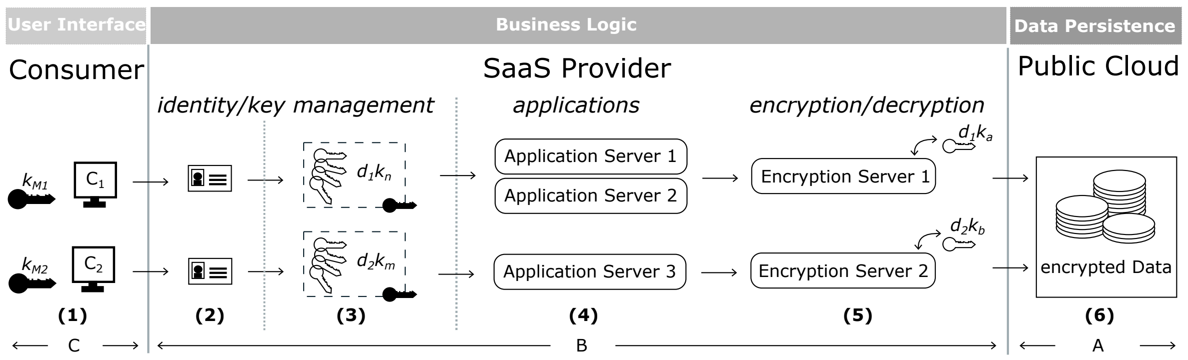


Figure 1. The hybrid cloud architecture for SaaS with an enhanced business logic layer by key management and encryption layer. Lines separate the actors and architecture layers. Dashed lines illustrate encrypted keys. Dotted lines represent a physical separation. Rounded and rectangular boxes represent virtual servers and data, respectively. A,B and C show the range of the thread scenarios.

procedure should be located on physically separated hardware or be outsourced to a trusted ID verification provider [12] (2). The key management system, storing encryption keys, is another security critical resource and should not be integrated in the private cloud (3). The private cloud structures contain the application server layer (4) and the encryption server layer (5). All communication pass these tiers, so they have to be highly scalable. The public cloud provider is illustrated in form of a persistence layer (6). The outgoing data flow differs slightly. The consumer's request is received directly by an application server, which asks on its part for the data. This request received by the encryption servers, sending a key request to the key management system. The resulting request to the cloud is encrypted by the encryption server with the received data key. The received data from the public cloud is decrypted with the same data key and afterwards send to the application server, that passes the data to the consumer.

A. Key Concept

The key concept is shown in Table II. Besides the hybrid cloud architecture, a hybrid encryption concept is used to provide a better performance. The master key is persisted by consumer and used to encrypt and decrypt the data keys. The data keys are persisted by the provider and used to encrypt and decrypt consumer's data. The transfer key pairs are generated during the customers registration and used for secure exchange of a temporary copy of the consumers' master key.

TABLE II. OVERVIEW OF EXISTING ENCRYPTION KEYS IN THE PROPOSED KEY CONCEPT.

Key	Type	Owner
master key k_M	symmetric	Consumer
data key d_xk_y	symmetric	SaaS Provider
transfer key k_T	asymmetric	Consumer/Provider

B. Security Overview

To evaluate the security of the architecture,we examined/constructed three threat scenarios that are indicated with A, B and C in Figure 1. All data traffic from the consumer to the SaaS provider and *vice versa* is encrypted through standards like Transport Layer Security (TLS) [13]. The consumer authenticates against the SaaS provider by multi-factor or strong authentication. After a successful login, the consumer allows the provider to decrypt the data keys d_xk_y with its master key k_M. The data keys allow decrypting the data stored in the public cloud. The master key is solely persisted by the user. This prevents the SaaS provider to decrypt data from not logged in consumers. As a result, the consumer gains high control over the data. The public cloud provider only stores encrypted files and never gets access to encryption keys. Neither the consumer nor SaaS provider must trust the public cloud provider in this scenario.

1) Threat Scenario A: This scenario describes an attack against the public cloud provider. Even if the attackers have full (or physical) access to resources of the cloud provider, all data of the consumer are secure. The data stored by the public cloud will never be in plaintext, so the privacy and confidentiality of the data is guaranteed. This requires an appropriate encryption by the SaaS provider and careful handling of the corresponding encryption keys. Depending on the encryption techniques the SaaS provider uses, the public cloud provider can obtain meta information about the stored data such as size, structure or access pattern. In the simplest case, only data confidence can be provided by the SaaS. To secure integrity and availability, the SaaS provider can mirror and distribute files over different non-cooperation public clouds. The higher level of security results in higher costs for the SaaS provider and therefore for the consumer. So, this scenario gives in part of information revealing, a positive answer to the first question in Section II.

2) Threat Scenario B: Threat scenario B describes the offered security if the SaaS provider is attacked. The attacker gets no access to consumer data, but to the data keys of active consumers. Accessing data of the inactive user is impossible, even if the attacker has full access. The SaaS provider does not persist the consumers master key; therefore, it's no possibility for the attacker to decrypt the data keys. The security of logged out consumers is still guaranteed.

3) Threat Scenario C: Threat scenario C describes an attack against the consumer. If the attacker obtains the login credentials, factors and the master key, he can get full access to the consumer's data. To prevent the attacker getting easy access to other consumer's data, the SaaS provider should be multi-tenancy capable. In detail, application servers of different consumers should be at least virtually separated.

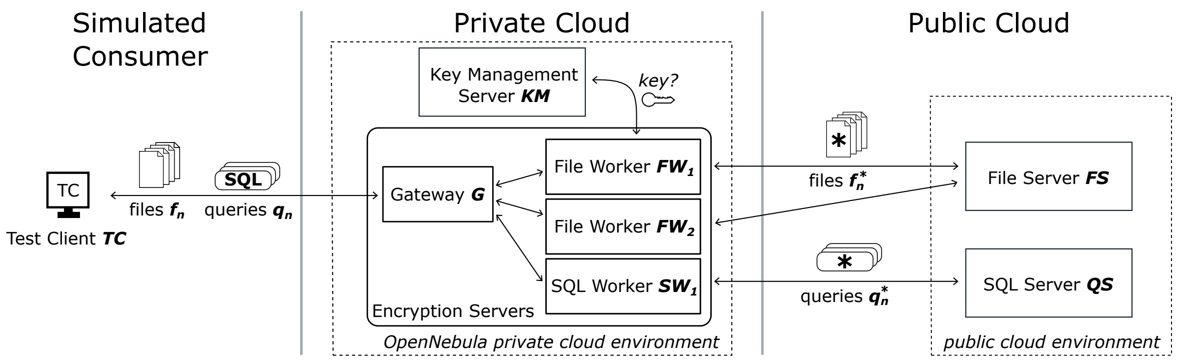


Figure 2. The implemented encryption server prototype as one core component of the proposed hybrid cloud architecture.

IV. Implemented Prototype

The developed prototype implements the encryption servers as one core component of the hybrid architecture. It is run in a private cloud environment based on OpenNebula 4.4 [14] powered by four physical hosts with 3 GHz Dual-Cores and 8 GB RAM. These machines consist of standard components to keep the hardware costs low.
Gateway G, File Worker FW_i, SQL Worker SQ_i and Key Management KM are virtual machines (VMs) and are part of an autoscaling service called *OneFlow*[15]. Figure 2 illustrates a minimal test setup with one gateway, three worker and the key management VMs. Load balancing is performed by the JBoss *mod_cluster 1.2.6* [16]. Therefore, G is a mod_cluster enabled Apache http-Server [17]. The implemented prototype is completely developed in Java. At the moment, the supported communication protocols are http and AJP [18]. The Test Client TC is able to simulate clients, which can imitate the behavior of application server(s) and consumers, respectively. That means, these simulated clients create http POST requests for file uploads and GET request for file downloads to test the very basic functionalities of file processing in a SaaS. The clients send also SQL Queries to test the database capability. The File Worker VMs FW_1, FW_2 were developed using Java Servlet and deployed on a JBoss AS7 Application Server [19]. To enable the servlet using different encryption methods, the installed JBoss AS7 was extended by a security provider module of Bouncycastle [20]. Files are encrypted with AES (256Bit) [21] in the private cloud and then uploaded to a public cloud server. WebDav [22] is used to provide a file server in the public cloud. The file servers is hosted by an European IaaS provider. The SQL Worker based on an CryptDB enabled mysql proxy server described by Popa et al. [23]. For storing this data a common MySQL database extended by CryptDB user defined functions is also hosted by the IaaS provider. Both, virtual server use minimal resources of 1 GHz with 1 GB RAM.

A. Test Setup

For the shown test results in Figure 3 and 4 the Test Client (TC) simulates four clients. Three clients, started with a delay of 20 s, send files, while one client sends SQL queries. Two file clients work after the following patterns. $ABABA$, where A stands for upload, download and delete (UDD) 12 files of 1 MB with a delay of two s and B stands for UDD 12 files of 1 MB with a delay of 10 ms. The third file client executes a CDC pattern, where C stands for UDD 5 files of 10 MB with a delay of 5 seconds and D stands for UDD 3 files of 10 MB with a delay of 10 ms. As long as these patterns are not finished yet, the SQL client repeats sending queries in form of 15 inserts, 10 selects and 15 deletes. To complete the test protocol, the simulated clients take 10 min and 23 s. Figure 3 shows the VM workloads of G and F_1, F_2 in 20 s intervals.

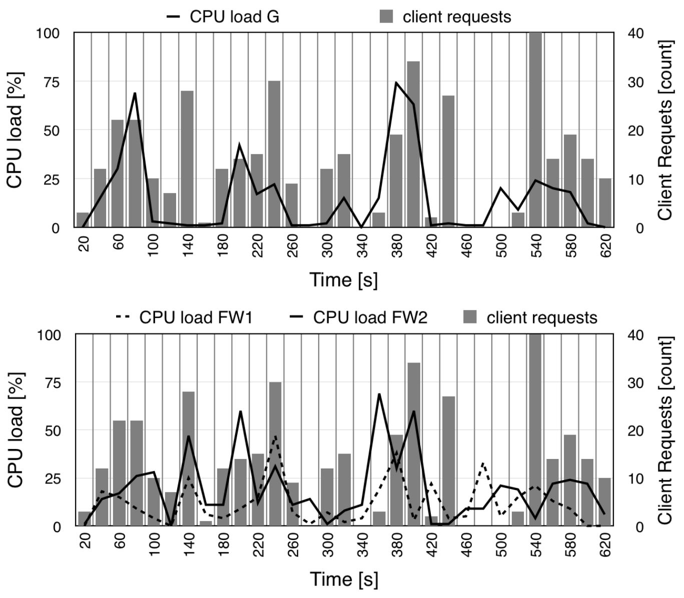


Figure 3. CPU load of G (0.2 vCPU, 512 MB RAM) FW_1, FW_2 (0.25 vCPU, 2 GB RAM) VMs depending on the number of client requests.

The load balancing metric configured in mod_cluster config in JBoss nodes combines CPU load, system memory usage and amount of outgoing/incoming requests traffic. Figure 4 shows the response times for processing the SQL queries.

B. Test results

The configured load balancing metric works very well, as shown in Figure 3. Especially, the timespan between 340 and 420 s is remarkable. The gateway recognize the high load of FW_2 sending client requests to FW_1. At time intervals of 380 s, it is inverse.
Figure 4 shows interesting results of the 18 rounds the SQL client executes its *'send 15/10/15 queries'* protocol. Although, the median of the response times is promising, there are lots

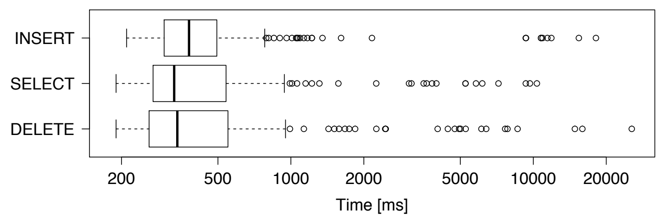


Figure 4. Logarithmic scaled response times for processing 234 INSERT, 180 SELECT and 234 DELETE queries. Whiskers maximums are 1.5 IQR.

of outliers. With regard to the logarithmic scale, a response time of ten to twenty seconds for a very basic query are unacceptable for practical use. Our best guess is the CryptDB proxy doing some internal recovery and key management operations. However, we bind the relevant folder via NFS to our key management KM, because all data in VMs is volatile, some file operations via the internal network should not last that long.

Table III displays our test results for different encryption algorithms, showing that AES works most efficient. For this reason, we use AES for encryption in the file workers.

In addition to the above-mentioned test, we have run some long term tests to get some impressions of efficiency and overall overheads. First of all, Figure 5a shows the percentage of times for encryption, communication with KM, and upload files to the cloud. It illustrates only four percent used for encryption, the rest of time the file worker are, roughly speaking, waiting. The same can be seeing on Figure 5b. It has to be noted that, these long term tests were done by one simulated client uploading, download, and deleting a 1MB file with a delay of 10 seconds. There is no waiting time, it is just the fact, as can be seen in Table III, that the encryption/decryption times compared to upload/download times are so small.

The overheads can be seen in Figure 6. In fact, the overhead to upload and download a file is around 51% and 28%, respectively. The difference can be explained by the required effort to store the encryption key and is also illustrated in Figure 5a. The overhead to delete a file is with around 126% very high. It is explainable with the additional roundtrip to the key management to delete the stored encryption key.

TABLE III. UPLOAD, DOWNLOAD, ENCRYPTION, AND DECRYPTION AVERAGE TIMES $\bar{t}$ IN SECONDS

encryption method	$\bar{t}_{up}$	$\bar{t}_{enc}$	$\bar{t}_{down}$	$\bar{t}_{dec}$
AES (265 bit)	1040.9	39.9	1116.4	51.45
DESede (168 bit)	1165.9	167.18	1239.4	135.83
Serpent (256 bit)	1180.9	57.18	1138.2	57.92
Twofish (256 bit)	1195.9	50.55	1160.4	50.45
CAST6 (256 bit)	1300.9	53.27	1037.6	40.09

V. DISCUSSION AND FUTURE WORK

Our test results showed that our enhanced hybrid cloud architecture is reasonable and applicable. With the statement of thread scenerio A, the first question can be answered positively, setting up the base for the more interesting questions two and three. Our proposed approach supports both, unstructured and structured data. However, our tests of the integration of the work of [23] show that database encryption is much more complicated than file encryption. This can be concluded from the fact that database encryption is not only a matter of data-at-rest encryption, but computation under encryption as well.

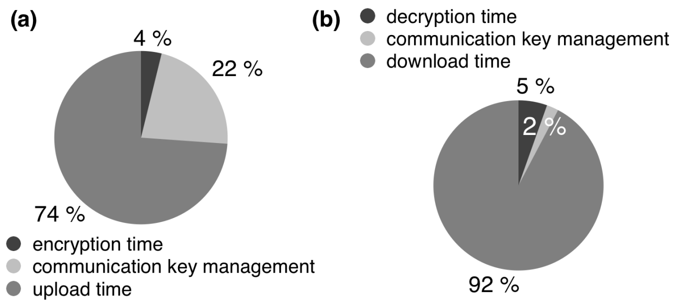


Figure 5. Percentage of time to encrypt and upload (a), decrypt and download (b), respectively.

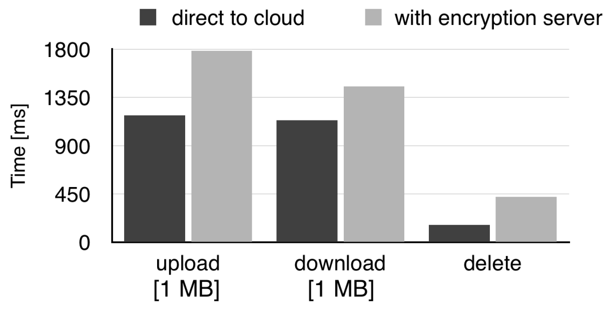


Figure 6. Diagram of average times to upload, download, and delete files with and without encryption

However, the support of databases is limited in a way not all queries can be supported; details can be seen in [24]. As a result the answer of question two is positive. Our test results confirm the mentioned fact in [23] that the implementation of CryptDB is highly prototypical. As the own implementations of Google and SAP show [25], more development effort - e.g. towards full support of JDBC - is necessary. As Figure 6 shows the overheads for file encryption are acceptable. Even the high overhead in deleting files, considering, the response time is still under half a second, the usability of the SaaS would be influenced in a very small way. An integration with low migration effort, as question three asks, is very realistic. In fact, as the gateway behaves like a file server/ database, the only change will be switching from old servers to the gateway server.

Figure 5 points out that the encryption workload of file worker is not high. This leaves space for additional functionalities like file compressing, for faster up- and downloads, or file indexing for possible searches over the encrypted files. The latter is mentioned in [26]. Also, the integration of a secure identity and key management system, e.g. Kerberos [27], is required to provide a SaaS solution with focus on the customers privacy. Attribute-based encryption concepts like [28][29] could be an interesting option for open questions like: How to integrate SaaS access rights in the key management system.

Moreover, the tests show that implementations of failure and backup routines are absolutely necessary. Despite, the different focus in this first implementation, we want to point out that security is not only about protecting data from unauthorized access or viewing, but also issues of auditing, data-integrity, and reliability should be concerned too. These points will be addressed in future works.

VI. Conclusion

This paper described a hybrid cloud architecture model for SaaS providers with special consideration of service consumers' privacy and security aspects. Section II compares the three models, private, public, and hybrid cloud from which SaaS providers can choose. The private cloud model is the most inflexible and cost intensive option. Probably being an option for large companies owning much hardware resources, it is not suitable for SME-SaaS providers. The public cloud model is the preferred choice for SaaS providers if the application has no particular high privacy or security requirements. Especially for private end-user applications the public cloud model is cost efficient with low time to market and high scalability. Public cloud services can be recommended to start-ups because of the low investment costs and great flexibility. The hybrid cloud architecture offers compromise for multiple reasons. First, the solution addresses SME with experience in SaaS and own hardware infrastructure. Second, this model offers a higher security level and lowers privacy concerns of consumers. Albeit the cost efficiency is not as high as for the public model, it is clearly higher than for the private model. Despite these advantages, the hybrid approach incurs efficiency penalties in form of a trade-off between increased security for decreased efficiency, flexibility and scalability of public cloud solutions. Besides developing cost-efficient hybrid and secure SaaS solutions, it is highly complex and needs lots of expertise. The hybrid model offers significantly improved security compared to a public cloud architecture and neither the consumer nor the SaaS provider have to trust the public cloud provider. Of course, the consumer has to trust its SaaS provider. However, this is more reasonable than to trust a public cloud provider with an obscure number of third parties.

The prototype includes scalable and flexible encryption servers, a minimal key management system, a public file and database server. Test results showed that a hybrid architecture SaaS extended with encryption servers is a practical solution. In addition, the results illustrated that on-the-fly encryption and decryption is not only a matter of fast encryption methods, but a matter of high network throughput as well. To be applicable in productive systems, improvements of performance and more research are necessary.

The implemented prototype shows that the suggested hybrid architecture is a first step to achieve a higher acceptance of cloud-based SaaS, where providers address the consumers' concerns of privacy and security.

Acknowledgment

The published work has been financially supported by the European Social Fund (ESF) and the EU. We would like to thank the anonymous reviewers for helpful comments.

References

[1] AWS. Amazon web services. [Online]. Available: http://aws.amazon.com [retrieved: April, 2014]

[2] Instagram. Instagram website. [Online]. Available: http://instagram.com [retrieved: April, 2014]

[3] P. Buxmann, T. Hess, and S. Lehmann, "Software as a service," Wirtschaftsinformatik, vol. 50, no. 6, 2008, pp. 500–503.

[4] BITOM and KMPG, "Cloud-monitor 2013 cloud-computing in deutschland – status quo und perspektiven," KMPG Study, Februar 2013.

[5] F. e. a. Lui, "Nist cloud computing reference architecture," National Institute of Standards and Technology, Tech. Rep., 2011.

[6] P. Mell and T. Grance, "The nist defintion of cloud computing," National Institute of Standards and Technology, Tech. Rep., 2011.

[7] M. A. Rahaman. How secure is sap business bydesign for your business. [Online]. Available: http://scn.sap.com/docs/DOC-26472 [retrieved: April, 2014]

[8] C. Gentry, "A Fully Homomorphic Encryotion Scheme," Ph.D. dissertation, Stanford University, 2009.

[9] M. a. a. van Dijk, "Hourglass schemes: How to prove that cloud files are encrypted," in Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012, pp. 265–280.

[10] J.-M. Bohli, N. Gruschka, M. Jensen, L. Lo Iacono, and N. Marnau, "Security and Privacy Enhancing Multi-Cloud Architectures," IEEE Transactions on Dependable and Secure Computing, 2013, pp. 1–1. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6487347

[11] K. Zhang, X. Zhou, Y. Chen, and X. Wang, "Sedic : Privacy-Aware Data Intensive Computing on Hybrid Clouds Categories and Subject Descriptors," 2011, pp. 515–525.

[12] D. Hühnlein, G. Hornung, H. Roßnagel, J. Schmölz, T. Wich, and J. Zibuschka, "Skidentity–vertrauenswürdige identitäten für die cloud," DA-CH Secur, 2011, pp. 296–304.

[13] T. L. Security. Tls documentation. [Online]. Available: http://tools.ietf.org/rfcmarkup/5246 [retrieved: April, 2014]

[14] OpenNebula. Opennebula website. [Online]. Available: http://docs.opennebula.org/4.4/release_notes/ [retrieved: April, 2014]

[15] ONEFlow. One flow doc. [Online]. Available: http://docs.opennebula.org/4.4/advanced_administration/application_flow_and_auto-scaling/appflow_configure.html [retrieved: March, 2014]

[16] ModCluster. mod-cluster website. [Online]. Available: http://www.jboss.org/mod_cluster [retrieved: April, 2014]

[17] Apache. Apache 2.2 website. [Online]. Available: http://httpd.apache.org/docs/2.2/en/ [retrieved: April, 2014]

[18] AJP. Ajp website. [Online]. Available: http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html [retrieved: April, 2014]

[19] JBossAS7. Jboss website. [Online]. Available: http://www.jboss.org/jbossas/ [retrieved: April, 2014]

[20] Bouncycastle. bouncy castle website. [Online]. Available: http://www.bouncycastle.org/java.html [retrieved: April, 2014]

[21] J. Daemen and V. Rijmen, The design of Rijndael: AES-the advanced encryption standard. Springer, 2002.

[22] WebDAV. Webdav website. [Online]. Available: http://www.webdav.org [retrieved: April, 2014]

[23] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB : Protecting confidentiality with encrypted query processing Accessed Citable Link Detailed Terms CryptDB : Protecting Confidentiality with Encrypted Query Processing," 2011.

[24] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," in Proceedings of the 39th international conference on Very Large Data Bases. VLDB Endowment, 2013, pp. 289–300.

[25] C. impact. Cryptdb website impact section. [Online]. Available: http://css.csail.mit.edu/cryptdb/#Impact [retrieved: April, 2014]

[26] S. Kamara, C. Papamanthou, and T. Roeder, "Cs2: A searchable cryptographic cloud storage system," Microsoft Research, TechReport MSR-TR-2011-58, 2011.

[27] Kerberos. Keberos documentation. [Online]. Available: http://tools.ietf.org/html/rfc4120 [retrieved: April, 2014]

[28] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007, pp. 321–334.

[29] C.-P. A.-B. Encryption. Advanced crypto software collection website. [Online]. Available: http://hms.isi.jhu.edu/acsc/cpabe/ [retrieved: April, 2014]

An Approach for Hybrid Clouds Using VISION Cloud Federation

Uwe Hohenstein, Michael C. Jaeger, and Sebastian Dippl
Corporate Technology, Siemens AG
Munich, Germany
Email: {Uwe.Hohenstein, Michael.C.Jaeger, Sebastian.Dippl}@siemens.com

Enver Bahar
Corporate Technology, Siemens AG
Munich, Germany
Email: bahar.enver@gmail.com

Gil Vernik and Elliot K. Kolodner
IBM Research - Haifa
Haifa, Israel
Email: {gilv, kolodner}@il.ibm.com

Abstract—**Hybrid clouds combine the benefits of a public cloud and on-premise deployments of cloud solutions. One scenario for the use of hybrid clouds is privacy: since public clouds are considered unsafe, sensitive data is often kept on premise. However, other non-sensitive resources can be deployed in a public cloud in order to benefit from elasticity, fast provisioning, or lower cost. In this work, we present an approach for hybrid cloud object storage based on the federation of storage resources. It uses the metadata of the objects and containers as a fundamental concept to set up and manage a hybrid cloud. Our approach extends an existing scheme for implementing federation for object storage developed by the VISION Cloud project.**

***Keywords*-Cloud storage;hybrid;federation.**

I. INTRODUCTION

Cloud storage refers to the broader term of cloud computing that represents a novel provisioning paradigm for resources. The National Institute of Standards and Technology (NIST) defines "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1].

Cloud storage as one specific cloud service applies the major characteristics of cloud computing to storage, which are a) virtually unlimited storage space, b) no upfront commitment for investments into hardware and software licenses, and c) pay per use [2]. Depending on the viewpoint, a couple of other characteristics may also become relevant, e.g., support for multi-tenancy in order to serve multiple customers at the same time.

Moreover, the term cloud storage covers a wider area than provisioning approaches and models; cloud storage refers to software implementing storage services. Most notably, *Not only SQL* (NoSQL) database servers are a specific cloud storage solution that gained popularity recently in this context [3]. The understanding about NoSQL databases is that corresponding database servers or services follow a different approach than the traditional table-model provided by relational database servers, implied by an adaptation to distributed systems and cloud computing environments, e.g., by relaxing the Atomicity, Consistency, Isolation, Durability (ACID) characteristics of the traditional (relational) database servers by means of *Basic Availability, Soft state, Eventual consistency* [4] (BASE).

A lot of cloud storage solutions such as NoSQL databases can be used even in a non-cloud computing manner. In such a setup, a storage server can be deployed similar to a traditional database server managed on premise. Of course, the advantages of cloud storage are then partially lost: virtually unlimited storage space is limited by the storage hardware provided by the own platform; investments for such hardware must be taken. The software must be set up and the setup must be planned in advance in contrast to flexible provisioning and pay-as-you-go models. Therefore, such a deployment makes sense, if a software system has to be installed locally while taking advantage of specific characteristics, e.g., of NoSQL technology. In fact, there are a number of motivations for keeping critical data on premise, on private servers rather than utilizing public cloud storage offerings:

- Data storage cannot be delegated because of regulatory certifications. For example, data store that contains legally relevant material could be subject to possible confiscation and thus provisioning of such data cannot be delegated.
- There are often privacy constraints. For example, a data store that holds employees' invention disclosures before being submitted to patent offices might not be suitable to be placed at a public cloud provider.
- A cloud provider offers a certain Service Level Agreements (SLA) or reliability, which is insufficient. In this case, a private on-premise proprietary storage solution may be the choice for keeping critical data.

As pointed out above, the disadvantage of using private on-premise storage solutions opposed to public cloud storage is obvious. In particular, it is likely less flexible and potentially more expensive in terms of cost, since an on-premise solution typically cannot always benefit from the same economy of scale that can be achieved by a public provider.

In this work, we show how to benefit from both worlds by integrating on premise storage services for critical data with public cloud storage service for non-critical data. Through this approach, flexibility and potential cost advantages as well as high SLA requirements can be achieved as required for the individual data entities.

We present our metadata-based approach for a hybrid cloud based on the cloud storage federation scheme developed by the European funded VISION Cloud project [5][6]. VISION Cloud aims at developing next generation technologies for cloud object stores including content centric access to storage. It offers first-class support for metadata for the storage entities, i.e., objects and containers, and enables management functionality based on such metadata, for example, describing and managing federation through container and object metadata.

In order to achieve a hybrid scenario, combining public and private storage, we also use container and object metadata to describe the federation setup. Such an approach to describing federation provides a unified and location-independent access interface, i.e., transparency for data sources, while leaving the federation participants autonomous.

The remainder of this work is structured as follows: Section II explains the VISION Cloud software that is relevant and is used for this work: the concept, particularly of using metadata, the storage interfaces, and the storage architecture. The hybrid cloud approach of VISION Cloud is then presented in Section III. We explain our approach, particularly the architectural setup in Section IV, and continue in Section V with further useful federation scenarios. Section VI is concerned with related work. A brief evaluation of this approach is explained in Section VII. This work ends with Section VIII providing conclusions and future work.

II. The VISION Cloud Project

The EU project VISION Cloud [5] is developing a cloud storage system that allows for the efficient storage of different types of content. The approach supports storage for objects (videos, etc.) together with metadata describing their content. An increasing number and a variety of applications exist that envisage a growing need for such an object store. New media authoring applications are currently arising with video files such as Ultra High-Definition and 4K resolutions. Archives for virtual file systems of virtual machines occur in the area of virtualization and cloud computing. And finally, mobile users use their smart phones for producing and capturing multimedia content in an exponentially growing manner [7].

These all are examples where we expect an increasing number of large and unstructured storage objects. The VISION Cloud project intends to provide an object storage system that is capable of handling large objects and files. Another goal of VISION is the ability to easily ingest content of different types, to analyze the content and enrich its metadata, and to smoothly access the content through a variety of end user devices. This functionality extends the plain data storage features offered by today's cloud storage providers.

A. The VISION Concept of Metadata

A common way to handle content is to put it into files and to organize it in a hierarchical structure. This enables navigating the hierarchy in order to finally find a particular item. However, it becomes more and more difficult to set up an appropriate hierarchy that provides flexible search options with acceptable access performance and intuitive categories for ever increasing amounts of data.

Thus, the target of the VISION Cloud project is to provide an appropriate basis for organizing objects including content-centric access facilities. In contrast to public cloud offerings such as Amazon S3, Microsoft Blob Service, or specific hardware appliances, VISION Cloud puts emphasis on supporting metadata flexibly and making metadata an integral part of the storage system [5][6]. Moreover, VISION Cloud supports private cloud installations and does not require any specialized hardware.

The approach of content-centric storage (CCS) does not restrict the user to organizing his content in hierarchies. Rather, the user describes the content through metadata allowing him to access the content based on its associated metadata. Moreover, an additional layer of the storage system derives metadata from usage statistics or access mechanisms. The basis for this approach to content-centric storage are efficient mechanisms to automatically create or ingest and retrieve metadata about the content. Then, this is the entry point to objects.

To illustrate the intention and scope, consider YouTube as an example: a video (i.e., an object) on YouTube can possess a set of metadata associated with it, metadata of different characteristics. Some metadata remains static (e.g., the uploader of a video), other is dynamic (for instance, the number of views). Some metadata is related to content (e.g., categories applied to the video) and other is of technical nature (e.g., the resolution of the video). YouTube enables one to access the video by searching metadata. Hence, a user can ask for a video that is 'most viewed' or that belongs to the category 'drama' [8].

VISION Cloud offers similar functionality in a more general form. In particular, it extends the scope to any type of data, no matter whether videos, text, pictures, or documents in any form and from any source. While handling and using metadata in a conventional data object storage system is typically restricted to storing and retrieving metadata together with the object, VISION Cloud provides the ability to update and append new metadata to objects as well as the ability to find objects based on their metadata values and their relationships to other objects.

B. Interface

The basic data model of VISION Cloud is similar to the Cloud Data Management Interface (CDMI) [9] standard interface for access and distinguishes between containers and objects. CDMI defines a standardized way to store objects in a cloud. The standard covers create, retrieve, update, and delete (CRUD) operations by defining simple requests with the HyperText Transfer Protocol (HTTP) according to the REpresentational State Transfer (REST) principle. In this model, the underlying storage space is abstracted and primarily exposed using the notion of a container.

Containers are similar to buckets in other storage solutions. A container is not only a useful abstraction for storage space, but also serves as a grouping of the data stored in it. The storage creates, retrieves, updates, and deletes each object as a separate resource. Containers might be organized in a hierarchical manner. The following REST examples give an impression about the CDMI-based interface of VISION: `PUT /CCS/MyTenant/MyContainer` creates a new container for a specific tenant $MyTenant$. Then, `PUT /CCS/MyTenant/MyContainer/MyObject` can be used store an object into this container.

The metadata is passed as content or payload of a HTTP PUT request. The first question arises how to distinguish the container from an object. By the CDMI standard, this is solved by using HTTP header fields indicating a data type for this request. A full request for creating a new container thus looks as in Figure 1:

```
Example: PUT /CCS/MyTenant/MyContainer
X-CDMI-Specification-Version: 1.0
Content-Type: application/cdmi-container
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept: application/cdmi-container
{   metadata : { key1 : value1, key2 : value2 }   }
```

Figure 1: HTTP PUT Request

III. HYBRID CLOUD APPROACH

The goal of cloud federation is to provide a unified and location-independent access interface, i.e., a transparency of data sources in various clouds of different providers, while leaving the federation participants autonomous. Thus, creating a federation of existing clouds supports the client with a unified and combined view of storage and data services across several providers and systems. There are several approaches to provide federation:

1) The first one is to put a new federation layer on top of the clouds to be federated. This means that every access to the federation must be passed to this federation layer. The federation layer might offer additional services such as distributed queries. In any case, each of the federation members remains accessible by its own interface.
2) Instead of introducing a new layer, each federated cloud can be an access point to the federation, i.e., can accept requests. If the cloud itself is unable to answer the request, it delegates the request or parts of it to respective clouds.
3) Controlling clouds and storage location can be part of a language in the sense of a multi-database approach [10]. That is, operations can be performed on storage locations that are explicitly specified, e.g., by means of wildcards expressions.

In the VISION Cloud project, approach (2) is pursued [11]. A federation is accessible from any cloud in the federation. The development of the VISION Cloud project mainly provides two mechanisms that can be used as a base for federations:

1) *Object storage with use of metadata:* In VISION Cloud, all objects are allowed to contain user-definable metadata items that can be used in several ways to query for objects inside the cloud storage system. It is possible to employ a schema for these metadata items to enforce the existence of certain metadata fields and hence enforcing a certain structure.
2) *Adapters for storage clouds:* VISION Cloud includes an additional layer for integrating cloud storage systems, e.g a blob storage service of some larger public cloud offering.
3) *A fine granular Access Control List (ACL) infrastructure:* VISION Cloud was designed with security in mind and provides fine granular access control that are attachable to tenants, containers and objects. If containers are taking part in a federation, the administrator of a federation can provide new ACLs for objects that are being moved in a federation.

Based on the existing work carried out by the VISION Cloud project, this work uses the concept of federation to define a hybrid cloud setup for leaving critical data on premise, while using the metadata processing facilities of the CCS functions of VISION Cloud. The approach consists of two main parts: the administration and setup of a hybrid cloud and its operation.

A. A Hybrid Cloud Setup Based on VISION Cloud Federations

First, there is an administrator for the setup. The administrator is responsible for creating and maintaining a federation of two cloud systems. We propose a way to introduce the clouds to each other by a data structure that contains *metadata* (rather than the data itself) about remote data. In VISION Cloud, a federation is defined between two data containers. In order to create a federation, the administrator has to send a data structure describing the federation to one of the containers taking part in the federation. A typical federation administration data structure in VISION Cloud, for example for an Amazon S3 member, uses the payload [11] in Figure 2:

```
"federationinfo": {
 // information about target cloud
 "eu.visioncloud.federation.status": "0",
 "eu.visioncloud.federation.job_start_time": "1381393258.3",
 "eu.visioncloud.federation.target_cloud_type": "S3",
 "eu.visioncloud.federation.target_container_name":
    "example_S3_bucket",
 "eu.visioncloud.federation.target_region": "EU_WEST",
 "eu.visioncloud.federation.type": "sharding",
 "eu.visioncloud.federation.is_active": "true",
 "eu.visioncloud.federation.local_cloud_port": "80",
 // credentials to access target cloud
 "eu.visioncloud.federation.target_s3_access_key":
   "AKIAIHSZSHAHVWZEAZTGWJOBQ",
 "eu.visioncloud.federation.target_s3_secret_key":
   "2glBUIdO3qQUTLoCeBxTrYoxYzqgV5A2us/Hcd+p",
 "eu.visioncloud.federation.status_time": "1381393337.72"   }
```

Figure 2: Sample Payload

This specification mainly describes the data required for accessing a member's cloud storage. This structure allows one to store some specific information about the clouds such as public/private cloud URIs, container names in both private and public containers, etc. Such a specification creates a link between the clouds and enables certain tasks as data sharding and querying among them. For hybrid scenarios, the federation type can now be chosen as `sharding`. Upon completion of the container linkage process, both private and public containers become aware of each other.

To manage federation, a REST Service is available in VISION Cloud providing the basic CRUD operations (create / read / update / delete) to administer federation instances over standard HTTP commands and to handle these structures. PUT creates a new federation instance by passing an id (in the Uniform Resource Identifier (URI) and the federation info in the body. GET gives access to a specific federation instance and returns the federation progress or statistical data. A federation specification can be deleted by DELETE. Finally, all federation instances can be listed with `GET /{tenant}/federations/`. The result will be an array of federation URIs. For details please refer to the project deliverable [12].

This concept of federation is also applied to the creation of the hybrid setup placing a similar request, although the implementation is different as explained later.

B. Operation of the Hybrid Cloud Setup

The second role involved in an hybrid cloud scenario is the client who wants to access the data in the new cloud system. The client now is provided with a unified view of the data that resides in both the private and public cloud and becomes ready to shard data among the clouds. Any CRUD operation will work on both of the clouds, and all sharding operations are completely abstracted from the client. The operations of the hybrid cloud setup is completely transparent for the client of a container, and the client might even not be aware where the data he accesses resides.

IV. ARCHITECTURE

Our main focus is put on hybrid clouds, i.e., using public and private clouds at the same time for an application. The goal is to provide a uniform access to several autonomous clouds each hosting a cloud store. Moreover, we want to benefit from the recent VISION Cloud architecture as much as possible.

In general, a hybrid cloud has to tackle heterogeneity of the units to be combined. In the context of storage federation, there are several types of heterogeneity. At first and most obvious, each cloud provider such as Amazon, HP, IBM, or Microsoft has management concepts and Application Programming Interfaces (APIs) of its own, which are currently proprietary, despite some emerging standards, e.g., CDMI [9]. And then at the next lower level, the federation has to take into account the heterogeneity of data models of the cloud providers.

The hybrid cloud should provide an abstraction over the individual storage clouds. This means on the one hand that a unified interface is offered for all the clouds and thus supporting query features without knowing what data is available in what cloud. On the other hand, the hybrid cloud should provide means to control the placement of storage items. Since we here focus on distributing data over private and public clouds according to their confidentiality, each item (object) must have a confidentiality level associated with it in order to allow for storing it in a corresponding cloud.

In fact, the implementation of the content-centric storage service (CCS) of VISION Cloud helps to handle heterogeneity by allowing us to wrap homogeneous units, each with a CCS interface. An instance of a VISION Cloud CCS sits on top of a single storage system. However, multiple underlying storage system types are supported by CCS due to an adapter architecture that accommodates multiple storage interfaces. Currently CCS adapters are available for the proprietary VISION Cloud storage service, Amazon S3, CouchDB, MongoDB, and the Windows Azure Blob storage. That is, the CCS architecture supports multiple cloud providers, as long as a storage adapter is provided.

The CCS directly connects to a storage server's IP and port number, either referring to a single storage server or to a load balancer within a cluster implementation. If we put CCS on top of each storage server, the CCS would have to manage all the distribution, scalability, load balancing, and elasticity. This would tremendously increase the complexity of CCS. Moreover, CCS would re-implement features that are already available in numerous cloud storage implementations. All of the currently supported storage system types have a built-in cluster implementation. Among the CCS candidates, CouchDB has an elastic cluster implementation named BigCouch. MongoDB has various strategies for deploying clusters of MongoDB instances. And Windows Azure Blob storage, as well, is a distributed environment. To our knowledge and published material by the vendors, we can assume that these cloud systems are able to deal with millions of customers and tens of thousands of servers located in many data centers around the world.

Hence, our decision was to put CCS on top of these cluster solutions due to several benefits. CCS is just a bridge between the load balancer and the client. All scalability, elasticity, and partitioning is done by the storage system itself. Therefore, there is no need for CCS to deal with scalability, elasticity, replication, or duplication within the cloud. Since there is only a single CCS instance for each type of cloud storage, consistency issues are also handled internally.

A. Technical Implementation

The implementation of the federation service of VISION Cloud is not used for the hybrid cloud setup. Instead, the service has been technically implemented in the layer that provides the content-centric storage functions (the "CCS"). In fact, in order to enable the hybrid cloud in the CCS, several extensions have been made to the CCS: A new *ShardService* has been added to CCS the task of which is to intercept requests to the CCS and decide where to forward the request. The ShardService implements a reduced CDMI interface and plays the key role to shard in hybrid environments. As already mentioned, we give the right to the client to determine data confidentiality. This selection is done through the data creation process and a metadata item should indicate data confidentiality.

Having researched several sharding mechanisms, we chose to implement *Key Based Partitioning*. Key based partitioning provides a perfect match to our needs by using a metadata key to define the sharding strategy for separating public and private clouds. We enable clients to create some keywords to define the confidentiality such as *confidential*, for example. If there is such a keyword among the metadata, then this data will be directed to the private cloud. Otherwise, it will be directed to the public cloud. The impact of keywords on sharding the data can be specified in the federation.

B. Scenarios

The following examples should illustrate the approach. A simple PUT request mechanism works as follows in the ShardService:

1) The metadata of the object is checked for an item indicating confidentiality such as *confidential* : *true*.
2) Then the metadata of the container is fetched (not the object) to obtain the connection information for both clouds.
3) If the data is confidential, the private cloud's connection information is used. The ShardService connects to this cloud and forwards the request to it.
4) If data is not confidential, then the public cloud's connection information is used to send the request to this cloud.

A GET request through the ShardService behaves differently, since it does not necessarily contain a specification regarding which cloud contains the object. Therefore, every GET request is sent to all clouds participating in the federation:

1) As soon as the GET request arrives in the cloud, the container metadata is requested to gather the

connection information about the private and public clouds.
2) The ShardService connects to both clouds and run the same GET request for each of them in parallel.
3) The results of requests are combined and the result is sent back to the client.

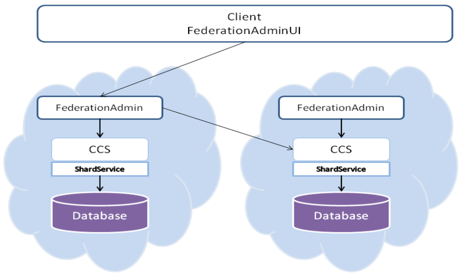


Figure 3: Federation creation

The entire key-based sharding principle can be explained with an example. Beforehand, we assume we have two different containers in two different clouds. Let us name these two containers as *vision1* and *vision2*. Additionally, our two clouds are addressed by these URLs respectively: *vision-tb-1.myserver.net* and *vision-tb-2.myserver.net*.

First, we need to make the two clouds aware of each other, to be more precise, we need to federate them. A sketch of the process can be seen in Figure 3. By using the newly introduced ShardService and its HTTP API, we do a PUT request with the payload of Figure 4.

```
http://cloud_url:cloudport/MyTenant/sharded/vision1
{  "target_cloud_url" : "vision-tb-2.myserver.net",
   "target_cloud_port" : "8080",
   "target_container_name" : "vision2",
   "local_container_name" : "vision1",
   "local_cloud_url" : "vision-tb-1.myserver.net",
   "local_cloud_port" : "8080",
   "type" : "sharding",
   "private_cloud" : "vision1",
   "public_cloud" : "vision2"   }
```

Figure 4: Federation payload

The above JSON string is a sample of our data component. We need to mention that a full dataset contains information regarding the private and public cloud types, urls, users, authorization information, etc. Upon completion of the request, the two clouds vision-tb-1 and vision-tb-2 enable sharding at the container level. From now on, *vision-tb-1* will be our private cloud and *vision-tb-2* will be our public cloud. Such a specification is needed for any container to act as a shard (cf., the local target_container_name).

The PUT request must also be sent to the second cloud, however, with an ”inverted” payload. This is done implicitly.

Since the two clouds are federated, we can now perform data CRUD operations in a sharded way. In order to store confidential data, we need to perform the request in Figure 5.

```
PUT vision-tb-1.cloudapp.net:8080/CCS/siemens
  /vision1/newObject
{  "confidential" : "true"  }
```

Figure 5: PUT request for storing confidential data

Figure 6 shows that the ShardService decides to store *newObject* in the *private* cloud which is *vision-tb-1*. Note that there is no need to indicate the access to the federation as part of the PUT request.

If the data is not confidential we can replace `"confidential" : "true"` with `"confidential" : "false"` in Figure 5 (see also Figure 7):

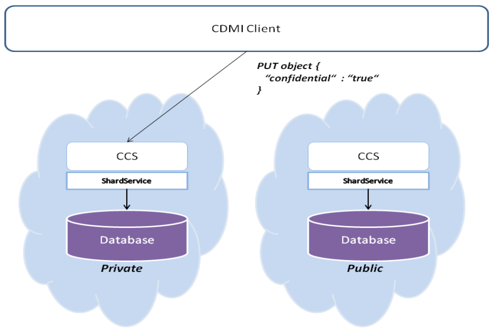


Figure 6: Storing confidential data.

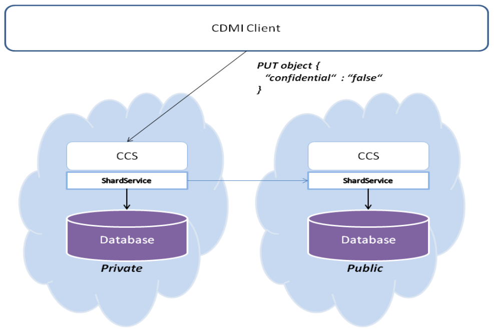


Figure 7: Storing non-confidential data.

Because of the metadata value, the ShardService will connect to the public cloud and the data will be stored in the public cloud, i.e., *vision-tb-2*. It is also possible to submit a such PUT requests to *vision-tb-2*.

There is no additional interface to which object creation operations and queries need to be submitted. Figure 6 and 7 show that a request can be sent to any shard in any cloud, and that cloud passes the request to the proper cloud to handle the request. This principle can be extended to handle several public or private clouds as well.

V. Further Hybrid Cloud Scenarios

So far, we have considered a hybrid cloud scenario where the location of objects is determined according to metadata. Being able to work with several cloud storage systems in a sharded manner offers several advantages. The first one is data confidentiality. Clients can store critical data on a secure cloud object store, and other data can be stored on general public cloud object stores, which might be cheaper and offer better extensibility.

However, the approach is more flexible and can be used in other scenarios as well. One scenario that VISION Cloud has implemented is a so-called on-boarding federation [11]. The purpose of this scenario is to migrate data from one cloud storage system to another. One important feature of the implemented on-boarding federation is to allow accessing *all* the data in the target cloud while the migration is in progress, i.e., while data is being transferred in the background. After the administrator has configured the federation, the objects will be transferred in the background to the client' container in the new cloud. If the client lists the contents of a federated container, all objects from all containers in the federation will be shown. If the client accesses objects which have not been on-boarded yet, the objects will be fetched on demand. The on-boarding handler intercepts GET-requests from the client and redirects them to the remote system on demand and schedules the background jobs for copying the clients' data from the remote cloud.

This helps to let a client become independent of a single cloud storage provider, i.e., a vendor lock-in is avoided. Having only a single vendor as a cloud storage service might limit the availability and scalability to that provider. In fact, the vendor lock-in of the stored data is the second among top ten obstacles for growth in cloud computing [2].

In the VISION Cloud project several further types of federation (cf. `federation.type` in Figure 4) have been discussed, but are not yet implemented: policy-based, synchronization, backup, multi-site-access, and company-merges. Based on the sharding mechanism presented in this paper, we can offer general strategies that distribute data according to a partitioning scheme. The idea behind sharding is splitting the data between multiple storage systems and ensuring that access to the data always occurs at the right place. In our case, we can partition and query data among the cloud storage systems.

The main advantage of database sharding is scalability, the ability to grow in a linear fashion as more servers are included to the system. Additionally, several smaller data stores are easier to handle and manage than huge ones. Furthermore, each shard contains less data and thus can be faster to query. Another positive effect is that each shard has a server of its own, resulting in less competition on resources such as CPU, memory, and disk I/O. And finally, there will be an increase of availability: If one shard fails, other shards are still available and accessible.

On the other side of the coin, several factors need to be considered to ensure an effective sharding. Although most of the applications are fault-tolerant, the storage tier is always the most critical part for reliability. Due to the distributed approach of multiple shards, the importance is even greater. To ensure a fault tolerant sharding precautions such as automated backups and several live copies of the shards need to be made. Next, a good partitioning scheme is required to match the needs of the system [13][14]; the following general approaches exist and can be controlled in our approach in principle:

1) *Vertical Partitioning*: All data related to a specific feature will be stored in the same place, i.e., images are in an image storage and videos in a video storage. On the one hand, this approach is easy to implement and causes only little overhead for the application. But on the other hand, the partitions might become uneven. Some shards might require more than one server.
2) *Range Based Partitioning*: In situations where data of a single feature needs to be distributed, it is important to find a meaningful way to spread the data. One approach is to define ranges for data values and to distribute data accordingly. Although this is also easy to implement, it will cause some unbalanced load distribution between the shards.
3) *Key Based Partitioning*: This requires a special entity with a unique key; this entity clusters data and can be used to identify the shard.
4) *Hash Based Partitioning*: Hash sharding involves processing values through a hash function to choose which server to store. Thus, each server will have a hash value, and each computed hash value will end up on one of those servers.
5) *Directory Based Partitioning*: This scheme keeps a lookup table somewhere in the cluster which keeps track of which data are stored in which storage. This approach means that the user can add servers to the system without the need of changing the application.

Further scenarios can support fault tolerance and high availability features in way that metadata control the number of replications. Those use cases will be part of future work.

VI. RELATED WORK

Though federation in cloud environments is still a research topic, some of the basic concepts and architectures of federation have already been researched intensively within the area of federated database management systems [10]. Sheth and Larson define a federated database system as a ”collection of cooperating but autonomous component database systems” including a ”software that provides controlled and coordinated manipulation of the component database system”. This definition places the federated database layer outside and on top of the component database systems that make up the federation. [10] also introduces a possible characterization of systems along the dimensions of distribution, heterogeneity and autonomy, and differentiates between tightly coupled systems (where administrators create and maintain a federation) and loosely coupled systems (where users create and maintain a federation). Moreover, the authors describe a five-layer reference architecture for federated database systems.

One project that offers a unified API between several data stores is presented by Bunch et al. [15]. In this work, the authors present a single API from Google App Engine to access different open source distributed database technologies. Such a unified API represents a fundamental building block for working with cloud storage as well as local NoSQL database servers. In contrast to our solution based on CCS, the implementation described by the authors provides access only to a single storage system at a time.

For the concurrent use of different storage providers or systems, Abu Libdeh et al. [16] propose a cloud storage system which is named as Redundant Array of Cloud Storage (RACS). It is placed as a proxy tier on top of several cloud storage providers. The authors describe adapters for three different storage interfaces, and point out that it can easily be expanded to additional storage interfaces. The approach uses erasure coding and distributes the contents of a single PUT request across the participating storage providers. Therefore, such a (write) operation must wait till the slowest of the providers completes the request. This is in contrast to the sharding of our work, where a PUT request is routed to a single storage system.

Another work which seems to be close to ours is presented by Brantner et al. in [17]. They build a database system on top of Amazon's S3 cloud storage. We also support Amazon S3 as one of our storage layer options. In their future work, they intend to include support for multiple cloud storage providers.

Additionally, there are a lot of multi-cloud APIs or libraries enabling unified access to multiple different cloud storage systems; these include Apache Libcloud [18], Smestorage [19] and Deltacloud [20]. They provide unified access to different storage systems, and protect the user from API changes. Although they enable administration features like stopping and running the storage instances, their storage driver functionalities are restricted to basic CRUD methods, most of them omitting a query interface.

A notable concept is found in the area of Content Delivery Networks (CDN). A content delivery network forms a network of servers around the world which maintain copies of data. When a user accesses the storage, the CDN infrastructure delivers the website from the closest servers. According to Broberg et al. [21], current storage providers have emerged as a genuine alternative to CDNs. In their work, they describe a cheaper solution by using cloud storage with their Meta Content Delivery Network (Meta CDN). Although Meta CDN makes use of several cloud storage providers, it is not a storage system by definition, and mostly focuses on read performance and lacks write performance.

In addition to the work mentioned above, there are also hybrid cloud solutions. Most of the current hybrid cloud offerings provide data transfer from private to public – instead of providing a unified view. The first example is Nasuni [22], which is a primary storage option. It is a form of network attached storage, which moves the user's on-premise data to a cloud storage provider following encryption. Nasuni's hybrid cloud approach combines on-premise storage nodes that gather the data and encrypt the data. Then, they send the encrypted data to a public cloud, which can be hosted at Amazon Web Services or at Microsoft Azure. The user can either store all the data in a single public cloud store, or can distribute them over multiple stores. Nasuni implements a migration approach, rather than a sharding approach such as ours, since data is eventually moved to the public cloud.

Nimbula [23] is another company that provides a service allowing the migration of existing private cloud applications to the public cloud using an API that permits the management of all resources. CloudSwitch [24] has also developed a hybrid cloud that allows an application to migrate its data to a public cloud.

Nirvanix [25] is one of the companies offering a hybrid cloud option. They offer a private cloud on premises to their customers, and enable data transfer to the public Nirvanix Cloud Storage Network. Although it is a hybrid cloud, it forces one to use only Nirvanix products. This represents a vendor lock in when it comes to the selection of the public cloud. In contrast, our adapter approach is not limited to a specific public cloud service.

Hybrid cloud storage solutions in the marketplace today provide a range of offerings to meet different demands of customers. Although there are many such offerings, they pose the risk of a vendor lock-in, because most of the companies use their own infrastructure. The most suitable work that matches the approach of our work is MetaStorage [26]. It represents a federated cloud storage system that is able to integrate different cloud storage providers. MetaStorage implements a distributed hash table service that replicates data on top of diverse storage services. It provides a unified view between the participating storage services or nodes while implementing sharding between them.

VII. SIMULATION RESULTS

In this section, we provide some basic performance tests for our architecture. We have examined different cases in order to evaluate the query performance using the CCS on top of two clouds, private and public. Our aim is first to check performance on a hybrid cloud setup with an increasing number of requests, and second, to see how the performance changes with a multi-threaded implementation.

For the tests we used the same data sets and similar setup as in our former work [8][27]. The test data is acquired from Deutsche Welle, which is one of the partners of the VISION Cloud project. Deutsche Welle has an analysis application which crawls data from YouTube across a number of news channels. 90 channels a day were tracked for a given time-frame, in total 490.000 videos have been collected together with their metadata. For the evaluation of our work, we used a subset of this data, which has in total 46.413 videos. Each video has the same amount of metadata and exactly the same fields.

We used two machines with the configuration as specified in Table I. We located these two machines in the same geographical area, having the same network. This is sufficient to analyze the performance overhead of the architecture, however, does not give results about the overall performance, which depends on the latency anyway . Each machine had a CouchDB database, Tomcat application server, and our Java Web Archive components installed. The main rationale was to test how well the implementation scales with larger volumes. The tests show three different cases: one querying for videos published in one day, one for videos published over two weeks, and one for videos published over four weeks. By using each of the resulting video ids, we sent 91, 1990 and 2908 consecutive requests to the underlying storage. Apart from that we re-ran the tests with a Java Thread Pool implementation to see the effects of parallelism in our system.

At first we uploaded our sample dataset by using the sharding implementation. This resulted in storage of 23206 video metadata on one machine, and 23207 video metadata on another machine. Afterwards we queried both of the storage systems. The total stack ran 20 times, and the average values are taken. To increase the precision, the longest and shortest run times are excluded from the overall measurement before taking the average. The results can be seen in Table II. The setup and the test runs were the same as used in our previous publications.

TABLE I: MACHINES USED IN A HYBRID SETUP

Designation, Processor	Cores	Clock Speed	L2 Cache	RAM	OS	Storage
2-core (4 threads)	1/1	1.90Ghz	4MB	4GB	64bit Win7	128GB SSD
2-core	1/1	2.20Ghz	2MB	3GB	32bit Win7	250GB HD

The first column represents the test configuration, single threaded or thread pooled. The next column gives the resulting number of requests to the underlying storage system. The average of the measured times are given in milliseconds, and

TABLE II: RESULTS OF THE HYBRID SETUP

	Request Count	Average Total (msec)	Standard Deviation	95% Conf.	Single request
Single Thread	91	2852,7	216,9	134,4	31,69
	1990	62404,9	4086,0	2532,5	31,38
	2908	96826,8	3670,6	2275,0	33,31
Thread Pool	91	1014,6	67,1	41,6	11,27
	1990	25764,2	2617,3	1622,2	12,95
	2908	39252,6	2839,6	1759,9	13,50

single request times are calculated as an average time divided by the request counts. As it can be seen, the single request times do not change much as the number of requests increases and are on average 30 milliseconds, which is acceptable. Also of notice is the multi-threaded implementation. In all of the cases, it boosted performance significantly.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we presented a new approach for hybrid cloud storage that is based upon the idea of federation as carried out by the VISION Cloud project. Our approach provides a uniform interface for handling confidential and non-confidential data, the first kept in an on-premise data store, the later stored in a public cloud. The key idea is to use metadata for controlling where data is stored. As a technical basis, we use the VISION Cloud software stack [7] where such a metadata concept is an integral part. We show in detail how well-suited VISION Cloud and its storage system is to support hybrid scenarios and how to extend it in order to support hybrid scenarios.

In principle, it is possible to offer additional sharding scenarios in the VISION Cloud project, beyond the privacy scenario we have presented for this work. The overall approach also allows for adding various further sharding strategies, such as region-based, load balancing, or storage space balancing, redundancy level control, etc. In fact, our future work will be dedicated to extending the hybrid approach and to elaborating more on query load balancing (based on metadata). Another aspect that requires attention is migration if security settings are changing.

ACKNOWLEDGMENT

The research leading to the results presented in this paper has received funding from the European Union's Seventh Framework Programme (FP7 2007-2013) Project VISION-Cloud under grant agreement number 217019.

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing (draft)," NIST special publication, vol. 800, no. 145, 2011, p. 7.

[2] A. Fox et al., "Above the clouds: A berkeley view of cloud computing," Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, vol. 28, 2009.

[3] "Nosql databases," at: http://nosql-database.org, [retrieved: April, 2014].

[4] B. Tudorica and C. Bucur, "A comparison between several nosql databases with comments and notes," in Roedunet International Conference (RoEduNet), 2011 10th, 2011, pp. 1–5.

[5] E. Kolodner et al., "A cloud environment for data-intensive storage services," in CloudCom, 2011, pp. 357–366.

[6] E. Kolodner (2) et al., "Data intensive storage services on clouds: Limitations, challenges and enablers," in European Research Activities in Cloud Computing, D. Petcu and J. L. Vazquez-Poletti, Eds. Cambridge Scholars Publishing, 2012, pp. 68–96.

[7] "Vision cloud project consortium, high level architectural specification release 1.0, vision cloud project deliverable d10.2, june 2011." at: http://www.visioncloud.com, [retrieved: April, 2014].

[8] M. C. Jaeger et al., "Cloud-based content centric storage for large systems," in Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on, 2012, pp. 987–994.

[9] "Cloud data management interface version 1.0," at:http://snia.cloudfour.com/sites/default/files/CDMI_SNIA_Architecture_v1.0.pdf, year = 2010, month = April, note = [retrieved: April, 2014], SNIA Storage Networking Industry Association.

[10] A. Sheth and J. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," ACM Computing Surveys, no. 22 (3, 1990, pp. 183–236.

[11] G. Vernik et al., "Data on-boarding in federated storage clouds," in Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, ser. CLOUD '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 244–251. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2013.54

[12] "Vision cloud project consortium: Data access layer: Design and open specification release 2.0, deliverable d30.3b, sept 2012." at: http://www.visioncloud.com/, [retrieved: April, 2014].

[13] D. Obasanjo, "Building scalable databases: Pros and cons of various database sharding schemes," URL: http://www.25hoursaday.com/weblog /2009/01/16/BuildingScalableDatabasesProsAndConsOfVariousDatabaseShardingSchemes.aspx, vol. 10, 2009, p. 2012, [retrieved: April, 2014].

[14] R. Cattell, "Scalable sql and nosql data stores," ACM SIGMOD Record, vol. 39, no. 4, 2011, pp. 12–27.

[15] C. Bunch et al., "An evaluation of distributed datastores using the appscale cloud platform," in Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, ser. CLOUD '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 305–312, [retrieved: April, 2014].

[16] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "Racs: a case for cloud storage diversity," in Proceedings of the 1st ACM symposium on Cloud computing, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 229–240, [retrieved: April, 2014].

[17] M. Brantner, D. Florescu, D. Graf, D. Kossmann, and T. Kraska, "Building a database on s3," in Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008, p. 251.

[18] "Apache libcloud a unified interface to the cloud," at: http://libcloud.apache.org/, [retrieved: April, 2014].

[19] "smestorage," at: https://code.google.com/p/smestorage/, [retrieved: April, 2014].

[20] "deltacloud," at: http://deltacloud.apache.org/, [retrieved: April, 2014].

[21] J. Broberg, R. Buyya, and Z. Tari, "Service-oriented computing — icsoc 2008 workshops," G. Feuerlicht and W. Lamersdorf, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, ch. Creating a 'Cloud Storage' Mashup for High Performance, Low Cost Content Delivery, pp. 178–183.

[22] "Nasuni," at: http://www.nasuni.com/, [retrieved: April, 2014].

[23] "Nimbula," at: http://en.wikipedia.org/wiki/Nimbula, [retrieved: April, 2014].

[24] "Cloudswitch," at: http://www.cloudswitch.com/, [retrieved: April, 2014].

[25] "Nirvanix," at: http://www.nirvanix.com/products-services /cloudcomplete-hybrid-cloud-storage/index.aspx, [retrieved: April, 2014].

[26] D. Bermbach, M. Klems, S. Tai, and M. Menzel, "Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs," in Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, ser. CLOUD '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 452–459, [retrieved: April, 2014].

[27] M. C. Jaeger et al., "Extending cloud-based object storage with content centric services," in CLOSER 2013 - Proceedings of the 3rd International Conference on Cloud Computing and Services Science, Aachen, Germany, 8-10 May, 2013, 2013, pp. 279–289.

High Performance Computing in a Cloud Using OpenStack

Roman Ledyayev and Harald Richter
Institute for Informatics, Clausthal University of Technology
Clausthal, Germany
Email: {roman.ledyayev, hri}@tu-clausthal.de

***Abstract*—Cloud computing is a rapidly gaining popularity computing paradigm, prompted by much research efforts. However, not much work is done in the area of joining cloud computing with high performance computing in an efficient way, e.g., for scientific simulation purposes. Moreover, there is even less research effort in making cloud computing for scientific simulations more efficient and suitable for specific simulation codes. This paper presents an ongoing "SimPaaS" project – our research efforts in building a cloud based platform for scientific simulations. It deals with some challenging features in cloud computing, such as performance. The concepts and methods proposed in this paper allow customizing and optimizing cloud infrastructure to increase its performance and to meet certain requirements, drawn from the analysis of case study simulation codes. Even though clouds are not the most suitable environments for high performance computing, the conclusion was drawn that there are ways to increase cloud performance and effectively combine the two paradigms.**

Keywords-simulation; cloud computing; High Performance Computing; OpenStack; VM

I. INTRODUCTION

In a project, entitled Simulation Platform as a Service (SimPaaS) [34], we attempt to marry two technology domains: cloud computing and scientific simulations, sometimes referred to as scientific computing or High Performance Computing (HPC). In this introduction, a brief definition of these two technologies in context of the ongoing research project is presented. Note, that in this paper, the terms scientific simulations and High Performance Computing are used interchangeably.

There are multiple definitions out there of the cloud [1]. In the project under consideration, the definition provided by the National Institute of Standards and Technology (NIST) [2] was adopted and relied on.

Cloud computing is a relatively new computing paradigm, composed of a combination of grid computing and utility computing concepts. Cloud promises high scalability, flexibility, cost-effectiveness, power savings and various other benefits to satisfy ever emerging computing requirements of modern applications.

The scalability, flexibility, cost-effectiveness and relative user-friendliness of various cloud services make it also an attractive model to address computational challenges in the scientific community. Individual research groups, who decide not to build their own cloud environments, do not need to provide and maintain IT-infrastructure on their own, but instead rely on cloud-computing services to satisfy their needs. However, they can also build their own specialized cloud services, which can be implemented on-site, and which could enable them to customize and optimize their cloud utilization specifically for scientific simulations.

High Performance Computing is an important field with two branches. These are: numerical simulations and big data analysis. The latter is well suited for clouds, because of the distributed file systems available in clouds. Massive amounts of data can be stored in a distributed file system and subsequently processed by individual cloud nodes. However, up to now it is an unsolved problem for numerical simulations to run efficiently on a cloud, because clouds, by nature, are distributed systems and thus based on TCP/IP communication. TCP/IP, in turn, has no quality of service with respect to bandwidth and latency, thereby creating much variance in both key parameters. As a result, exchanging data with high bandwidth and low latency becomes dependant on the traffic in the Internet. On the other hand, HPC is a numerical intensive task, based on highly efficient Inter-Process Communication (IPC). It is difficult to join both worlds, clouds and HPC, because in parallel and multicore computers, which are the hardware basis for HPC, interprocess, interprocessor and intercore communications are highly optimized. Additionally, clouds are intensively using the concept of virtualization, which results in computing overheads, as seen from the HPC's point of view. As a consequence, a lot of CPU power is not used for executing HPC codes, but to run the cloud operating system, such as OpenStack [3].

In this paper, a set of methods that can transform OpenStack into a middleware, able to accommodate HPC, is presented. The proposed method set is based on a mixture of hardware and software changes.

The ultimate goal of the project is to provide a cloud-based software platform for scientific simulations (Simulation as a Service). Figure 1 shows how such service would fit in the cloud stack. SimPaaS project prototype cloud will implement a platform on top of Infrastructure as a Service (IaaS), which provides virtualized resources for automatic distributed and scalable deployment. The specific simulation applications can be implemented as Software as a Service (SaaS) on top of the simulation cloud.

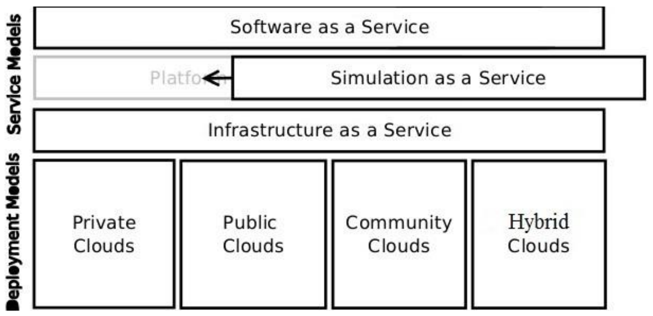


Figure 1. Simulation as a Service in the cloud stack

The rest of the paper is organized as follows. Section II presents related work. In Section III, various types of simulation codes and their impact on Inter-Process Communication and cloud efficiency are discussed and more details about the current test bed cloud setup are presented. Next, Section IV presents the proposed method set of making the current cloud suitable for High Performance Computing. Finally, some preliminary conclusions and directions for future research are outlined in Section V.

II. RELATED WORK

There are projects and research articles that have made fairly successful attempts to introduce scientific simulation into the area of cloud computing on various levels and to various degrees.

First, we must admit that significant research efforts [4] have been made to migrate scientific computing simulations to the cloud. Several science clouds [5] have been established. There have also been attempts to design distributed computing frameworks, which would fully support scientific computing algorithms and take advantage of those characteristics of a cloud that have made it such a convenient and popular source for utilizing computing resources [6]. Several simulations and applications have been executed on these hybrid clouds. Research teams have been able to measure the performance of running those simulations in the clouds, thus evaluating the efficiency of scientific computing on the cloud in general [33]. Jakovits et al. [6] drew a conclusion that clouds are perfect environments for scientific simulations. It was observed that the communication, interference and other latencies added by the virtualization technology are the major hindrances for executing scientific computing applications on the cloud [4][7][9]. The cloud computing community is trying to address these issues and a number of interesting solutions have been proposed over the last few years [7][8].

Even though virtualization is to be taken into account when using clouds for HPC, some studies show, that for running scientific codes in parallel, performance is comparable, indicating that the virtual machine hosting environment introduced little overhead, even when all of the available cores were running at full capacity [34].

There have also been projects that used OpenStack to build and manage a scientific cloud. One such project worth mentioning was called Applied Computational Instrument for Scientific Synthesis (ACISS) [10]. Some objectives of the ACISS project overlap with our own goals for SimPaaS project [10].

Second, even though there has been much research in cloud computing and related technologies, comparatively little work has focused on their use in simulation, especially parallel and distributed simulation. Execution of parallel and distributed simulations over clouds not only represents an interesting opportunity, but also presents certain technical challenges, as discussed by Fujimoto et al. [9].

However, it is clear that significant further developments are needed to create a platform for materials simulations that meets all the particular needs of HPC without requiring further configuration and is accessible not only to system administrators but also to general public users. Furthermore, the questions of cost-effectiveness and performance have not been conclusively answered and need to be addressed for each type of scientific cloud application. In particular, concerns about cloud computing performance are strong in the materials science community. Jorissen et al. [11] shows that Scientific Cloud Computing (SCC) is especially appropriate for materials science and quantum-chemistry simulations, which tend to be dominated by computational performance rather than data transfer and storage.

Despite the number of cloud computing research projects mentioned above, which deal with cloud computing and scientific simulation, there have not been, to the best of our knowledge, very many efforts to design and implement techniques to address specific performance requirements and optimize the overall resource utilization of simulation applications run in a cloud.

III. CURRENT SETUP AND SIMULATION APPLICATIONS

This section describes the current setup of the cloud test bed based on OpenStack, the choice of tools and gives brief information about the applications used in the experimental research (simulation case studies).

A. *Scientific Cloud Based on OpenStack*

OpenStack, co-founded by Rackspace and NASA in 2010, is quickly becoming one of the most popular open source cloud computing platforms. According to its mission statement, the OpenStack developers strive to produce the platform that will be simple to implement and massively scalable [12]. Admittedly, there are a number of alternatives to OpenStack, both in the open source and commercial arena. Eucalyptus [13], CloudStack [14], Joyent [15], OpenNebula [16] and proprietary-powered pioneers like Amazon Web Services [17] and VMware [18]. Our choice of OpenStack over these other platforms was motivated by a few factors. One of them is active development, which keeps it up to date with new releases every half a year. Other reasons are: less overhead, better scalability, and its open source nature. It has also been actively used by our partners – GWDG [19].

Figure 2 depicts a high level diagram of the current prototypical cloud setup. Grizzly 1.3 release of OpenStack was used and deployed on Ubuntu 12.04.3 LTS Precise 64 bit (kernel version 3.2.0-52-generic) as the operating system. Ubuntu with KVM are used as Hypervisor on one of the machines, which serves as cloud controller.

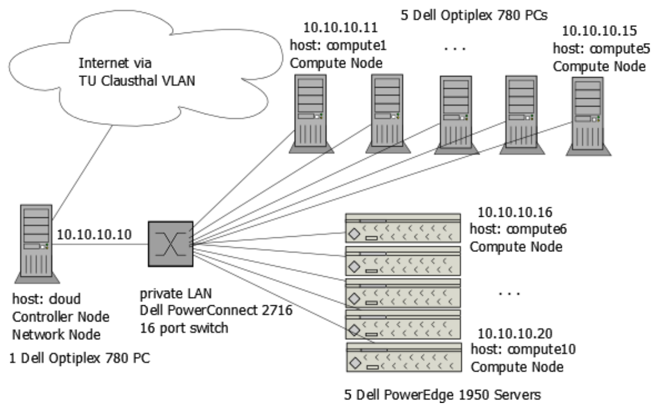


Figure 2. Current setup of the prototypical cloud

To measure the performance of the cloud, Ganglia [20] monitoring system (Ganglia Web 3.5.10) was installed on each node. It measures CPU, memory and network performance and loads. At the moment, we also consider automation tools like Puppet modules [21] and attempt to integrate the prototypical cloud with other research projects, such as Cloud4E project [22].

B. Simulation Applications

The primary focus was made on simulation applications in materials science, high energy physics and public transport networks.

1) Modelling and Optimization of Public Transport Networks

These are mathematical solvers for planning and optimization of public networks, which are running on multi-core platforms. The software used: Xpress and Mosel [23], LinTim [24].

2) High Energy Physics Monte Carlo Simulation and Data Analysis

This is both computational intensive and data-intensive simulation and data analysis. These applications are using data-parallelism and there is no communication between the processes. The software used: ROOT/PROOF [25].

3) Material Simulation

Open Source Field Operation and Manipulation (OpenFOAM [26]) is used to perform simulations in computational fluid dynamics (CFD). It covers the whole spectrum of the machines on which it can be compiled and run, from single-core and multi-core (e.g., Intel) to parallel computer (e.g., HLRN supercomputer [27]) and Graphical Processing Units (GPUs) which are currently in progress. The default, however, is parallel computer. Currently, only MPI library (OpenMPI) is used for message-passing.

C. Initial Simulation Test Results

In Figure 3, some preliminary simulation test results are presented. OpenFoam's "breaking of a dam" use case was selected for test runs, using the same configuration files (mesh size, decomposition). The simulation was executed on both, dedicated physical machines and VMs in the cloud.

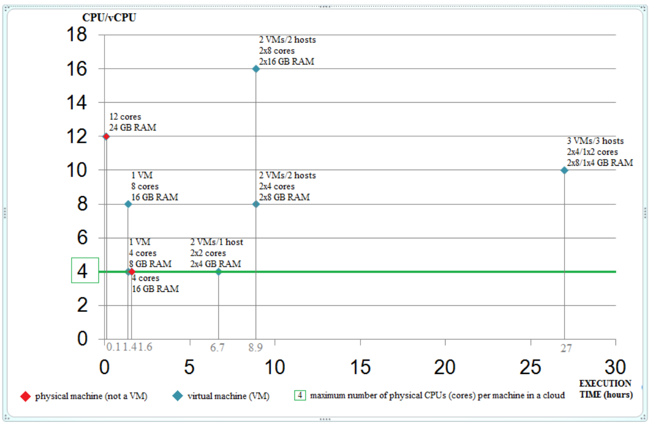


Figure 3. Initial OpenFOAM simulation test results

It was observed, that increasing the number of VMs and/or the number of hosts on which those VMs are run, drastically increases the time necessary to complete the simulation. It is also worth noting, that increasing the number of virtual cores per VM, even if that number is above the number of the physical cores available, had no visible impact on performance and provided no speedup in execution time.

Thus, some preliminary conclusions could be drawn from these test results. First of all, virtualization does not seem to have any significant impact on performance when 1 VM is used and there is no overcommitting in the number of CPUs (the number of virtual CPUs is no larger than the number of physically available CPUs). However, increasing the number of VMs significantly increases simulation execution time. This is especially noticeable if those VMs are run on two or more separate hosts. In such cases, the simulation is executed in parallel and in distributed mode, as opposed to cases when it is run on one VM (non-distributed). The fastest execution time was achieved when simulations were run on 1 VM with the number of virtual cores equal to the number of physical cores of the physical computer on which it was run. In this case, the execution time was almost equal to the time it takes to run the same simulation on a physical computer outside of the cloud.

Independent of the used programming paradigm and library, simulation codes can be categorized with respect to their interprocessor communication. These categories are important for understanding the measures proposed.

D. Types of HPC Simulation Codes

1) Category 1: Multiple Runs of a Sequential Code

In this case, a sequential code has to be executed multiple times but with different input values respectively, which is called parameter sweep. In theory, a cloud can achieve high throughput here, because all sweeps can be executed in parallel by different virtual cores on the same Virtual Machine (VM) or by different VMs. Inter-Process Communication is not needed, with the exception of the beginning and the end of the runs, where some master node must distribute the set of input values to computing nodes and, subsequently, collect their results.

2) Category 2: Single Run of a Parallel Code

The decisive question in this case is how much IPC is needed in the considered code, because IPC is the bottleneck in a cloud compared to a parallel computer or a multi-core CPU. Fortunately, if the code is data parallel and not function parallel then IPC is mostly moderate. Most solvers for partial differential equations, for example, belong to data parallel class. However, if the code is function parallel then standard clouds are not a good choice, because a high fraction of IPC and an intricate pattern of communication are normally present. Only a significant improvement of the cloud's inter-core and/or inter-VM bandwidth and latency could help, which is suggested in the next section.

3) Category 3: Multiple Runs of a Parallel Code

This is the combination of the two cases discussed above, which means that respective problems and prerequisites are also combined. To avoid slowdowns, either cloud nodes must be upgraded as described below or the scheduler must be modified and the cloud's IPC must be made more effective. A code written for a parallel computer with specific features should be executed on a machine in a cloud with the same properties, if possible. However, OpenStack does not guarantee that a specific real machine is indeed chosen for a given VM. Therefore, changes must be made to address this shortcoming as well.

IV. METHODS OF INCREASING CLOUD PERFORMANCE

The method set proposed in this paper can be divided into two categories of hardware and software changes.

A. Hardware Changes

In OpenStack, it is a common practice that heterogeneous computers are configured to be computing or storage nodes, together with a controller node. These nodes and the controller are coupled on ISO layers 1 and 2 by a Gigabit Ethernet switch. Optionally, a VLAN can also be established. The hardware proposed here uses a 10 Gigabit switch and proper network cards in 10 Gigabit Ethernet or Infiniband technology in a way that a Beowulf cluster comes into existence. Beowulf cluster implies that homogenous computers are coupled with identical network interfaces and with high-speed switches, such that deterministic latency and bandwidth are achieved on ISO layers 1 and 2. Additionally, the TCP/IP protocol set has to be abandoned, because it is slow and non-deterministic and because a Beowulf cluster is not world-wide and distributed but localized in a computing center. As a consequence, most of the TCP/IP features are not needed. However, in order to maintain compatibility with existing MPI and OpenMP implementations, the Berkeley socket API must be preserved. This is possible by employing the commercially available "VMA Messaging Accelerator Protocol" [28] from Mellanox. It provides the Berkeley API but bypasses the TCP/IP protocol stack and writes user data directly into the Mellanox network cards and reads input from them without much protocol overhead. So, changes in the user codes are not needed.

1) Unnecessary TCP/IP functions in Beowulf cluster

Data transmission errors are practically excluded in the described cluster, because of the relatively short distances between switches and nodes. As a consequence, the automatic packet retransmission of TCP is not needed. Additionally, the TCP sliding window protocol is unnecessary, because all network interfaces are identical and have the same speed and buffer sizes. Furthermore, the packet reorder function of TCP is also not needed, because only point-to-point connections without routers exist. No packet that is sent later can arrive earlier. Additionally, IP packet segmentation and reassembly is also not necessary, because there is the same LAN technology in usage everywhere, without differences in maximum frame size. Finally, IP routing is not useful here, because there is no router but switches in the cluster. As a consequence, nearly all functions of the TCP/IP stack can be dismissed, as it is done by the VMA messaging accelerator, which boosts bandwidth and drastically reduces latency.

2) Bandwidth and latency improvements

Mellanox claims that their accelerator reduces latency on a 10 Gigabit Ethernet to 1.6 µs between two sending and receiving Berkeley socket APIs in case of 12 bytes payload per frame. This is significantly faster than via the Internet, but it means that the compute and storage nodes of the cloud are no longer part of the Internet. Only their controller node can stay connected. However, this is fully compatible with concepts used in OpenStack, which allows using floating IP addresses for compute and storage nodes that must overlap with publicly used addresses. Additionally, the network service of OpenStack can be exclusively localized in the controller node which has Internet connection.

Suggestions could be made to fix bandwidth and latency issues with hardware devices such as fiber optic networking, ramdisks/SSDs, etc. The substitution of copper cables as computer interconnects by glass fiber optics can fix the bandwidth problem only if fiber speeds are significantly higher than 10 GB/s, which is the limit for copper. However, it will not fix the latency problem, because the electric/optic converters introduce additional delays and because the speed of light is similar in glass and in copper (about 0.75c). The replacement of hard drives in the cloud by SSDs could accelerate the throughput of some OpenStack services. But, as with the glass fiber solution, the cloud costs would significantly increase, which is not desirable in this case.

3) Hardware scalability

The hardware scalability, necessary to engage thousands of CPUs in one cluster, is achieved by a hierarchical cascade of layer 2 switches and by employing VLAN technology. This allows enlarging the spatial circumference of the cloud to several kilometers, which is sufficient to accommodate thousands of computers. There should be no problem with scalability, since Mellannox 10 Gigabit Ethernet switches are supported by OpenStack via using Mellanox plugins.

B. Software Changes

Software changes have to be made in the underlying operating system (Linux Ubuntu), in the OpenStack network service (Neutron) and its scheduler.

1) New operating system for OpenStack

The first software change is to replace Linux host OS, on which OpenStack runs, by a Real-Time Linux such as the RT Preempt Patch [29] or Xenomai [30]. The reason for that is that standard Linux, such as Ubuntu, uses the Completely Fair Scheduler from Ingo Molnar, which does not allow prioritizing processes as real-time operating systems do. The selected RT OS is used not only as host OS for OpenStack, but also as guest OS for every VM. This gives users a precise control over which task is scheduled at which time. Such a feature is important when two user tasks want to exchange data at the same time, because of Inter-Process Communication.

2) New OpenStack scheduler

The Nova scheduler of OpenStack determines which physical compute node is allocated to which VM as soon as the VM is provisioned. This reflects a scheduling in space, but not in time, and is of static nature. An automatic live migration of VMs that periodically balances out the load between physical nodes does not exist in the Grizzly release. This is not favorable for HPC, because resources may become extremely overloaded.

a) Periodic live migration

Because of the fact that an RT OS was chosen for both, host and guest OS, it is possible to re-allocate VMs for load balancing, because Nova can be prioritized before user tasks and executed periodically. To achieve this, scheduling in host and guest OS must be synchronized, so that all schedulers act together.

b) Gang scheduling

Schedulers must schedule all sets of communicating VMs simultaneously, as soon as they are starting data exchange, so that they can send and receive information efficiently (Gang Scheduling). Otherwise, unnecessary waiting times would be the consequence, because rendezvous between senders and receivers cannot take place. In that case, sent data must be buffered until receivers are ready to proceed and receivers must wait until senders can send data, which is unfavorable for HPC.

c) Advanced reservation

Accessing a hard drive requires up to 10 µs until the searched record is found. Such I/O delays are unbearable for HPC, especially if they occur one after another in physical nodes that must communicate with each other. Advanced reservation, in this case, means that user code can be instrumented with an OpenStack call that is directed to the storage services, Cinder and Swift, and that read and cache or cache and write a whole data object in advance before the hard drive is ready. This allows hiding I/O latencies and thus improves communication latency.

d) High priority scheduling

Some system processes, such as Nova itself or the Mellanox messaging accelerator, must be scheduled before user tasks. Consequently, user tasks must be rescheduled for high priority system tasks. Priority scheduling is a standard feature of all RT OS. However, for the synchronous cooperation of host and guest schedulers, a software framework must be created that instructs all schedulers via a common API, that should also be available in Open Cloud Computing Interface (OCCI) [31], which is both, a protocol and an API for all kinds of management tasks.

3) New OpenStack networking service

VLANs are needed for a scalable Beowulf cluster to extend cable lengths and cascade switches. To make this possible, the OpenStack networking service must be enhanced with the following functions:

- Generation of VLAN-IDs.
- Creation of a mapping table “VLAN-ID – Port number” for every switch, according to the cluster topology used, so that each switch can forward a frame to correct destination.
- Generation of IEEE 802.1Q Ethernet header tags at every switch input port and removal of these tags at every switch output port. This is needed for Ethernet interfaces that do not support VLAN tags.
- Defining which method is used for assigning a specific frame to its VLAN.
- Automatic setting of frame priorities. This is needed in cases when multiple frames collide at the same time at the same switch output, and a decision must be made by the switch which frame gets a passage first. This allows resolving conflicts in the transport system of the cloud under full control of the user code, thus avoiding speed up degradations.

V. CONCLUSION AND FUTURE WORK

The contribution of this paper is two-fold. In the first place, the possibility of effectively combining cloud computing, which by its very nature is not a very suitable environment for applications designed for HPC platforms, with High Performance Computing was examined. In the second place, a set of methods which, if followed and implemented, could make clouds more suitable for running HPC codes was proposed.

By running OpenFoam as a benchmark for OpenStack, we found out that this cloud operating system is not well-suited for OpenFoam, because it degrades performance with respect to a reference computer of the same capabilities that is outside of OpenStack. The results can be generalized to any cloud operating system, to any computational fluid dynamics codes and to any HPC codes in general.

We plan to investigate why this happens. At the moment, we believe this happens because of potential overcommitting of physical resources by virtual ones, and by data exchanges needed between VMs running on two computers and between virtual cores running in the same VM, as soon as it takes place via TCP/IP. With TCP/IP, parallel computing mutates into distributed computing which results in code slow down. However, three use cases with different slow down factors could be identified, and measures could be given to repair this behavior. These measures are: 1) Abandon the internal functions of TCP/IP, but preserve its Berkeley socket API, because of the often used MPI library. 2) Replace the distributed cloud by a Beowulf cluster and install OpenStack on it. 3) Replace guest and host operating systems by real-time Linux. 4) Add a frame-work that schedules communicating VMs and cores synchronously. 5) Introduce disk I/O in advance for data objects to avoid unpredictable message latencies. 6) Add periodic live-migration of VMs for load balancing between physical CPUs.

Future work will be to investigate how much these measures can change the cloud's HPC efficiency under the boundary conditions of given user codes and cloud hardware.

Our future efforts will concentrate on further analysis of the issues mentioned above and the propositions described in the previous section, experimenting with the results of running the simulation codes inside the cloud, and, finally, designing and implementing one or more of the proposed solutions from the method set.

References

[1] A. Isherwood (Hewlett-Packard's Vice President of European Software Sales), quoted in ZDnet News, December 11, 2008.

[2] P. Mell and T. Grance, "The NIST definition of cloud computing", NIST Special Publication 800-145, September, 2011.

[3] http://www.openstack.org/, [accessed: January, 2014].

[4] S. Srirama, O. Batrashev, P. Jakovits, and E. Vainikko, "Scalability of Parallel Scientific Applications on the Cloud," Scientific Programming, vol. 19, Apr. 2011, pp. 91–105, 2011, doi:10.3233/SPR-2011-0320.

[5] S. N. Srirama, O. Batrashev, and E. Vainikko, "SciCloud: Scientific Computing on the Cloud," Proc. 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010, pp. 579–580, doi:10.1109/CCGRID.2010.56.

[6] P. Jakovits, S. N. Srirama, and I. Kromonov, "Stratus: A Distributed Computing Framework for Scientific Simulations on the Cloud," Proc. IEEE 14th International Conference on High Performance Computing and Communications (HPCC 2012), IEEE Press, 2012, pp. 1053–1059, doi: 10.1109/HPCC.2012.154.

[7] 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, 2011, pp. 23–50, doi:10.1002/spe.995.

[8] Q. Li and Y. Guo, "Optimization of Resource Scheduling in Cloud Computing," Proc. 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2010), pp. 315–320, doi: 10.1109/SYNASC.2010.8.

[9] R. M. Fujimoto, A. W. Malik, and A. J. Park, "Parallel and distributed simulation in the cloud," SCS M&S Magazine, issue 3, July 2010, [Online]. Available: http://www.scs.org/magazines/2010-07/index_file/Articles.htm, [accessed: January, 2014].

[10] C. Hoge, "Building a scientific cloud computer with OpenStack," OpenStack Day, Portland, July 16-20, 2012. [Online]. Available: http://www.oscon.com/oscon2012/public/schedule/detail/24261, [accessed: January, 2014].

[11] K. Jorissen, F.D. Vila, and J.J. Rehr, "A High Performance Scientific Cloud Computing Environment for Materials Simulations," Computer Physics Communications, vol.183, issue 9, pp. 1911-1919, doi: 10.1016/j.cpc.2012.04.010.

[12] https://wiki.openstack.org/wiki/Main_Page, [accessed: January, 2014].

[13] http://www.eucalyptus.com/, [accessed: January, 2014].

[14] http://cloudstack.apache.org/, [accessed: January, 2014].

[15] http://www.joyent.com/, [accessed: January, 2014].

[16] http://opennebula.org/start, [accessed: January, 2014].

[17] http://aws.amazon.com/, [accessed: January, 2014].

[18] http://www.vmware.com/, [accessed: January, 2014].

[19] http://www.gwdg.de/index.php, [accessed: January, 2014].

[20] http://ganglia.sourceforge.net/, [accessed: January, 2014].

[21] https://wiki.openstack.org/wiki/Puppet-openstack, [accessed: January, 2014].

[22] http://www.cloud4e.de/, [accessed: January, 2014].

[23] http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Mosel.aspx, [accessed: January, 2014].

[24] http://lintim.math.uni-goettingen.de/index.php?lang=en, [accessed: January, 2014].

[25] http://root.cern.ch/drupal/content/proof, [accessed: January, 2014].

[26] http://www.openfoam.com/, [accessed: January, 2014].

[27] https://www.hlrn.de/home/view, [accessed: January, 2014].

[28] http://www.mellanox.com/related-docs/prod_acceleration_software/VMA.pdf, [accessed: January, 2014].

[29] https://rt.wiki.kernel.org/index.php/Main_Page, [accessed: January, 2014].

[30] http://www.xenomai.org/, [accessed: January, 2014].

[31] http://occi-wg.org/, [accessed: January, 2014].

[32] J.J. Rehr, F.D. Vila, J.P. Gardner, L. Svec, and M. Prange, "Scientific Computing in the Cloud," Computing in Science & Engineering, vol. 12, issue 3, pp. 34-43, 2010, doi: 10.1109/MCSE.2010.70.

[33] P. Saripalli, C. Oldenburg, B. Walters, and N. Radheshyam, "Implementation and Usability Evaluation of a Cloud Platform for Scientific Computing as a Service (SCaaS)," 2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC), 2011, pp. 345-354, doi: 10.1109/UCC.2011.58.

[34] http://www.simzentrum.de/en/education/cloud-basierte-software-infrastruktur-fuer-verteilte-simulation/, [accessed: May, 2014].

Performance Comparison of KVM, VMware and XenServer using a Large Telecommunication Application

Sogand Shirinbab, Lars Lundberg, Dragos Ilie
School of Computing, Blekinge Institute of Technology, Sweden
{Sogand.Shirinbab, Lars.Lundberg, Dragos.Ilie}@bth.se

***Abstract*—One of the most important technologies in cloud computing is virtualization. This paper presents the results from a performance comparison of three well-known virtualization hypervisors: KVM, VMware and XenServer. In this study, we measure performance in terms of CPU utilization, disk utilization and response time of a large industrial real-time application. The application is running inside a virtual machine (VM) controlled by the KVM, VMware and XenServer hypervisors, respectively. Furthermore, we compare the three hypervisors based on downtime and total migration time during live migration. The results show that the Xen hypervisor results in higher CPU utilization and thus also lower maximum performance compared to VMware and KVM. However, VMware causes more write operations to disk than KVM and Xen, and Xen causes less downtime than KVM and VMware during live migration. This means that no single hypervisor has the best performance for all aspects considered here.**

Keywords-Cloud Computing; KVM; Live Migration; VMware vMotion; XenMotion.

I. INTRODUCTION

Virtualization has many advantages over non-virtualized solutions, e.g., flexibility, cost and energy savings [19][34]. As a more specific example, consider the cost associated with test hardware used during professional software development. This includes the initial price for purchasing the equipment, as well as operational costs in the form of maintenance, configuration and consumed electricity. For economic reasons, organizations often choose to use virtualized test servers, so that the test hardware can be shared and maintained in a cost-effective way [20]. In order to provide maximum resource utilization, there should be no restrictions on the mapping of VMs to physical computers, i.e., it should be possible to run a VM on any physical server. In order to balance the load, it is desirable that a VM running on a physical host could be restarted on another physical host, i.e., there is a need for migrating VMs from one physical server to another [21][22][23]. There is support for migration in many commonly used virtualization systems, e.g., KVM Live Migration [16], VMware's vMotion [18] and XenServers's XenMotion [17].

There are three different approaches to VM migration: cold migration, hot migration and live migration. When cold migration is used the guest Operating System (OS) is shut down, the VM is moved to another physical server and then the guest OS is restarted there. Hot migration suspends the guest OS instead of shutting it down. The guest OS is resumed after the VM is moved to the destination host. The benefit of hot migration is that application running inside the guest OS can preserve most of their state after migration (i.e., they are not restarted from scratch). In the live migration approach [13], the VM keeps running while its memory pages are copied to a different host. Live migration reduces the downtime dramatically for applications executing inside the VM. Live migration is thus suitable for high-availability services.

In this paper, we compare the performance of KVM, VMware and XenServer, for two different scenarios: when no VM is migrated, and when a VM is migrated from one physical server to another. The work load is, for both scenarios, a large real-time telecommunication application. In the case when no VM is migrated, we measure the CPU utilization, the disk utilization (the number of write operations), and the average application response time. When a VM is migrated we measure the CPU utilization, the disk utilization (the number of write operations), and the down time due to live migration.

The rest of the paper is organized as follows. In Section II the state of the art is summarized. Section III describes the experimental setup for the different hypervisors, and, in Section IV, we compare and analyze the results for KVM, VMware and XenServer. Finally, related work is discussed in Section V. Section VI concludes the paper.

II. STATE OF THE ART

A. Virtualization

In its simplest form, virtualization is a mechanism for several virtual OS instances on a single physical system. This is typically accomplished using a Hypervisor or Virtual Machine Monitor (VMM), which lies between the hardware and the OS. Virtualization is often beneficial for environments consisting of a large number of servers (e.g., a datacenter).

A virtualization solution relies on several components, such as CPU virtualization, memory virtualization, I/O virtualization, storage virtualization, and so on. In this paper we focus specifically on CPU and memory virtualization.

Current approaches to virtualization can be classified into: full virtualization, paravirtualization and hardware assisted virtualization [11][12].

Full virtualization uses binary translation which translates the kernel code so that privileged instructions can be converted to user-level instructions during runtime. Detection and translation of privileged instructions typically carries a large performance penalty. KVM and VMware support this approach.

Paravirtualization attempts to alleviate the performance of full virtualization by replacing privileged instructions with specific function calls to the hypervisor, so called hypercalls. This requires changes to the guest OS source code, which is not always possible. In particular, access to the source code of commercial OSs is heavily restricted. Both XenServer and KVM support paravirtualization.

Recent innovations in hardware, particularly in CPU, Memory Management Unit (MMU) and memory components (notably the Intel VT-x and AMD-V architectures [12]), provide some direct platform-level architectural support for OS virtualization. Hardware assisted virtualization offers one key feature: it avoids the need to trap and emulate privileged instructions by enabling guests to run at their native privilege levels. VMware and KVM support this approach.

B. *Live Migration*

Live migration is a mechanism that allows a VM to be moved from one host to another while the guest OS is running. This type of mobility provides several key benefits, such as fault tolerance, hardware consolidation, load balancing and disaster recovery. Users will generally not notice any interruption in their interaction with the application, especially in the case of non-real-time applications. However, if the downtime becomes too long, users of real-time applications, in particular interactive ones may experience serious service degradation [4].

To achieve live migration, the state of the guest OS on the source host must be replicated on the destination host. This requires migrating processor state, memory content, local storage and network state. The focus of our study is on network state migration.

Pre-copy is the memory migration technique adopted by KVM live migration, vMotion and XenMotion [13][28][27][32]. With this approach, memory pages belonging to the VM are transferred to the destination host while the VM continues to run on the source host. Transferred memory pages that are modified during migration are sent again to the destination to ensure memory consistency. When the memory migration phase is done the VM is suspended on the source host, and then any remaining pages are transferred, and finally the VM is resumed on the destination host [8]. The pre-copy technique captures the complete memory space occupied by the VM (dirty pages), along with the exact state of all the processor registers currently operating on the VM, and then sends the entire content over a TCP connection to a hypervisor on the other host. Processor registers at the destination are then modified to replicate the state at the source, and the newly moved VM can resume its operation [7][27][31].

The Kernel-based Virtual Machine (KVM) is a bare-metal (Type 1) hypervisor. The approach that KVM takes is to turn the Linux kernel into a VMM (or hypervisor). KVM provides a dirty page log facility for live migration, which provides user space with a bitmap of modified pages since the last call [5][6]. KVM uses this feature for memory migration.

VMware is bare-metal (Type 1) hypervisor that is installed directly onto a physical servers without requiring a host OS. In VMware vSphere, vCenter Server provides the tools for vMotion (also known as Live Migration). vMotion allows the administrator to move a running VM from one physical host to another physical host by relocating the contents of the CPU registers and memory [9][10].

XenServer is bare-metal (Type 1) hypervisor and runs directly on server hardware without requiring host OS. XenMotion is a feature supported by XenServer, which allows live migration of VMs. XenMotion works in conjunction with Resource Pools. A Resource Pool is a collection of multiple similar servers connected together in a unified pool of resources. These connected servers share remote storage and common networking connections [1][15][30].

KVM, VMware and XenServer aim to provide high utilization of the hardware resources with minimal impact on the performance of the applications running inside the VM. In this study, we compare their performance by measuring downtime and total migration time during live migration as well as their CPU utilization, when running large telecommunication applications in the VMs.

III. EXPERIMENTAL SETUP

Two HP DL380 G6x86 hosts have been used to test the performance of KVM and VMware ESXi 5.0. On top of the VMware ESXi 5.0, RedHat Enterprise Linux, Version 6.2 has been installed as a guest OS. The same hardware was used to test the performance of Xen for Linux Kernel 3.0.13 running as part of the SUSE Linux Enterprise Server 11 Service Pack 2. Each server is equipped with 24 GB of RAM, two 4-core CPUs with hyperthreading enabled in each core (i.e., a total of 16 logical cores) and four 146 GB disk. Both servers are connected via 1 Gbit Fibre Channel (FC) to twelve 400 GB Serial Attached SCSI (SAS) storage units. All devices are located in a local area network (LAN) as shown in Figure 1.

A. *Test Configurations*

Three different test setups were evaluated:

- KVM-based setup
- VMware-based setup
- XenServer-based setup

In each setup, two VMs are created inside hypervisor1 and hypervisor2, resulting in a total of four VMs (see Figure 1). One large industrial real-time telecommunication application is installed in the VMs. The application, referred to as server in the reminder of this paper, handles billing related requests. The server instances running on the VMs controlled by hypervisor1 are active in the sense that they are the primary consumers of the requests. The remaining two VMs under the control of hypervisor2 are running one passive instance of the server. Each active server is clustered together with one passive server. Thus, two clusters are created. Both the active and the passive server in a cluster can receive requests. However, all traffic received by the passive server is forwarded to the corresponding active server. The active server then sends the response back to the passive server.

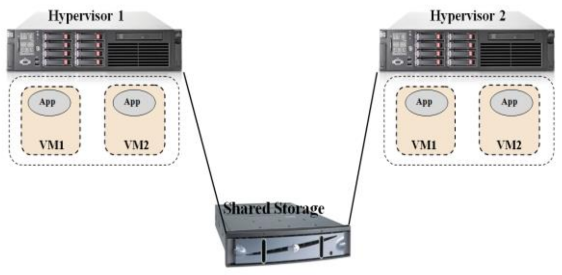


Figure 1. Network Plan

Finally, the passive server sends the response to the requesting system. Traffic going directly to the active server is handled without involving the passive server.
Another separate server runs a simulator that impersonates a requesting system in order to generate load towards the servers running in the clusters. The simulator is also located in the same LAN, but is not shown in Figure 1.

B. Test Cases

Two kinds of tests are considered in this study: performance tests and live migration tests.

1) Performance tests

In these tests, we vary the number of CPU cores (logical cores) in the VMs as well as the load towards the application.

We have three different core configurations: 6, 12 and 16 cores. For test cases with 12 cores and 16 cores the RAM for the VM is set to 24 GB, but for test case with 6 cores, the RAM size set to 14 GB for each of the VMs. This is an application specific setting that is recommended by the manufacturer. A single cluster is used for the case with 12 and 16 cores, respectively. Both clusters are used when testing the 6 cores configuration in order to assess the performance of two 6-core systems versus the performance a single 12-core system.

There are five load levels used in this test: 500, 1500, 3000, 4300, and 5300 incoming requests per second (req/s).

For each setup the following metrics are measured: CPU utilization, disk utilization and response time.

CPU utilization and disk utilization are measured inside the hypervisor on both servers using the commands presented in Table I. For disk utilization, we consider only write operations to the shared storage shown in Figure 1. The response time is measured inside the simulator as the duration from the instant a request is sent from the simulator to the application until the simulator receives the corresponding reply.

2) Live Migration tests

In these tests, we measure CPU and disk utilization during live migration. Four VMs with 6 cores CPU and 14 GB of RAM were created. For each configuration, a single VM (active server, e.g., VM1 on Hypervisor1 in Figure 1) is migrated from the source host to the destination host while the simulator creates a load of 100 req/s for the VM. At the same time the other VM (e.g. VM2 on Hypervisor1 in Figure 1) on the source host is receiving 1500 req/s. The other VMs (VM1 and VM2 on Hypervisor2 in Figure 1) on the destination host receive negligible traffic in the form of 100 req/s and thus are not completely idle.

TABLE I. CPU and DISK UTILIZATION COMMAND API

Virtualization System	Command Interface	
	CPU Utilization	***Disk Utilization***
KVM	ssh + sar	ssh + iostat
VMware	vCenter Server-performance graphs	vCenter Server-performance graphs
XenServer	ssh + xentop	ssh + iostat
Non-virtualized Server	ssh + sar	ssh + iostat

The application manufacturer considered this as a realistic example when one would like to migrate a VM to load-balance the system.

In addition to CPU and disk utilization, we measure the downtime and the total migration time. The total migration time is obtained from the hypervisor for KVM and XenServer, and from vCenter for VMware (see Table I). Downtime is defined as the time from the instant when the VM is suspended on the source host (Hypervisor1 in Figure 1) until the VM is restarted on the destination host (Hypervisor2 in Figure 1). We measured the downtime inside the simulator and our results indicate that it corresponds to the maximum response time of the application.

IV. COMPARISON BETWEEN KVM, VMWARE AND XENSERVER

In Section IV-A, the KVM, VMware and XenServer virtualization systems are compared in terms of CPU utilization (6 cores, 12 cores, and 16 cores), disk utilization and response time. These values have been measured for different loads (500, 1500, 3000, 4300, 5300 req/s) except for XenServer, which could not handle the highest load (5300 req/s).In Section IV-B, we compare the CPU utilization and the disk utilization during live migration, and in Section IV-C, we compare the total migration time and downtime of the VMs during live migration for the KVM, VMware and Xen Server setups, respectively.

A. CPU, Disk Utilization and Response Time (6 cores, 12 cores, 16 cores)

CPU and disk utilization are measured inside the hypervisors. We also performed the same measurements on the non-virtualized (target) server in order to establish a baseline for our results (see Table I). The response time is measured in the simulator.

As shown in Figure 2, Xen has the highest CPU utilization (approximately 80%) in the test case with 16 cores. Because of this high CPU utilization the application failed for traffic loads higher than 4300 req/s. KVM and VMware CPU utilization increases proportional to the load with an increase rate similar to that of the target. In Figure 3, we can observe that again Xen CPU utilization is significantly higher compared to VMware, KVM and the target in case of 12 cores. As shown in Figure 4, KVM, VMware and the target CPU utilization in case of 6 cores, are almost identical while Xen CPU utilization is the highest and at the highest point is around 70% which is the 20% higher CPU utilization compared to KVM, VMware and the target.

In Figure 5, we can observe that in case of 16 cores, VMware has the highest disk utilization, up to 25000 KB/s. KVM and Xen the disk utilization is linearly increasing with a rate similar to that of the disk utilization of the target. However, for KVM's and Xen's disk utilization is always around 2000 KB/s higher compared to the target. As shown in Figure 6, in case of 12 cores, Xen and KVM disk utilization is 5000 KB/s higher compared to the target while disk utilization for VMware is the highest, with a maximum around 30000 KB/s. In Figure 7, we can observe that VMware has the highest disk utilization compared to KVM and Xen, which show 34000 KB/s at the highest point. Xen's disk utilization in case of 6 cores is higher than KVM. The maximum disk utilization for Xen is around 25000 KB/s while the maximum KVM disk utilization is around 20000 KB/s. That is 5000 KB/s higher compared to the target but still the lowest compared to other virtualization systems.

Figure 8 shows that the response time of the application when using Xen is the highest for all traffic loads except for loads higher than 4300 req/s. Since for loads higher than 4300 req/s the application failed when using Xen, KVM has the highest response time after Xen, and at the highest point is around 25 ms in case of 16 cores. The response times of the application when using VMware is similar to the response times we had on the target. As shown in Figure 9, the response time of the application when using Xen reaches 26 ms at the highest point. In case of KVM the application has also high response times with a maximum of around 20 ms, which is higher than VMware's. The application response times when using VMware is similar to the response times of the application on the target. In Figure 10, we can observe that response time of the application when using Xen at the highest point is more than 25 ms, which is twice the application response time in case of the non-virtualized target. In the case of the 6 cores configuration using KVM the response time increases with a similar rate to the case when using VMware. However, for KVM and VMware the response times are around 5 ms higher compared to the target.

B. CPU, Disk Utilization and ResponseTime during Live Migration

CPU utilization is measured inside the hypervisors on both the source and the destination servers, during the live migration. Disk utilization is also measured inside both hypervisors. We initiate a migration after the system has been running for 15 minutes.

As shown in Figure 12, KVM's CPU utilization on the source is around 26% before the live migration begins. The CPU utilization on the destination is around 6%. After the live migration has been started, the CPU utilization first increases to 35% and then decreases to 18% on the source. However, on the destination server the CPU utilization settles around 13% after the live migration. As shown in Figure 12, VMware's CPU utilization before live migration is around 20% on the source hypervisor and around 4% on the destination hypervisor. When the live migration has been started, the CPU utilization on source increases to about 34% and remains at that level during the live migration. On the destination, the CPU utilization becomes around 15% after the live migration has started. After the live migration has stopped, the CPU utilization decreases to around 15% on the source hypervisor and to around 10% on the destination hypervisor. In Figure 12, we can observe that the Xen CPU utilization before live migration is around 34% on the source hypervisor and around 7% on the destination hypervisor. In the beginning of the live migration, the CPU utilization on source increases to around 40% and on the destination the Xen CPU utilization increases to around 13%. After the live migration is completed, the CPU utilization on the source decreases to around 29%, while on the destination's CPU utilization increases to 15%.

As shown in Figure 13, KVM's disk utilization on the source is around 10000 KB/s before live migration. On the destination, the disk utilization is around 6000 KB/s before live migration. After the live migration has started, the disk utilization on the source decreases to 9000 KB/s, while on the destination's disk utilization increases to 7000 KB/s. As shown in Figure 13, VMware's disk utilization is around 15000 KB/s on the source before live migration while on the destination the disk utilization is around 7000 KB/s. After the live migration, the disk utilization on the source decreases to around 13000 KB/s and on the destination it increases to around 9000 KB/s. In Figure 13 we can observe that the Xen disk utilization before the live migration is around 13000 KB/s on the source and around 6000 KB/s on the destination. When the live migration has started, the disk utilization increases to around 30000 KB/s on the source and to around 23000 KB/s on the destination. After the live migration has completed, the disk utilization on the source decreases to around 9000 KB/s and on the destination the disk utilization increases to around 10000 KB/s.

C. DowntimeandTotal Migration Time

The downtime has been obtained from the maximum response time, which is measured inside simulator during the live migration. Downtime corresponds to the time that application is not available and the VM is suspended.

As shown in Figure 11, the response time of the application when using KVM as hypervisor is around 1 ms before the live migration is started, but when the VM is suspended the response time increases to 700 ms. So the application was down for less than 700 ms. In Figure 11, we can observe that the response time of the application when using VMware as hypervisor is around 1 ms, but when the VM is totally down the application response time increases to 3000 ms. So the application downtime was around 3000 ms. As shown in Figure 11, before the live migration starts the application response time when using Xen is around 4 ms. When the live migration begins, the response time increases to 280 ms. So the application was down for less than 4 ms.

The total migration time is calculated inside the source hypervisor. It corresponds to the time that the VM started to be migrated until the complete VM state has been transferred to the destination hypervisor (see Figures 12-13). The total migration time for VMware, KVM and Xen is around 2 minutes.

V. RELATED WORK

In recent years, there have been several efforts to compare different live migration technologies. Xiujie et al. [1] compare the performance of vMotion and XenMotion under certain network conditions defined by varying the available bandwidth, link latency and packet loss. Their results show that vMotion produces less data traffic than XenMotion when migrating identical VMs. However, in networks with moderate packet loss and delay, which are typical in a Virtual Private Network (VPN), XenMotion outperforms vMotion in total migration time.

Tafa et al. [2] compare the performance of three hypervisors: XEN-PV, XEN-HVM and Open-VZ. They simulated the migration of a VM using a warning failure approach. The authors used a CentOS tool called “Heartbeat” that monitors the well-being of high-availability hosts through periodic exchanges of network messages. When a host fails to reply to messages the tool issues a failure notification that causes the hypervisor to migrate the VM from the “dead” host to one that is “alive”. Further, they compared CPU usage, memory utilization, total migration time and downtime. The authors have also tested the hypervisor’s performance by changing the packet size from 1500 bytes to 64 bytes. From these tests they concluded that Open-VZ has a higher CPU usage than XEN-PV, but the total migration time is smaller for Open-VZ (3.72 seconds for packet size of 64 bytes) than for XEN-PV (5.12 seconds for packet size of 64 bytes). XEN-HVM has lower performance than XEN-PV; especially regarding downtime. XEN-HVM had16 ms downtime while XEN-PV had 9 ms downtime for packet size of 64 bytes compared to our results with the large application we have got 300 ms downtime for Xen and total migration time of around 2 minutes.

In Chierici et al. [3] and Che et al. [29] present a quantitative and synthetically performance comparison between Xen, KVM and OpenVZ. They used several benchmarks (NetIO, IOzone, HEP-Spec06, Iperf and bonnie++) to measure CPU, network and disk accesses. According to their measurements, the OpenVZ has the best performance; also Xen hypervisor offers good performance while KVM has apparently low performance than OpenVZ and Xen.

There has been a similar study to our work carried out by Hicks, et al. [14], in which the authors focused only on memory migration and storage migration in the KVM, XenServer, VMware and Hyper-V virtualization systems. However, they did not consider CPU utilization of hypervisor during live migration in their study.

Clark et al. [27] introduced a method for the migration of entire operating system when using Xen as a hypervisor. They have tested different applications and recorded the service downtime and total migration time. Their results show 210 ms downtime for SPECweb99 (web-server) and 60 ms downtime for Quake3 (game server) during the migration.

Du et al. [24] proposed new method called Microwiper which makes less dirty pages for live migration. They implemented their method on the pre-copy based live migration in Xen hypervisor. They’ve tested two different programs with one with fixed memory writes and the other one with very quick memory writes. They compared the downtime and total migration time when using their method (Microwiper) versus the original Xen live migration (XLM). Their results show the original Xen live migration gets 40 ms downtime for VM memory size of 1024 MB when running quick memory writes program and total migration time of 11 seconds while their technique (Microwiper) decreases the downtime so it became around 10 ms but they got the same total migration time.

Web 2.0 application [33] has been evaluated by Voorsluys et al. [25] in terms of downtime and total migration time during live migration. They run XenServer as a hypervisor on their VM hosts. According to their experiments downtime of their system when serving 600 concurrent users is around 3 seconds and their total migration time is around 44 seconds which is much higher compared to our results because of the application that they’ve used also their setup is different.

Jo et al. [26] implemented a technique to reduce the duplication of data on the attached storage. They used different applications, RDesk I and II, Admin I, etc. and they measured the down time and total migration time during live migration when using XenServer as hypervisor. Their experiment shows 350 seconds total migration time for the original Xen live migration when the maximum network bandwidth is 500 megabits per second while using their proposed technique reduces this number to 50 seconds when duplication ratio is up to 85 percent.

VI. CONCLUSION AND FUTURE WORK

The results of the performance tests for different configurations of number of CPU cores show that KVM and VMware CPU utilization is almost identical and similar to CPU utilization on the target machine (non-virtualized) while XenServer has the highest CPU utilization with a maximum around 80%. In terms of disk utilization, the results indicate that KVM and Xen have similar disk utilization while VMware has the highest disk utilization (around 30000 KB/s for the highest load). The response time of the application is the highest when using Xen as hypervisor showing around 25 ms at the highest point. For KVM and VMware, the response time is almost similar (around 20 ms).

In general, KVM and VMware perform better in terms of CPU utilization while Xen CPU utilization is the highest. In terms of disk utilization KVM and Xen have similar performance while VMware has the highest disk utilization. Further, in terms of response time Xen has the longest response times compared to KVM and VMware.

As the results have shown, the CPU utilization during live migration is lower for KVM than for VMware while Xen had the highest CPU utilization during live migration. The disk utilization when KVM is used is 1000 KB/s lower compared to VMware during the migration.

For VMware, the downtime is measured to 3 seconds during live migration. For KVM and Xen the measured downtime are only 0.7 seconds and 0.3 seconds, respectively.

In general, the results presented in this study show that both VMware and KVM perform better in terms of application response time and CPU utilization for a configuration of two VMs with 6 cores each, compared to a configuration with a single VM with 16 or 12 cores. Xen’s performance is below that of the two other virtualization systems tested. However, Xen’s live migration technology, XenMotion, performs better than VMware’s vMotion and KVM live migration technology in terms of downtime.

REFERENCES

[1] F. Xiujie, T. Jianxiong, L. Xuan, and J. Yaohui, “A Performance Study of Live VM Migration Technologies: vMotion vs XenMotion,” Proceedings of the International Society for Optical Engineering, Shanghai, China, 2011, pp. 1-6.

[2] I. Tafa, E. Kajo, A. Bejleri, O. Shurdi, and A. Xhuvani, ”The Performance between XEN-HVM, XEN-PV And OPEN-VZ During Live Migration,” International Journal of Advanced Computer Science and Applications, 2011, pp. 126-132.

[3] A. Chierici and R. Veraldi, “A Quantitative Comparison between Xen and KVM,” 17th International Conference on Computing in High Energy and Nuclear Physics, Boston, 2010, pp. 1-10.

[4] D. Huang, D. Ye, Q. He, J. Chen, and K. Ye, “Virt-LM: a benchmark for live migration of virtual machine,” ACM SIGSOFT Software Engineering Notes, USA, 2011, pp. 307-316.

[5] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, ”KVM: the linux virtual machine monitor,” OLS, Ottawa, 2007, pp. 225-230.

[6] Red Hat. (2009). “KVM- Kernel based virtual machine,”[Online]. Available from: http://www.redhat.com/rhecm/rest-rhecm/jcr/repository/collaboration/jcr:system/jcr:versionStorage/5e7884ed7f00000102c317385572f1b1/1/jcr:frozenNode/rh:pdfFile.pdf
2014-03-10

[7] M.R. Hines and K. Gopalan, “Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning,” international conference on virtual execution environments, USA, 2009, pp. 51-60.

[8] P. Svärd, B. Hudzia, J. Tordsson, and E. Elmorth, ”Evaluation of delta compression techniques for efficient live migration of large virtual machines,” 7th ACM SIGPLAN/SIGOPS International conference on Virtual execution environments, USA, 2011, pp. 111-120.

[9] S. Lowe, “Mastering VMware vSphere 5,” Book, 2011.

[10] E. L. Haletky, “VMware ESX and ESXi in the Enterprise:Planning Deployment of Virtualization Servers,” Upper Saddle River, NJ: Prentice Hall, 2011.

[11] A.Kovari and P.Dukan, ”KVM & OpenVZ virtualization based IaaS Open Source Cloud Virtualization Platform,” 10th Jubilee International Symposium on Intelligent Systems and Informatics, Serbia, 2012, pp. 335-339.

[12] A.J. Younge, R. Henschel, and J.T. Brown, “Analysis of Virtualization Technologies for High Performance Computing Environments,” 4th IEEE International Conference on Cloud Computing, Washington, DC, 2011, pp. 9-16.

[13] A. Warfield, et al., ”Live Migration of Virtual Machines,” Proceedings of the 2nd conference on Symposium on Network Systems Design and Implementation, USA, 2005, pp. 273-286.

[14] A. Hicks, et al., “A Quantitative Study of Virtual Machine Live Migration,” Proceedings of the ACM Cloud and Autonomic Computing Conference, USA, 2013, pp. 1-10.

[15] J. Wang, L. Yang, M. Yu, and S. Wang, ”Application of Server Virtualization Technology Based on Citrix XenServer in the Infromation Center of the Public Security Bureau and Fire Service Department,” Proceedings of the Computer Science and Society, Kota Kinabalu, 2011, pp. 200-202.

[16] KVM. [Online]. Available from: http://www.linux-kvm.org/page/Main_Page
2014-03-10

[17] XenServer. [Online]. Available from: http://www.citrix.com/products/xenserver/overview.html
2014-03-10

[18] VMware. [Online]. Available from: http://www.vmware.com/
2014-03-10

[19] Ch. Cai and L. Yuan, ”Research on Energy-Saving-Based Cloud Computing Scheduling Strategy,” Journal of Networks, 2013, pp. 1153-1159.

[20] E.Michael and F. Janos, “A Survey of Desktop Virtualiztion in Higher Education: An Energy-and Cost-Savings Perspective,” 19th Americas conference on Information Systems, 2013, pp. 3139-3147.

[21] X. Li, Q. He, J. Chen, K. Ye, and T. Yin, “Informed Live Migration Strategies of Virtual Machines for Cluster Load Balancing” Proceedings of the 8th IFIP International Conference, 2011, pp. 111-122.

[22] Z. Wenyu, Y. Shaoubao, F.Jun, N. Xianlong, and S. Hu, “VMCTune: A Load Balancing Scheme for Virtual Machine Cluster Based on Dynamic Resource Allocation” Proceedings of the 9th International Conference on Grid and Cloud Computing, 2010, pp. 81-86.

[23] P. Riteau, C. Morin, and T. Priol, “Shrinker: Efficient Live Migration of Virtual Machines” Concurrency and Computation: Practice and Experience, 2013, pp. 541-555.

[24] Y. Du, H. Yu, G. Shi, J. Chen, and W. Zheng, ”Microwiper: Efficient Memory Propagation in Live Migration of Virtual Machines,” 39th International Conference on Parallel Computing, 2010, pp. 141-149.

[25] W. Voorsluys, J. Broberg, S.Venugopal, and R. Buyya, “Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation,” Proccedings of the 1st International Conference on Cloud Computing, 2009, pp. 254-265.

[26] Ch. Jo, E. Gustafsson, J. Son, and B. Egger, “Efficient Live Migration of Virtual Machines Using Shared Storage,” Proceedings of the 9th International Conference on Virtual Execution Environments, 2013, pp. 41-50.

[27] C. Clark, et al., “Live Migration of Virtual Machines,” Proceedings of the 2nd symposium on Networked Systems Design and Implementation, 2005, pp. 273-86.

[28] S. Akoush, R. Sohan, A. Rice, A.W. Moore, and A. Hopper, “Predicting the Performance of Virtual Machine Migration,” Proceedings of the 18th IEEE/ACM international symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, 2010, pp. 37-46.

[29] J. Che, Y. Yu, C. Shi, and W. Lin, “A Synthetical Performance Evaluation of OpenVZ, Xen and KVM,” Proceedings of the IEEE conference on Asia-Pacific Services Computing, 2010, pp. 587-594.

[30] S. Kikuchi and Y. Matsumoto, “Impact of Live Migration on Multi-tier Application Performance in Clouds,” Proceedings of the 5th IEEE international conerence on Cloud Computing, 2012, pp. 261- 268.

[31] H. Liu, H. Jin, Ch. Xu, and X. Liao, “Performance and Energy Modelling for Live Migration of Virtual Machines,” Proceedings of the conference on Cloud Computing, 2013, pp. 249-264.

[32] S. Kikuchi and Y. Matsumoto, “Performance Modelling of Concurrent Live Migration Operations in Cloud Computing Systems using PRISM Problemabilistic Model Checker,” Proceedings of the IEEE 4th international conference on Cloud Computing, 2011, pp. 49-56.

[33] L. Wang, et al., “Cloud Computing: a Prespective Study,” Proceedings of the New Generation Computing conference, 2010, pp. 137-146.

[34] J. Che, Q. He, Q. Gao, and D. Huang, “Performance Measuring and Comparing of Virtual Machine Monitors,” Proceedings of the 5th interantional conference on Embedded and Ubiquitous Computing, 2008, pp. 381-386.

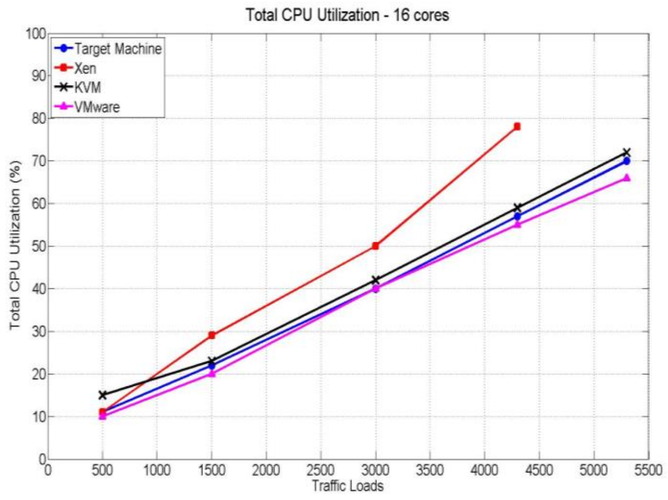


Figure 2. KVM, VMware and Xen CPU utilization for 16 cores

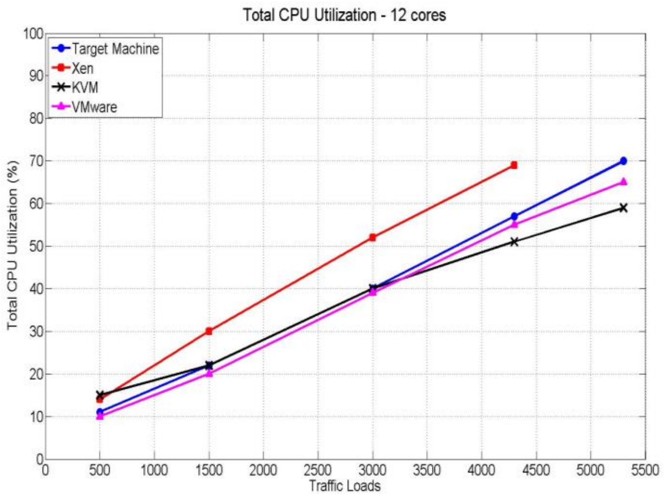


Figure 3. KVM, VMware and Xen CPU utilization for 12 cores

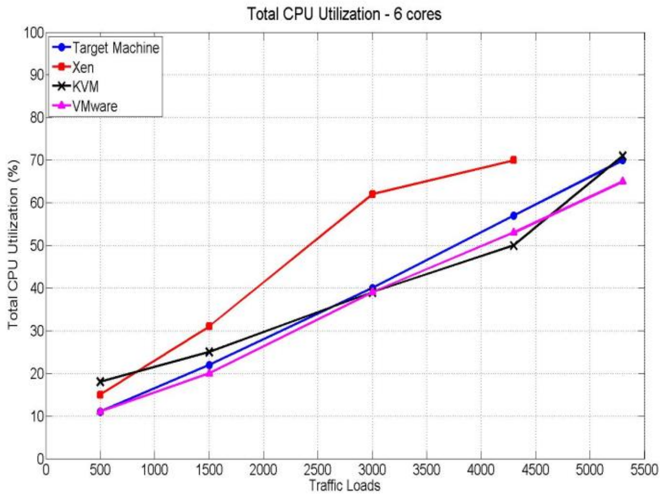


Figure 4. KVM, VMware and Xen CPU utilization for 6 cores

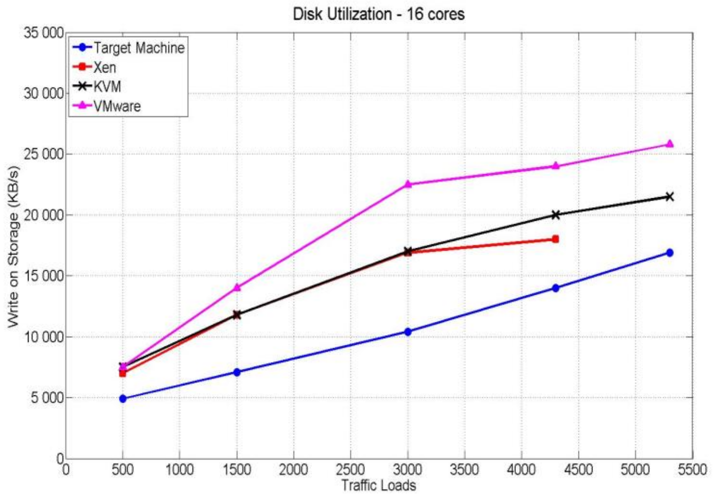


Figure 5. KVM, VMware and Xen disk utilization for 16 cores

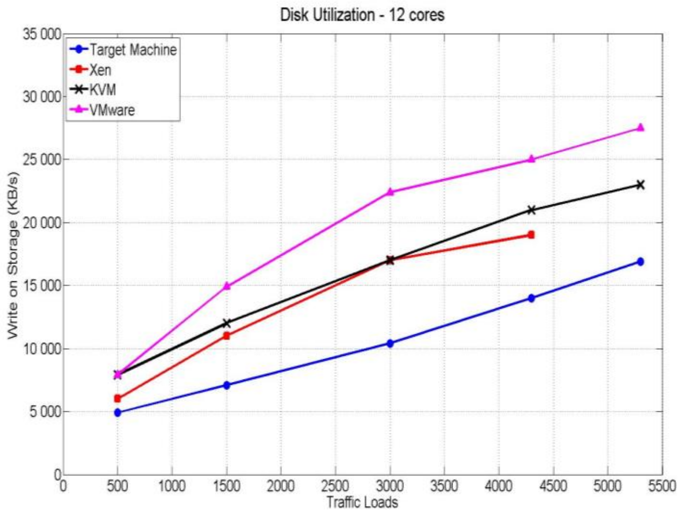


Figure 6. KVM, VMware and Xen disk utilization for 12 cores

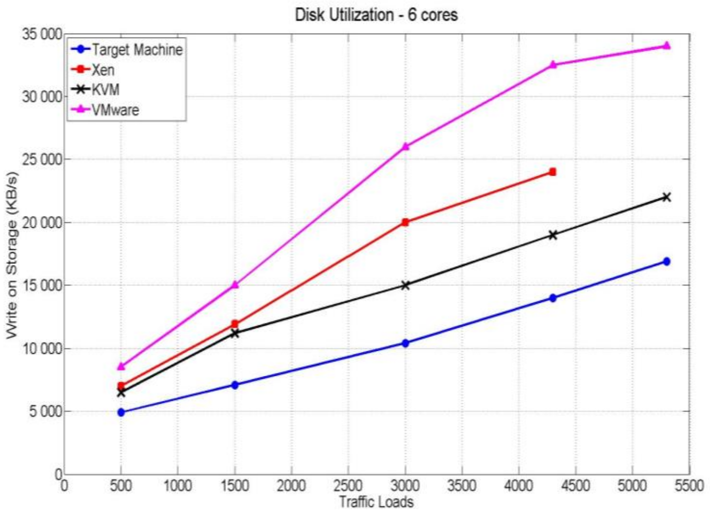


Figure 7. KVM, VMware and Xen disk utilization for 6 cores

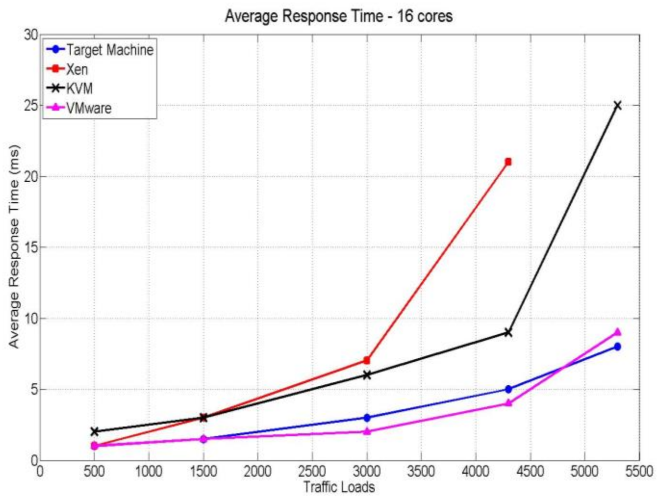


Figure 8. KVM, VMware and Xen response time for 16 cores

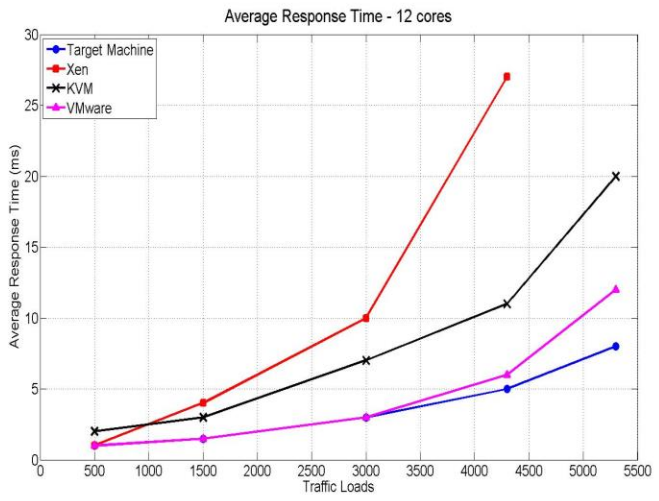


Figure 9. KVM, VMware and Xen response time for 12 cores

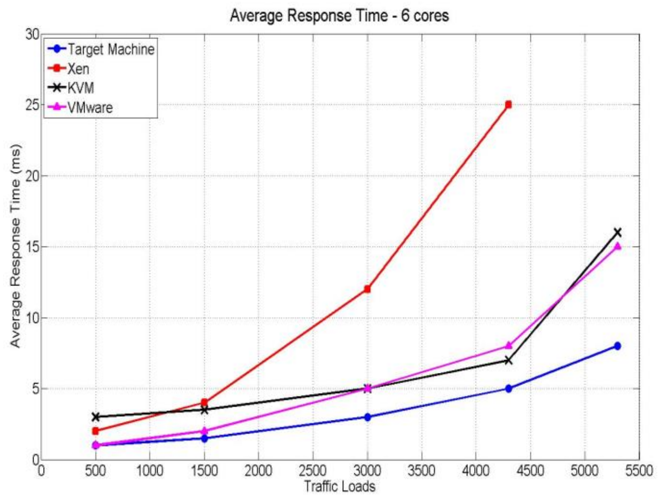


Figure 10. KVM, VMware and Xen response time for 6 cores

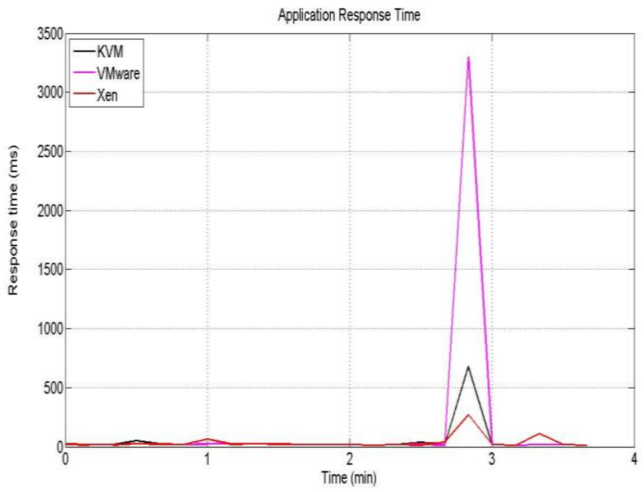


Figure 11. KVM , VMware and Xen response time during live migration

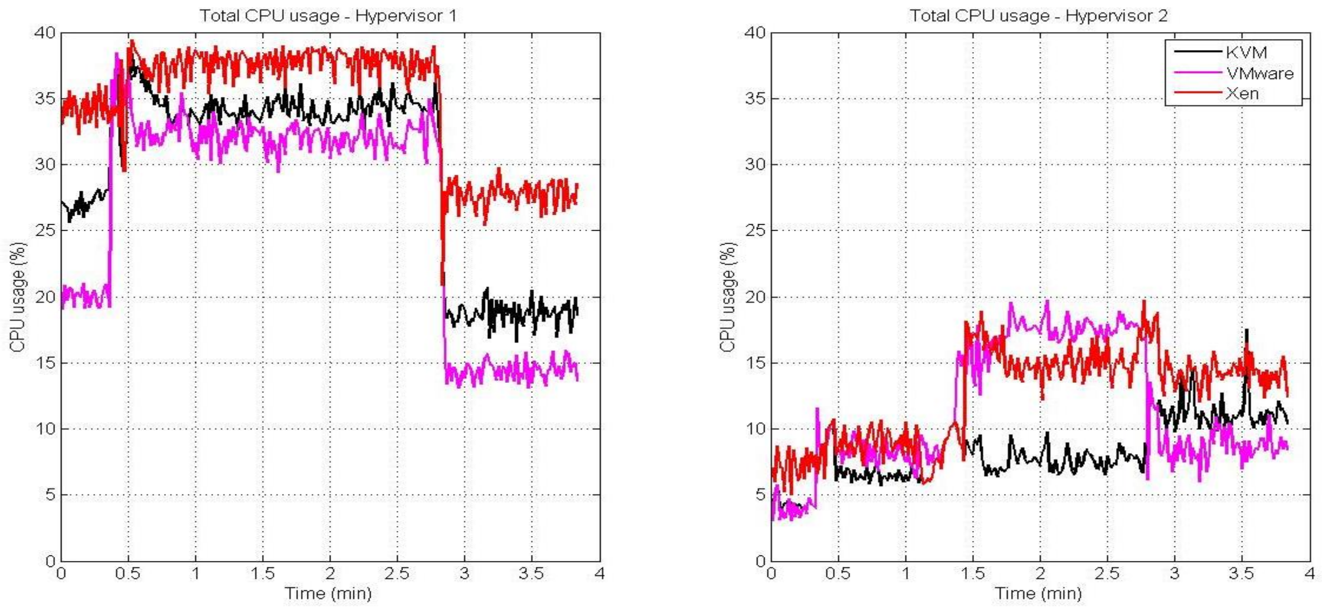


Figure 12. KVM , VMware and Xen CPU utilization during live migration

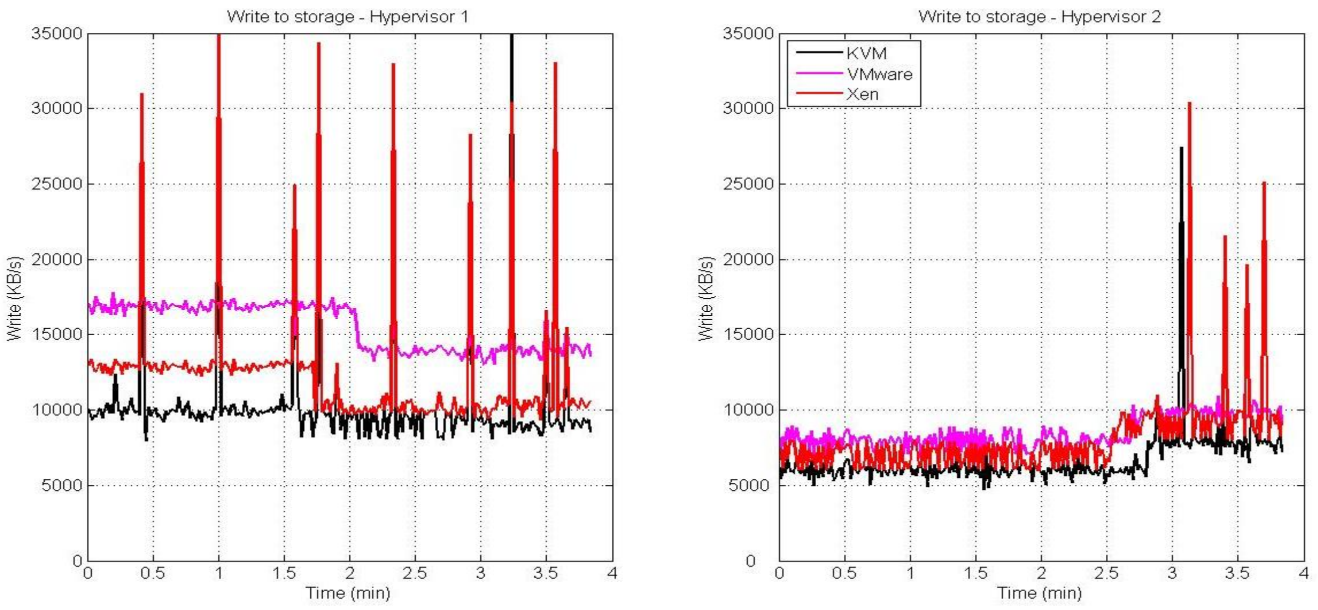


Figure 13. KVM , VMware and Xen disk utilization during live migration

Deployment of Secure Collaborative Softwares as a Service in a Private Cloud to a Software Factory

Guilherme F. Vergara, Edna D. Canedo, Sérgio A. A. de Freitas
Faculdade UnB Gama - FGA
University of Brasilia
Gama, DF – Brazil
gfv.unb@gmail.com,ednacanedo@unb.br, sergiofreitas@unb.br

***Abstract*—This paper presents a proposal of deploying secure communication services in the cloud for software factory university UNB (University of Brasília - Brazil). The deployment of these services will be conducted in a private cloud, allocated in the CESPE (Centro de Seleção e de Promoção de Eventos) servers. The main service that will be available is the EXPRESSO, which is a system maintained by SERPRO (Serviço Federal de Processamento de Dados). These services increase the productivity of the factory members and increase their collaboration in projects developed internally.**

Keywords-software factory; deployment of services; SaaS.

I. INTRODUCTION

The technology of cloud computing aims to provide services on demand, being billed or not by usage, as well as other basic services. Prior trends to cloud computing were limited to a certain class of users or focused on making available a specific demand for IT resources [1]. This technology tends to comply with wide goals, being used not only by big companies that would outsource all its IT services to another company, but also for user who wants to host their personal documents on the Internet. This type of technology allows not only the use of storage resources and processing, but all computer services.

In cloud computing, resources are provided as a service, allowing users to access without knowing the technology used. Thus, the users and companies began to access services on demand, independent of location, which increased the amount of services available [2]. With this, users are moving their data and applications to the cloud and can access them easily from any location.

Cloud computing emerges from the need to build less complex IT infrastructures compared to traditional, where users have to perform installation, configuration and upgrade of software systems, also infrastructure assets are inclined to become obsolete quickly. Therefore, the use of computational platforms of others is a smart solution for users dealing with IT infrastructure.

Cloud computing is a distributed computing model that derives characteristics of grid computing, with regard to the provision of information on demand to multiple concurrent users [2]. A cloud service provider offers cloud applications without the user having to worry about where the services are hosted or how they are offered. Slices of the computational power of the nodes of the network are offered, reducing costs to purvey own infrastructure to provide services. Resources are assigned only during the period of use, reducing power consumption when utilization is no longer needed.

Virtualization technology [6] provides the foundation for many cloud solutions. Moreover, in many solutions environments, where users are able to choose their virtualized resources, such as programming languages, operating system and other custom services are offered. The main benefits are reduction of the costs of infrastructure investment, operating costs and scalability for the provision of services on demand.

Cloud computing is an area that is increasingly growing and attracting diverse audiences. Ever more organizations has adopted cloud computing based solutions.

The objectives of this paper can be summarized by making a study of existing technologies in cloud computing with application to a software factory and contribute for a collaborative environment for software factory.

For a better understanding of this paper is explained now your organization. Section 2 provides a review of the concepts of cloud computing. Section 3 presents a set of possible implementations of solutions for factory software. Section 4 presents an implementation proposal for the software factory. Finally, Section 5 shows some results of this deployment.

II. CLOUD COMPUTING

Cloud computing refers to the use, through the Internet, of diverse applications as if they were installed in the user's computer, independently of platform and location. Several formal definitions for cloud computing have been proposed by industry and academy. We adopt the following definition: "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [3].

Cloud computing is being progressively adopted in different business scenarios in order to obtain flexible and reliable computing environments, with several supporting solutions available in the market. Being based on diverse technologies (e.g., virtualization, utility computing, grid computing, and service-oriented architectures) and constituting a completely new computational paradigm, cloud computing requires high-level management routines. Such management activities include: (a) service provider selection, (b) virtualization technology selection, (c) virtual resources allocation, and (d) monitoring and auditing in order to guarantee Service Level Agreements (SLA)

A solution of cloud computing is composed of several elements, as shown in Figure 1. These elements form the three parts of a solution cloud [6]. Each element has a purpose and has a specific role in delivering a working application based on cloud.

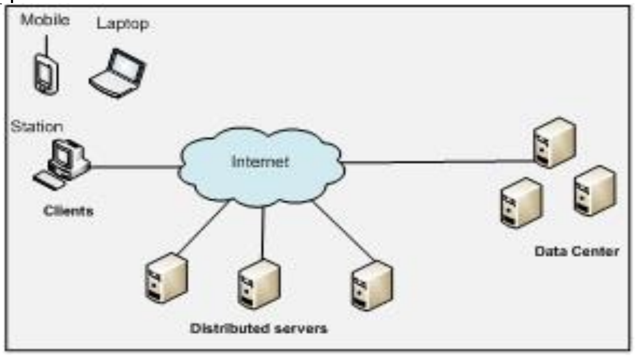


Figure 1. Three Elements of Cloud Computing Solution [6].

- Customers are in a cloud computing architecture, exactly what they are in a simple network. Are the devices with which end users interact to manage your information in the cloud.
- DataCenter is a set of servers where the application (Customer Relationship Management (CRM), financial, etc.) is stored. A growing trend in the IT is virtualization of servers, for example, the software can be installed allowing multiple virtual servers are used.
- Distributed servers: the servers do not have to be allocated in one location. Typically, servers are in different geographic locations, allowing the service provider more flexibility in options and security, for example, Amazon has a cloud solution worldwide. If something happens in one place, causing a failure, the service can be accessed through another site.[3]

A. Cloud Computing Architecture

Cloud computing architecture is based on layers. Each layer deals with a particular aspect of making application resources available. Basically, there are two main layers, namely, a lower and a higher resource layer. The lower layer comprises the physical infrastructure and is responsible for the virtualization of storage and computational resources. The higher layer provides specific services, such as: software as service, platform as service and infrastructure as service. These layers may have their own management and monitoring system, independent of each other, thus improving flexibility, reuse and scalability. Figure 2 presents the cloud computing architectural layers [4][5].

1) Software as a Service (SaaS): Provides all the functions of a traditional application, but provides access to specific applications through Internet. The SaaS model reduces concerns with application servers, operating systems, storage, application development, etc. Hence, developers may focus on innovation, and not on infrastructure, leading to faster software systems development. SaaS systems reduce costs since no software licenses are required to access the applications. Instead, users access services on demand. Since the software is mostly Web based, SaaS allows better integration among the business units of a given organization or even among different software services. Examples of SaaS include [7] Google Docs and CRM.

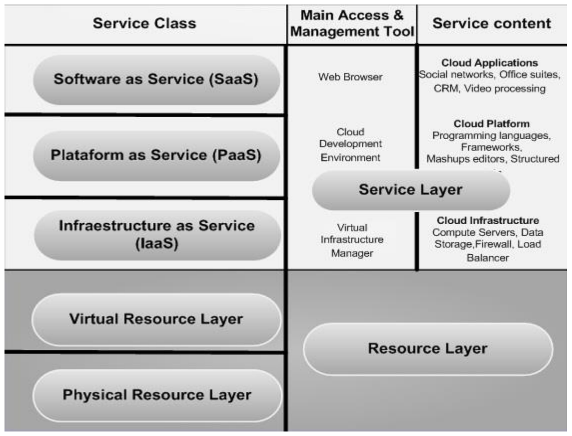

Service Class
Main Access & Management Tool
Service content
Software as Service (SaaS)
Web Browser
Cloud Applications
Social networks, Office suites, CRM, Video processing
Plataform as Service (PaaS)
Cloud Development Environment
Cloud Platform
Programming languages, Frameworks, Mashups editors, Structured
Service Layer
Infraestructure as Service (IaaS)
Virtual Infrastructure Manager
Cloud Infrastructure
Compute Servers, Data Storage,Firewall, Load Balancer
Virtual Resource Layer
Physical Resource Layer
Resource Layer

Figure 2. Cloud Computing Architecture [4]

2) Plataform as a Service (PaaS): Is the middle component of the service layer in the cloud. It offers users software and services that do not require downloads or installations. PaaS provides an infrastructure with a high level of integration in order to implement and test cloud applications. The user does not manage the infrastructure (including network, servers, operating systems and storage), but he controls deployed applications and, possibly, their configurations [4]. PaaS provides an operating system, programming languages and application programming environments. Therefore, it enables more efficient software systems implementation, as it includes tools for development and collaboration among developers. From a business standpoint, PaaS allows users to take advantage of third party services, increasing the use of a support model in which users subscribe to IT services or receive problem resolution instructions through the Web. In such scenarios, the work and the responsibilities of company IT teams can be better managed. Examples of SaaS [7] include: Azure Services Platform (Azure), Force.com, EngineYard and Google App Engine.

3) Infrastructure as a Service (IaaS): Is the portion of the architecture responsible for providing the infrastructure necessary for PaaS and SaaS. Its main objective is to make resources such as servers, network and storage more readily accessible by including applications and operating systems. Thus, it offers basic infrastructure on-demand services. IaaS has a unique interface for infrastructure management, an Application Programming Interface (API) for interactions with hosts, switches, and routers, and the capability of adding new equipment in a simple and transparent manner. In general, the user does not manage the underlying hardware in the cloud infrastructure, but he controls the operating systems, storage and deployed applications. Eventually, he can also select network components such as firewalls. The term IaaS refers to a computing infrastructure, based on virtualization techniques that can scale dynamically, increasing or reducing resources according to the needs of applications. The main benefit provided by IaaS is the pay-per-use business model [4]. Examples of IaaS [7] include: Amazon Elastic Cloud Computing (EC2) and Elastic Utility Computing Architecture Linking Your Programs To Useful Systems (Eucalyptus).

B. *Roles in Cloud Computing*

Roles define the responsibilities, access and profile of different users that are part of a cloud computing solution [8]. The provider is responsible for managing, monitoring and guaranteeing the availability of the entire structure of the cloud computing solution. It frees the developer and the final user from such responsibilities, while providing services in the three layers of the architecture. Developers use the resources provided by IaaS and PaaS to provide software services for final users. This multi-role organization helps to define the actors (people who play the roles) in cloud computing environments. Such actors may play several roles at the same time according to need or interest. Only the provider supports all the service layers.

C. *Cloud Computing Deployment*

According to the intended access methods and availability of cloud computing environments, there are different models of deployment [9]. Access restriction or permission depends on business processes, the type of information and characteristics of the organization. In some organizations, a more restrict environment may be necessary in order to ensure that only properly authorized users can access and use certain resources of the deployed cloud services. A few deployment models for cloud computing are discussed in this section. They include private cloud, public cloud, community cloud, and hybrid cloud.

Private: The cloud infrastructure is exclusively used by a specific organization. The cloud may be local or remote, and managed by the company itself or by a third party. There are policies for accessing cloud services. The techniques employed to enforce such private model may be implemented by means of network management, service provider configuration, authorization and authentication technologies or a combination of these.

Public: The infrastructure is made available to the public at large and can be accessed by any user that knows the service location. In this model, no access restrictions can be applied and no authorization and authentication techniques can be used.

Community: Several organizations may share the cloud services. These services are supported by a specific community with similar interests such as mission, security requirements and policies, or considerations about flexibility. A cloud environment operating according to this model may exist locally or remotely and is normally managed by a commission that represents the community or by a third party.

Hybrid: It involves the composition of two or more clouds. These can be private, community or public clouds which are linked by a proprietary or standard technology that provides portability of data and applications among the composing clouds.

III. Possible solutions for deployment

This section aims to investigate what are the possible options available in cloud computing to the university software factory. This chapter is divided by type of architecture, IaaS, PaaS and SaaS for a better understanding.

A. *IAAS*

IAAS solutions primarily aim to provide virtual machines with all the features of cloud for software factory.

1) Microsoft Windows AZURE: Windows Azure, which was released on February 1, 2010. Is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed datacenters. It provides both PaaS and IaaS services and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third party software and systems [11].

OpenStack: OpenStack is an open source software able to manage components of multiple virtualized infrastructures, like the operatinal system manages the components of a computers, OpenStack is called Cloud Operating System, to fulfill the same role in larger scale.

OpenStack is a collection of open source software projects that companies and service providers can use to configure and operate their infrastructure computing and cloud storage. Rackspace and NASA (the U.S. space agency) were the main contributors to the initial project. Rackspace provided a platform "Cloud Files" to implement the storage object OpenStack, while NASA entered the "Nebula" for implementing the computational side.

Eucalyptus: The Eucalyptus project [10] is an open source infrastructure that provides a compatible interface with Amazon EC2, S3, Elastic Block Store (EBS) and allows users to create an infrastructure and experience the cloud computing interface. The Eucalyptus architecture is simple, flexible, modular and contains a hierarchical design which reflects the common features of the environment.

Eucalyptus aims to assist research and development of technologies for cloud computing and has the following features: compatible with EC2, simple installation and deployment management using clusters tools, presents a set of allocation policies interface extensible cloud overlapping functionality that requires no modification on Linux environment, tools for managing and assisting the management of the system and users and the ability to configure multiple clusters, each with private internal network addresses in a single cloud.

B. PAAS

The PaaS solutions offer a platform for users to simply and quickly put their programs into production, thus providing an environment for quickly testing.

Tsuru: Tsuru is a open source and polyglot platform for cloud computing developed by globo.com since 2012 and it has began to be offered in a preliminary version in 2013 [12]. Like other platforms, the tsuru helps the development of web applications without the any charge of a server environment.

Tsuru uses Juju orchestration of services and takes advantage of the attractive features of its architecture. Supported programming development languages include Go, Java, Python and Ruby.

1) Heroku: Heroku [13] is a platform as a cloud service with support for several programming languages. Heroku was acquired by Salesforce.com in 2010. Heroku is one of the first cloud platformsa and it has been in development since June 2007, when it supported only the Ruby programming language, but since then added support for Java, Node.js, Scala, Python and Perl. The base operating system is Debian or in the latest Ubuntu based on Debian.

C. SAAS

1) Owncloud: The ownCloud is a free and open source web application for data synchronization, file sharing and remote storage of documents written in scripting languages PHP and Java Script. The Owncloud is very similar to the widely used Dropbox, with the primary difference being that ownCloud is free and open-source, thereby giving to anyone the option to install on your own private server, with no limits on storage space (except for hard disk capacity).

2) Expresso: Expresso [14] is a complete communication solution that brings together email, calendar, address book, instant messaging and workflow in a single environment. Because it is a custom version of the E-GroupWare, its development is also based entirely on free software.

IV. PROPOSED DEPLOYMENT

From the studies of cloud computing solutions, a proposal was created based on ease of deployment X relevance X time to deployment. For this paper, two software (SaaS) were chosen to contribute in collaborative software factory. Such software should help to ensure that members of the factory could share project documents, tasks, shared calendars and other tools for project management. An important factor to be considered during the time of adoption of a cloud service is the security. This security issue has attracted several discussions with the Brazilian Federal Government to the extent of having been issued a presidential decree (Decree No. 8.135, of November 4, 2013). The first article of this decree is as follows:

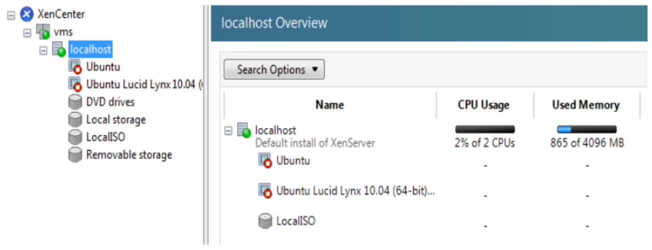
> *"Article 1 - Data communications direct, independent federal government and foundations shall be conducted by telecommunications networks and services of information technology provided by agencies or entities of the federal public administration, including public enterprises and joint stock companies of the Union and its subsidiaries."*

This article clearly shows the concern of the Government, which began to be widely commented after being publicize cases of espionage on the emails of the President of Brazil.

A. IAAS Proposal

The first option selected for the software factory was in IaaS solutions, i.e., to provide a software factory the entire necessary infrastructure in a transparent and scalable way. The software factory does not provide today's servers and storage required to keep, such hardware resources are still going through the bidding process. By aiming to solve this problem, the main solution is that use virtual machines until the factory have their own means of keeping it going. For the provisioning of virtual machines, the XEN Hypervisor [15] was chosen as a solution, because it has a large use in the market besides being open source and have already been studied previously.

Figure 3 shows the installation of two Linux virtual machines on the Windows client. One of the machines is a Linux server and the other one with an Ubuntu GUI.



Name	CPU Usage	Used Memory
localhost Default install of XenServer	2% of 2 CPUs	865 of 4096 MB
Ubuntu	-	-
Ubuntu Lucid Lynx 10.04 (64-bit)...	-	-
LocalISO	-	-

Figure 3. Xen Client.

The interesting aspect that can be seen in the Figure 3 is that we can control independently and completely the memory, the disk and the network; this way, we can have an idea if the VM was well provisioned.

But, in the course of the project, we came across a pleasant surprise; CESPE (Center of Selection and Promotion Events), which is one of the partners of the factory, provides virtual machines using the Windows solution, Microsoft Azure and then making XEN relevant only for research. These machines will be used for the allocation of software offered as a service (SaaS) to the factory.

B. PaaS Proposal

As a software factory, it is very interesting that the factory is able to produce prototypes as quickly as possible, because, as soon as the customer has a prototype, reduces the risks of software that does not add useful features to the client and sooner he will have a preview of the software that will be delivered. It can be very interesting to the factory because it will have a platform where you can quickly put the software into production and can show their customers.

For this type of service, both OpenShift and Tsuru tools were analyzed, but neither locally, although OpenShift proved to be a powerful tool for such functions. OpenShift supports major language currently used and is a leading solution Open Source.

C. SAAS Proposal

Lastly, we evaluated SaaS solutions and at the first view, these solutions did not prove to be relevant compared to IaaS and PaaS, because the Internet is full of them, such as Dropbox and Google Docs; but, one of the main reasons that lead to not use such software is the security that they do not provide. Because it is a software factory, is extremely important that their projects are not opened at all to other users. It is directly related to the decree mentioned before, SERPRO (Federal Data Processing Service) as the primary IT Company of the federal government, which was designed to provide such services and especially covering the second paragraph.

> "The agencies and entities of the Union referred to header of this article should adopt the email services and its additional features offered by agencies and entities of the federal public administration."

With this scope, Expresso [14] comes in as an email platform and other features that SERPRO is the main developer and it has an internal allocation of its resources, thereby increasing the security of the application.

As previously mentioned, the factory needs solutions that add value to collaborative work. As a solution to this problem, two solutions for SaaS are mentioned:

1) Expresso:

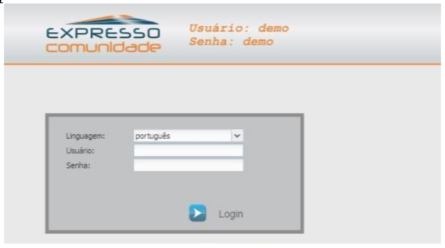
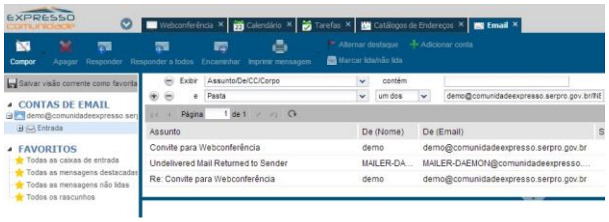


Figure 4. Expresso Log in.

Expresso, as mentioned in the previous section, is one complete communication solution, which includes email, calendar, address book, instant messaging and workflow in a single environment; in Figure 4, the log in of Expresso is shown. This solution greatly facilitates the work of the members of the software factory because they can set up meetings and schedule them on a shared agenda, facilitating the allocation of free time between them. Another very interesting service to be used by the factory is the video call, where project members can have meetings without leaving home, facilitating meetings and expediting meetings, which no longer need to be physically occurred. We can not forget of course the principal Expresso service, which is email, which will be much more secure than allocated in the internet environments, such as Gmail, Hotmail, etc. In Figure 5, the main screen of Expresso is shown.



Assunto	De (Nome)	De (Email)	S
Convite para Webconferência	demo	demo@comunidadeexpresso.serpro.gov.br	
Undelivered Mail Returned to Sender	MAILER-DA...	MAILER-DAEMON@comunidadeexpresso....	
Re: Convite para Webconferência	demo	demo@comunidadeexpresso.serpro.gov.br	

Figure 5. Main Screen of Expresso

Looking at the top of Figure 5, we can see all five tabs that represent the key features of Expresso. First, the selected tab shows the functionality of email with multiple filter options, favorites, etc. Beside the email tab, we have the address book, where all the user's contacts are saved. The next tab shows a very interesting feature that is the task manager where for example a team manager can delegate tasks to any of his members in a simple and fast way. Perhaps, after the email feature, the second most used feature used by the teams is the calendar functionality, because the managers can easily manage the agenda of each of the participants, and facilitate the allocation of meetings, deadlines, etc.

Lastly, we have the functionality of web conferencing, where members of a meeting for example, may join a video conference by simply accepting a request via email, facilitating the occurrence of meetings distributed teams or with difficulty time to face meetings.

2) Owncloud:

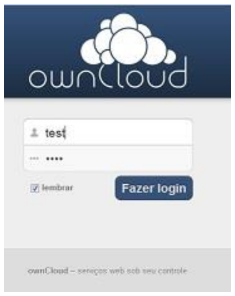


Figure 6. Owncloud Log-in

Besides the implementation of the email service, is intended to provide to members of the factory an archive, where it will be possible to share important documents in a safer environment then other solutions, because the files are storage in the factory servers.

The software chosen was the ownCloud, which is a similar solution as the Dropbox; ownCloud is a free open source software and has great community support; in Figure 6, the log in of ownCloud is shown. OwnCloud has as main feature the ability for the users to store their files, so that it can access on any computer with the client, both desktop and smartphones.

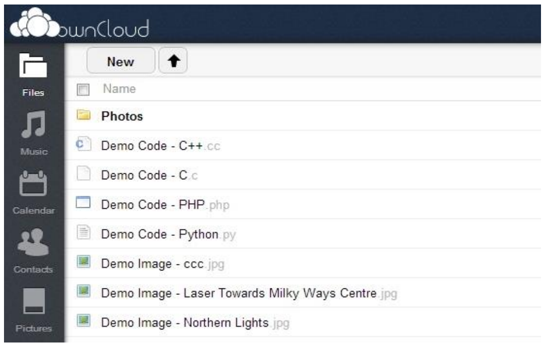


Figure 7. Main Screen of Owncloud

Looking at Figure 7, it is possible to notice the options allocation of ownCloud; the first file is an example of code written in C ++. This type of file can be viewed on its own web interface, already accepted by application and a similar visualization to the IDE for this extension and can it be edited directly from the browser, thus making it easier for programmers in the factory. The same applies for all other languages shown in Figure 7 and also to txt files, printing, and others. By clicking on the images, a pop-up appears with the selected image, facilitating the visualization of it. Lastly we have the songs that can be played directly from the internet.

All these files can be downloaded in two ways: First, when passed the mouse over the file you click on download, and the download is done. On the other hand, the second way, that is by downloading the client ownCloud, so you keep all your files updated.

A. *Deployment*

1) *Roles*

The implementation of Expresso and ownCloud in the software factory is a partnership between three institutions SERPRO, UNB and CESPE. The CESPE assumes the role of infrastructure provider, providing virtual machines and the entire necessary hardware infrastructure for deploying SaaS subsequently this work will be assuming the role of supplier of service provider, providing Expresso to the factory and its customers. This interaction can be best represented by Figure 8.

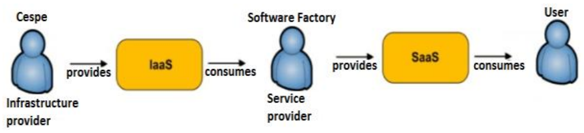


Figure 8. Roles in cloud computing in this context

2) *Process*

The deployment of services is a process that will take all the time devoted to a future work. Figure 9 proposes an initial process that can be appropriately adjusted within the first stage of this process.

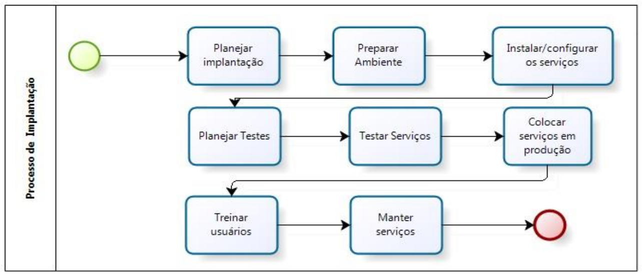


Figure 9. Process of deployment

a) *Implementation Plan*

This will be the beginning of the process for the services implementation; this phase is intended to study more "low level" of that material (storage, servers) and will be needed for deployment. In this case, we need a better study of the Expresso and the ownCloud to clarify what are the elements that need to be installed and configured.

b) *Prepare Environment*

At this stage, we already have a good overview of the elements necessary for the installation of the Expresso and the ownCloud. CESPE will help during this phase preparing and configuring all virtual machines.

c) *Install services*

This phase is intended to install/configure two services presented; this phase is perhaps one of the most time-consuming. Since there is not yet a broad understanding of the process of the installation of services, always unexpected can occur and end up taking a long time. In the case of Expresso, we have a greater ease, since they can have SEPRO aid.

d) *Test Plan*

After the services are already installed, it is planned to test the software; one can create a document specifying at this stage all the tests that will be performed, as well as expected results and comparing them with the results in the next stage.

e) *Test services*

After services are fully installed, you must run the tests planned in the previous step for problems that may have occurred in the installation, as a wrong connection with database, causing when people try to login or save your files. In this case you should review if the database has been properly installed and configured, and test again until all tests of test plan are completed

f) *Putting services into production*

This phase along with the installation is perhaps the most difficult and time consuming. This is because one must ensure that the services are running correctly, and ensure that clients are using them. It should always be close to customers looking to receive feedback on potential improvements and problems.

g) *Users training*

Right after the services have been put into production, users need a little coaching on how to use the tools. Since users are mostly students of software engineering, this phase will be very short and easy. However, as the services needed for future maintenance is also included in this phase the transfer of knowledge for any member of the factory, so it can continue the operation of the service. At this stage, we will have the help of SERPRO also, which may give workshops on the operation of the Expresso, thus arousing the interest of members of the factory proposing future functionalities for the tool.

h) Maintenance Services

As most software evolves and presents problems, there is always the need for continued maintenance and upgrades thereof. The members walked in the previous training will first take the maintenance services.

V. DEPLOYMENT RESULTS

After having defined the Express and the ownCloud as software as services to be deployed, it moved to the implementation step. Firstly CESPE released 3 virtual machines with Debian 7 operating system and Intel Xeon x5660 2.8 GHZ 4 cores and 4GB memory, two machines with 100GB HD and the third one with 1TB . In the first virtual machine was installed the Express, in the second one the OwnCloud in the third one the Postgres database, this organization can be seen in Figure 10. Today, 2 months after the installation, Expresso proved to be a powerful communication suite and the OwnCloud surprises by its functionality and interface very similar to the already known Dropbox. We opted for a centralized database for both applications, thereby centralizing all information; thus, it is safer to keep the data because you can restrict the access to this virtual machine, for example by closing all ports in the firewall, leaving only port 5432 open, hindering unauthorized access.

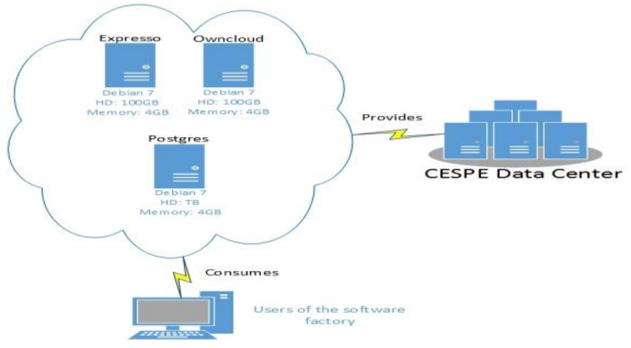


Figure 10. Physical architecture

VI. CONCLUSION

With each passing day cloud computing has been present in our lives, not only for personal use, but also increasingly in professional use. As much as cloud computing brings many benefits, it also brings many challenges. One of the major challenges is in relation to data security, especially after the scandals in Brazil and in the world in relation to privacy.

This paper provided an overview of tools that implement the three main types of cloud architecture; additioanlly, a proposal for the implementation of two of the tools analyzed, one ownCloud, which deploys a system of file sharing, and Expresso as email service.

Using these services is expected that the collaborative development of the factory becomes more facilitated and reliable with the exchange of files and emails, which will be more secure than allocated in the internet environments, such as Gmail and Dropbox.

ACKNOWLEDGMENT

Special thanks to FAPDF (Fundação De Apoio À Pesquisa Do Distrito Federal) for helping regarding the financial issue on this Conference in Venice. We also want to thank SERPRO and CESPE for helping us throughout this project.

REFERENCES

[1] M. Miller. (2008). Cloud Computing Web-Based Applications, That Change the Way You Work and Collaborate online, Que Publishing, Pearson Education, Canada.

[2] M. Zhou, R. Zhang, D. Zeng, and W. Qian, "Services in the cloud computing era: a survey," Software Engineering Institute. Universal Communication. Symposium (IUCS), 4th International. IEEE Shanghai, 2010, pp. 40-46. China.

[3] S. Uppoor, M. Flouris, and A. Bilas, "Cloud-based synchronization of distributed file system hierarchies," Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), IEEE International Conference, 2010, pp. 1-4.

[4] X. Jing and Z. Jian-jun," A Brief Survey on the Security Model of Cloud Computing," Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), Hong Kong IEEE, Aug. 2010, pp. 475-478.

[5] M. Zhou, R. Zhang, D. Zeng, and W. Qian, "Services in the cloud computing era: a survey," Software Engineering Institute. Universal Communication. Symposium (IUCS), 4th International. IEEE Shanghai, 2010. pp. 40-46. China.

[6] A. T. Velve and T. J. Elsenpeter, "Cloud Computing – Computação em Nuvem - Uma Abordagem Prática. Translator: Mei, Gabriela Edel. Publishr: Alta Books. Edio: 1ł, (2011), pp. 352-359.

[7] Z. Jian-jun and X. Jing, "A Brief Survey on the Security odel of Cloud Computing", Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science. 2010, pp. 475 – 478

[8] A. Marinos and G. Briscoe, "Community cloud computing, in First International Conference Cloud Computing", CloudCom, volume 5931 of Lecture Notes in Computer Science, 2009, pp. 472-484.

[9] P. Mell and T. Grance, The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology. Aug 2009.

[10] S. Liu, Y. Liang, and M. Brooks "Eucalyptus: a web service-enabled einfrastructure", Conference of the Centre for Advanced Studies on Collaborative Research, (CASCON 2007), October 2007, pp. 1-11

[11] Windows Azure - The Official Microsoft. [Online]. Available from: http://azure.microsoft.com/en-us

[12] Tsuru-A Open Source PAAS from Globo.com. [Online]. Available from: http://www.tsuru.io/

[13] Heroku. [Online]. Available from: https://devcenter.heroku.com/

[14] Expresso. [Online]. Available from: http://www.expressolivre.org/

[15] Xen Hypervisor. [Online]. Available from: http://www.xenproject.org/