Cloud-Based Software Engineering

PROCEEDINGS OF THE SEMINAR NO. 58312107

DR. JÜRGEN MÜNCH **5.8.2013**

Professor

Faculty of Science

Department of Computer Science

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

EDITORS

Prof. Dr. Jürgen Münch
Simo Mäkinen, Course Assistant

ABSTRACT

The seminar on cloud-based software engineering in 2013 covered many interesting topics related to cloud computing and software engineering. These proceedings focus on decision support for moving to the cloud, on opportunities that cloud computing provides to software engineering, and on security aspects that are associated to cloud computing.

Moving to the Cloud – Options, Criteria, and Decision Making: Cloud computing can enable or facilitate software engineering activities through the use of computational, storage and other resources over the network. Organizations and individuals interested in cloud computing must balance the potential benefits and risks which are associated with cloud computing. It might not always be worthwhile to transfer existing services and content to external or internal, public or private clouds for a number of reasons. Standardized information and metrics from the cloud service providers may help to make the decision which provider to choose. Care should be taken when making the decision as switching from one service provider to another can be burdensome due to the incompatibilities between the providers. Hardware in data centers is not infallible: the equipment that powers cloud computing services is as prone to failure as any computing equipment put to high stress which can have an effect on the availability of services.

Software Engineering – New Opportunities with the Cloud: Public and private clouds can be platforms for the services produced by parties but the cloud computing resources and services can be helpful during software development as well. Tasks like testing or compiling - which might take a long time to complete on a single, local, workstation - can be shifted to run on network resources for improved efficiency. Collaborative tools that take advantage of some of the features of cloud computing can also potentially boost communication in software development projects spread across the globe.

Security in the Cloud – Overview and Recommendations: In an environment where the resources can be shared with other parties and controlled by a third party, security is one matter that needs to be addressed. Without encryption, the data stored in third-party-owned network storage is vulnerable and thus secure mechanisms are needed to keep the data safe.

The student seminar was held during the 2013 spring semester, from January 16th to May 24th, at the Department of Computer Science of the University of Helsinki. There were a total of 16 papers in the seminar of which 11 were selected for the proceedings based on the suitability to the three themes. In some cases, papers were excluded in order to be published elsewhere. A full list of all the seminar papers can be found from the appendix. We wish you to have an interesting and enjoyable reading experience with the proceedings.

KEYWORDS

cloud computing, software engineering, cloud-based software engineering, software development

PAGES	**LANGUAGE**
76	English

Table of Contents

Part I: Moving to the Cloud – Options, Criteria, and Decision Making

Part II: Software Engineering – New Opportunities with the Cloud

Part III: Security in the Cloud – Overview and Recommendations

Appendix

Comparing preconditions for cloud and on-premises development

Teemu Mattila (Author)
Department of Computer Science
University of Helsinki
Helsinki, Finland
Email: teemu.mattila@helsinki.fi

***Abstract*—Before companies and developers can utilize the benefits of cloud computing there are some issues that need to be considered. These early or preliminary stages of cloud development deal for example with managerial, security-related and cost-effectiveness questions.**

The focus of this article is cloud economics and development in the cloud. From the cloud economics point of view, article compares the costs of cloud software development with traditional, on-premises development. There are many situations when using cloud can enable cost savings, but this is not always the case. From a development point of view article presents what needs to be considered, and what changes, when using cloud development instead of on-premises development. Though some things are common for both on-premises and cloud development, there are some differences and even limitations on when cloud cannot be used.

***Keywords*-cloud computing; cloud computing development; cloud vs on-premise development; cloud economics**

I. Introduction

There are many definitions for cloud computing, but in general it refers to both software and hardware resources that are delivered over the internet [1] [2]. These resources, more commonly known as services, are scalable, configurable, measurable, and easily accessible on-demand self-service resources [1] [3]. By using these services software development in the cloud can be cheaper, more efficient and more flexible way of producing new software than traditional on-premises software development.

Although cloud computing has many promising qualities, there are many important topics that need to be considered when deciding whether to start using cloud development. Cloud development is not always useful or automatically cost-efficient approach for all development requirements [4]. Also, all cloud environments are not intended for wider audiences [3] and all internet-based services cannot be considered automatically as complete cloud computing systems [1].

The cloud development approach also changes some aspects of the software development process in general [5]. In the extreme cloud development scenario, the only software a developer needs is a web browser and the only required hardware is a computer cabable of running the browser with a decent internet connection for connecting to cloud services [6]. However, this is not usually the case since the cloud environments do not provide all the required features. There are often specific requirements and in order to acquire them there may be need to combine different cloud services, use local development tools and communicate with cloud service providers [6].

This article is constructed as follows. Chapter two defines shortly the different cloud service and deployment models and also explains further what are the most essential topics that affect choosing cloud development approach over on-premises development. The third chapter, cloud economics, has the key focus of this article. It introduces different views of if and how cloud software development is more cost-effective than the on-premises development. Fourth chapter describes cloud adoption from software developer's point of view: what are the cloud development's strengths and weaknesses when compared to on-premises development. Also some topics that restrict the cloud software development are discussed. Finally, the last chapter summarizes the results and concludes the article.

The research method for this article is a literature review. The primary focus in this article are the cloud economics. Many other topics pointed out in this article, such as security and legal issues, are complex and could be analyzed at length. Yet for the purposes of this article they are discussed only in a cursory manner.

II. Background

Before analyzing suitability of cloud computing for software development it is useful to define the general service types and deployment models for cloud.

A. Cloud service models

The following three service models are often used when describing different kinds of cloud services: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [3].

Software as a Service, SaaS refers to all kinds of applications that are created by vendors into their cloud systems. These applications run inside the cloud: the end users can access them with various devices via different networks without a need to install the applications to their devices [3] [4].

Also, users are not owners of the services. The owner is the vendor who hosts them. Users, who subscribe these services, can then execute them as on-demand, scalable services. Users are charged either how much they use different resources (per-use basis) or by their subscription fee [4] [7].

In some definitions SaaS also refers to software components for application developers [8]. This is somewhat confusing with another common term, service-oriented architecture, or SOA. Indeed, Saas and SOA are not the same thing and it is useful to differentiate them [7]. SaaS should be described as a software delivery model: how complete software can be delivered to an end user. In this article the term SaaS is used in this manner. SOA should be considered as a software construction model. SOA components are reusable web-based services, building blocks for actual applications [7].

Platform as a Service, PaaS is a development platform hosted in the cloud and it is accessed via network [4]. The main difference between SaaS and PaaS is that SaaS hosts complete, unmodifiable services, whereas PaaS offers both complete and in-progress cloud applications that can be used to develop new software [3].

Infrastructure as a Service, IaaS is the third distinct cloud service model. IaaS involves physical computing resources, such as storage, networks and processing cababilities [3]. IaaS resources are often shared by many customers by virtualization systems. They can be scaled automatically based on resource demand of the customers.

For cloud users, IaaS is sometimes interrelated with PaaS and SaaS. Whereas PaaS provides means to create software than can be considered as SaaS, IaaS provides means to allow end user access to these new SaaS services [4].

B. Cloud deployment models

Another concept that needs to be defined is how the aforementioned services are deployed into cloud. There are four major models for this: private, community, public and hybrid clouds [9].

Private cloud is an entire cloud infrastructure that is created solely for a single organization. The ownership, management and operation responsibility of the cloud belongs to the organization, third party or some combination of them. [9]. The primary goal in this situation is not to sell cloud capacity through publicly accessible interfaces but to give local users a flexible and private infrastructure to run service workloads within their administrative domains. [10].

Community cloud is a deployment model where multiple organizations construct a cloud infrastructure together. They need to have similar usage, security and compliance considerations [9].

Public cloud is intended for open use by the general public. They are operated by commercial cloud providers, vendors, and they offer a publicly accessible remote interface for creating and managing cloud instances within their proprietary infrastructure [10].

Hybrid cloud is a system that combines infrastructures from two or more above-mentioned deployment models. They are separate cloud systems but are connected by interfaces that enable sharing or utilizing each others' data, application or computational resources [9] [10].

C. Reasoning for cloud development

The requirements for software development are extremely various. Some software can be built easily by one person with little resources. On the other hand there are software that is built by thousands of people internationally with various computational requirements. Still, there are some main areas that can be inspected when deciding whether to use cloud development or on-premises, local development.

The most important business question is if the cloud enables cost savings. From a business point of view, cloud computing can sometimes create overly optimistic expectations to business managers. Marston describes the expectations saying "the promise of cloud computing is to deliver the functionality of existing information technology (IT) and enable new features, yet at the same time it should dramatically reduce the costs of IT" [11]. This can probably be achieved only in rare cases. A more realistic question is whether the usage costs of, or transition costs to, external cloud will be low enough to benefit from any medium-term savings [4]. These economic considerations are discussed in-depth in chapter three.

Though cost efficiency is probably the most important factor, there are other areas to be considered. Another managerial question is that if and how the software development changes when using cloud. Cloud option is attractive if the quality delivered and the total cost is satisfying and the risks are reasonable [4]. This can lead to a more technical analysis about the cloud development's feasibility in a given situation and the possible problems when adopting cloud. These are discussed in chapter four.

III. Cloud economics

A recent study published in February 2013 by KPMG International shows that the cost reduction is clearly the most important objective for organizations' cloud adoption [12]. Almost half of the respondents that contained business and IT executives listed it as one key objective of their cloud strategy. 70% of those organizations who are using cloud answered that cloud is delivering efficiencies and cost savings today [12]. Though this is a high number there were some 20% who were not certain of any efficiencies or cost savings, and the rest responded that the cloud is actually hindering their efficiencies. This probably implies that cloud adoption is not automatically the best way for software development.

In order to compare cloud and on-premises development economics there is a wide variety of elements to consider. Costs for producing entire software include infrastructure and software development. These are explained in the first two sections of this chapter. For more detailed analysis, third section compares costs of different cloud deployment models and fourth section remarks about costs for migrating old software to cloud.

A. Infrastructure costs

In some cases infrastructure costs can be up to 60% of the total costs of the software development [4]. The costs can be divided into operational attributes and business premises. Operational attributes refer to three elements: hardware costs, software costs and license fees. Business premises are personnel expenses and costs of physical locations, such as rental and electricity costs [4].

Table I lists some of the required operational attributes. In the table the second column marks if given attribute is relevant when using cloud services and the third column marks if the attribute is relevant when using on-premises development. Note that the last row in table, software licenses for server, is dependant about selected software choices. When only open-source software is used, there are not necessarily any server side software license fees. Table is based on the works of Bibi et al. [4].

Table I
INFRASTRUCTURE COSTS

Operational attribute	Cloud	On-premises
Development devices (computers)	*	*
Peripheral devices (accessories)	*	*
Device maintenance	*	*
Server infrastructure	-	*
Server maintenance	-	*
Subscription fees	*	-
Server software licenses	*	*

All basic infrastructure costs are not averted by simply choosing the cloud approach. However, server infrastructure contains not only the server computers themselves but also the physical space, network connections, spare parts and maintenance personnel. Thus there are many cases that favor cloud IaaS model over conventional hosting. Five examples are listed in this section.

A first case is when demand for a service varies with time [1]. This can be e.g. if it is known that peak load occures only a few days per month or few months per year. A second case is when demand is unknown in advance [1]. For example new product can suddenly need a lot of resources if it becomes popular. These both cases benefit the easy scalability of cloud. IaaS cloud users do not need to buy enough server capacity for these peak load situations and thus there is no costly capacity overprovisioning [11].

Third case is when the use of batch processing can be divided to multiple instances [1]. The monetary cost is the same but results can be achieved significantly faster. For example, if one machine processes data for 1,000 hours, thousand cloud instances with similar computational capacities could process the same data in one hour while total cost remains the same.

Fourth case is that the cloud dramatically lowers the entry costs for smaller organizations that still need a lot of computational resources [11]. The purchase price for conventional data center server can be very high. Even when the need for high resources is regular, the monthly charges from cloud vendors can be easier to maintain than the start-up cost of dedicated server.

Last mentioned example is that by using IaaS cloud the need for technology management is potentially much simpler [11]. On the one hand, the physical server device maintenance is left to cloud vendor. Furthermore, even multiple server configurations and access controls can be handled with cloud vendor's management interfaces.

Table II lists some business premises than are a part of infrastructure costs. As with table I, the second column marks if given expense is relevant in cloud services and third column marks if it is relevant in on-premises development. Also this table is based on the works of Bibi et al. [4].

Table II
BUSINESS PREMISES EXPENSES

Business premises	Cloud	On-premises
Personnel expenses, salary	*	*
Personnel expenses, training	*	*
Electricity costs	limited	*
Physical locations, rental	limited	*
Network costs	*	limited

New personnel requirements are not directly affected by selecting IaaS cloud. In any case, part of development personnel needs to know about the underlying server-side software. Note that if the development can be done by using PaaS there could actually be less need for personnel training, because there is no need or access to low-level server-side software. By using IaaS cloud the electricity and rental expenses are most likely smaller. But if the developed system is very data intensive, the data transfer costs can be an important issue [1]. Currently transferring one terabyte of data from Amazon EC2 instance to internet costs from 50 to 120 dollars [13]. Cloud users need to optimize data placement and traffic at every level of the system in order to minimize costs [1].

B. Software development costs

Software development costs can be divided into four groups: product, platform, process and personnel attributes [4].

Product attributes include descriptive variables and size indicators [4]. Descriptive variables provide information about development type, application type and end-user type. Size indicators are often created with the help of function points (FP) or kilo-lines of code (KLOC) estimation. These attributes can be used together to indicate the complexity of given development task.

When estimating software's total complexity and size IaaS cloud approach does not seem to give any economical benefits over on-premise development. PaaS services do not automatically solve the complexity problems and even when they can be used they are probably suitable only for the simplest of applications [2]. For SaaS approach the situation is different. New services can be created by combining data or functionality from two or more existing services [11]. Because of the component reuse the size of the software in terms of FP or KLOC will reduce, but at the same time complexity of the project will multiply [6]. This is because the implementation details and integration requirements are often documented poorly or not at all.

Platform costs relate to technical, non-functional requirements for software [4]. Some examples include requirements such as software reliability, security issues and usability issues. Cloud can help small and medium sized organizations reach high infrastructural availability, such as service availability (uptime) and performance cheaper than creating their own infrastructure. For example, current Amazon Web Services Service Level Agreement (SLA) commits to an "annual uptime percentage" of 99.95% over the preceding year. Still, these same service levels provided by cloud vendors can be too low for mission critical applications and large organizations [11]. Also, the different IaaS vendors have different service level definitions and committed uptime percentages, so for this development cost it is not easy to call if the cloud approach is cheaper or not.

Process attributes refer to project supplements that enable the development and delivery of quality software within cost and time limitations [4]. Both cloud and on-premise development need some form of project management and management software. If cloud approach can change the process to be more streamlined and faster than on-premise development then it could be argued that also costs related to management are reduced.

The fourth cost type is the personnel attributes. Typical examples of this group are the experience of the team, the analysts capabilities, the familiarity with the programming language and the application [4]. These attributes apply to both cloud and on-premise development. While required expertise can be different, there are no real cost differences: personnel familiar with cloud development can be as expensive as personnel for on-premises development.

C. Costs for different cloud deployment models

The cloud deployment model can be a major factor when estimating total costs of the cloud development. Cloud development process can involve usage of distinct platforms from multiple vendors that are geographically dispersed around the world [6]. If for some reason an organization does not want to deploy its data or business logic into a public cloud, they may create their own cloud infrastructure for some or all of their needs [14]. However, the interoperability between cloud infrastructures is not easy to achieve [14]. And as a part of requirements gathering it may be needed to communicate with cloud vendors for more exact technical details and even some customizations [6].

Figure 1, improved from the works of Patidar et al., presents a relationship of software development complexity and infrastructure costs for cloud deployment models. The figure shows that using multiple (public) cloud vendors makes software development more complex and thus more expensive to develop. If again only private and hybrid clouds are used, the interoperability is easier, but the price for creating actual private cloud is high. Also it has been stated that except for extremely large data centers, private clouds provide only a subset of advantages of actual cloud infrastructure [1]. The hybrid model is the alternative that some organizations use to avoid large upfront investments while they still maintain some critical parts of their applications under more rigid control [14].

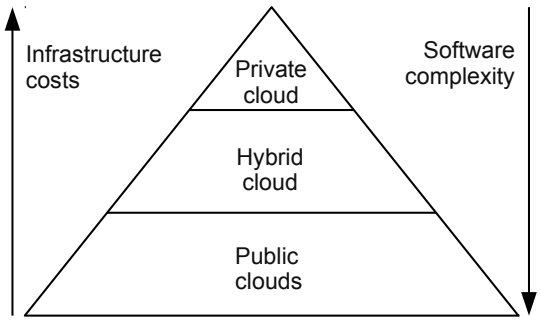


Figure 1. Comparing infrastructure costs to software complexity for different cloud deployment models

D. Costs for migrating legacy software to cloud

There are many applications and systems that have become outdated, either because they use unsupported technology or that they have been built to spesific, now outdated, platform or operating system. Some of these legacy systems are still actively used because costs for their modernization is high and the fact that they still provide essential services [14]. Most legacy and even some more recent systems are not easy to migrate into a cloud environment. There are two common methods to enable the transition. First way is a complete re-engineering where the system is reconstructed

to a new platform, but this approach is normally considered too expensive [14]. Another way is partial migration. The legacy system's infrastructure is moved to a cloud and then system is tested to ensure that it still functions normally [14]. When this can be done a complete re-engineering can be avoided making cost of the transition more tolerable.

IV. Other preconditions for cloud development

Economics do not cover all questions about feasibility of the cloud development. Other issues include the technical suitability, security, performance and process management. As is the case with cloud economics there are not always a conclusive answer to which development method is better, cloud-based or on-premises.

A. Cloud security

Security is one of the most often cited objection to cloud computing [1]. There are many both specific and general descriptions of cloud security questions. Here security issues are listed in a cursory manner.

Cloud users face security threats both from outside and inside the cloud [1]. The cloud users are responsible for application level security, while cloud vendors need to take care of the physical security [1]. Also, cloud users should be certain that other users cannot access their services and that the data is not accessed by unauthorized persons, including vendors' personnel [1].

All security problems that manifest in the cloud are not related to cloud at all [15]. In all development scenario the software developers are responsible from the application level security. To the same degree the physical security of the servers is the service providers responsibility, were it cloud environment or general hosting provider.

However, there are three security issues that can be described as both general issues in computer science as well as vulnerabilities in cloud computing's core technology. The primary security mechanism in cloud systems is virtualization. Virtualization is used to differentiate all running cloud instances and it also protects against most attempts by cloud users to attack eiter one another or underlying cloud infrastructure [1]. Still, there is a possibility that an attacker might successfully escape from a virtualized environment. Hence this vulnerability can be considered as intrinsic to virtualization and highly relevant to cloud computing [15].

Second problem relates to data privacy in public clouds. Users can try to make the reading of their data more difficult by encrypting the data before sending it to cloud. The problem is that cryptoanalytic advances can render any cryptographic mechanism or algorithm insecure as novel methods of breaking them are discovered [15]. Third issue relates to the fact that cloud applications are used mostly with web browsers with HTTP protocol. The HTTP protocol is a stateless protocol, whereas applications usually require some kind of session handling. This makes session hijacking and similar problems, whether caused by application or browser, a relevant issue for cloud computing [15].

B. Cloud interoperability

Currently, most cloud systems have their own way to offer their services to clients [3]. This can lead to vendor lock-in, which makes changing current service provider difficult. This also makes utilizing of multiple cloud systems difficult, because the same operations must be developed separately for each cloud. Also, integrating organization's own existing systems to proprietary cloud via vendor's application programming interface (API) can be difficult [3].

There are a few solutions proposed. One is to to standardize the APIs in such a way that a developer could deploy services and data across multiple cloud vendors [1]. Another one is to create intermediary layer between the cloud consumers and cloud spesific resources [3]. Sun Microsystems tried to create open API for cloud platforms [3] but the project was discontinued in 2010 soon after Sun Microsystem was acquired by Oracle Corp.

So far, the standardization and middleware layer proposals have not been widely adopted [16]. There are still many active projects related to ease cloud interoperability [16], such as European Commission supported mOSAIC Framework. However, since they are more or less work-in-progress, there is not yet one easy way to create cloud applications for multiple vendors' systems. It needs to be considered as a development threat, since lock-in issue can be critical if vendor decides to increase prices or encounters serious financial or technical problems that can lead to service interruptions.

C. Legal issues

Regulation issues that can happen in national or international level can make the cloud adoption problematic [11]. For example, regulation can impose requirements to data privacy, data access and data location requirements. The data in the cloud system can be distributed and it can be physically located to different country than its user. If this data is for example copyrighted, it is not clear which country's privacy laws should be followed [11]. Also, if there is a need for data auditing, such as some countries do for financial markets, the audit can be difficult to execute if the data is distributed to different countries [11].

D. Performance considerations

There are findings that sharing processing power and memory works well in virtualized cloud instances but that disk performance is more problematic [1]. As such the cloud user cannot be completely certain if their instance works always at the highest possible speed.

Transferring large amounts of data can be problematic, though this is relevant to both cloud and self-owned servers. If the data quantities needed to transfer are in terabyte

ranges, it can take many days to transfer the data via internet [1]. As a solution some cloud service providers have started to offer a service where the data is sent in physical storage devices to or from cloud vendor's data centers.

E. Applications unsuitable for cloud development

There seems to be very little research about the application types that are not optimal when created in cloud environments. Still, it is reasonable to estimate that all systems cannot be developed in cloud systems. The definition for cloud computing implies that cloud services are "services delivered over the network". As such, the software for devices without network connection cannot be completely developed in cloud, without at least on-premises testing with the physical devices.

Another obvious group is real-time systems. There are some definitions for real-time cloud systems [17] but in practice at least public clouds that are accessed via internet cannot be trusted to be real-time systems.

There are also some existing application types that are considered difficult to migrate to the cloud. Legacy systems, internally developed applications, mission critical applications and third-party software are examples of this group [11]. Also for some very large organizations that already have considerable server resources it may not be necessary nor economical to migrate their operations to public clouds for many years to come [11].

V. Conclusion

Software development in the cloud environment can be more efficient and more cost-effective than traditional, on-premises development. However, the cloud approach is not automatically suitable nor profitable for all kinds of software. Business and IT managers need to take into consideration many technical aspects before cloud development can be justified.

In summary, it can be argued that cloud approach is profitable to organizations who start to build new web-based services and who do not own any or have redundant server capacity. Also services that may encounter sudden or occasional increases in demand can benefit from the scalability of cloud systems.

On the other hand, public cloud systems do not necessarily suit systems that require extreme availability, need real-time computational capabilites or handle sensitive information such as data that legislation dictates to be privacy protected. Also if an organization already has large data centres with ample processing power, the need for public cloud is limited. But even within these situations cloud computing can sometimes be useful. Organizations can create their own private clouds to create similar scalable systems as public clouds. Also, they can utilize some services from the public clouds to improve their own systems and create some form of a hybrid cloud that best suits to their needs.

References

[1] M. Armbrust, A. Fox, R. Griffith *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672

[2] H. Erdogmus, "Cloud Computing: Does nirvana hide behind the nebula?" *Software, IEEE*, vol. 26, no. 2, pp. 4–6, 2009.

[3] T. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and challenges," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, 2010, pp. 27–33.

[4] S. Bibi, D. Katsaros, and P. Bozanis, "Application Development: Fly to the clouds or stay in-house?" in *Proceedings of the 2010 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, ser. WETICE '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 60–65. [Online]. Available: http://dx.doi.org/10.1109/WETICE.2010.16

[5] L. Cocco, K. Mannaro, and G. Concas, "A Model for Global Software Development with Cloud Platforms," in *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, 2012, pp. 446–452.

[6] S. Patidar, D. Rane, and P. Jain, "Challenges of software development on cloud platform," in *Information and Communication Technologies (WICT), 2011 World Congress on*, 2011, pp. 1009–1013.

[7] P. Laplante, J. Zhang, and J. Voas, "What's in a name? Distinguishing between SaaS and SOA," *IT Professional*, vol. 10, no. 3, pp. 46–50, 2008.

[8] S. Yau and H. An, "Software engineering meets services and cloud computing," *Computer*, vol. 44, no. 10, pp. 47–53, 2011.

[9] P. Mell and T. Grance, "The NIST definition of cloud computing (draft)," *U.S. National Institute of Standards and Technology special publication*, vol. 800, p. 145, 2011.

[10] B. Sotomayor, R. S. Montero, I. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *Internet Computing, IEEE*, vol. 13, no. 5, pp. 14–22, 2009.

[11] S. Marston, Z. Li, S. Bandyopadhyay *et al.*, "Cloud computing - The business perspective," *Decision Support Systems*, vol. 51, no. 1, pp. 176 – 189, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167923610002393

[12] K. International, "The cloud takes shape: Global cloud survey," 2013. [Online]. Available: http://www.kpmg.com/cloud

[13] "Amazon EC2 pricing," 2013. [Online]. Available: http://aws.amazon.com/ec2/pricing

[14] E. da Silva and D. Lucredio, "Software engineering for the cloud: a research roadmap," in *Software Engineering (SBES), 2012 26th Brazilian Symposium on*, 2012, pp. 71–80.

[15] B. Grobauer, T. Walloschek, and E. Stocker, "Understanding cloud computing vulnerabilities," *Security & privacy, IEEE*, vol. 9, no. 2, pp. 50–57, 2011.

[16] J. Miranda, J. Guillén, J. M. Murillo, and C. Canal, "Enough about standardization, let's build cloud applications," in *Proceedings of the WICSA/ECSA 2012 Companion Volume*, ser. WICSA/ECSA '12. New York, NY, USA: ACM, 2012, pp. 74–77. [Online]. Available: http://doi.acm.org/10.1145/2361999.2362011

[17] S. Liu, G. Quan, and S. Ren, "On-line scheduling of real-time services for cloud computing," in *Services (SERVICES-1), 2010 6th World Congress on*. IEEE, 2010, pp. 459–464.

Decision Making About Migrating To The Cloud Model

Emad Nikkhouy

***Abstract*—Today cloud and cloud computing become one of the hottest topics for research. There are different reasons for cloud popularity such as excessive scalability, reduced IT cost and accessibility. However, many companies still find it difficult to migrate their contemporary application to the cloud. The difficulty of migrating to cloud could be due to different reasons such as not knowing which cloud service or model to choose or being unaware of potential benefits and risks. In this seminar paper we are going to discuss about decision making in order to migrate to the cloud. First we are going to have a brief introduction about the cloud, followed by step by step decision making, then we will discuss potential benefits and risks that cloud might have, and finally we will discuss should the company migrate its legacy software to the cloud or not.**

Keywords: cloud, cloud computing, migration to cloud

I. INTRODUCTION

Migration of many IT companies to the cloud is predicted to be decisive but slow. Decision Support Systems (DSS) are playing an important role to simplify cloud adoption and migration. Whether migrate to the cloud or not, is one the most difficult decisions, which is made by Chief Information Officer (CIO) or IT manager. According to Saripalli and Pingali, researches show that there is a big concern 'why, where, when and what' type of tasks should be moved to the cloud [1].

Decisions concerning migration to the cloud are complicated since they are affected by multiple, conflicting principles such as quality of service (QoS) and service, where each of them has a significant effect on the enterprise bottom-line. Throughout this paper different aspects of decision making for migrating to the cloud will be analyzed [1].

II. STEP BY STEP DECISION MAKING

Cloud computing has many benefits, which interests many companies and organizations to migrate their existing software and solutions to the cloud. Nevertheless, most of the companies find it problematic and challenging in order to adapt cloud-based solutions, especially when it comes to migrating legacy software to public cloud providers [2]. This section, discusses a step-by-step method, which makes companies informed for choosing cloud selection and migration.

In this method, the company's key characteristics, focused application, and few potential cloud providers make a profile; then this profile is analyzed in order to discover constraints that prevent the company to migrate to the cloud. After analyzing the profile, discovered constraints can be resolved if possible, and the company can adopt a cloud solution, which suits the best company needs. According to Beserra, et al., this process is divided into nine activities and figure 1 shows the workflow [2].

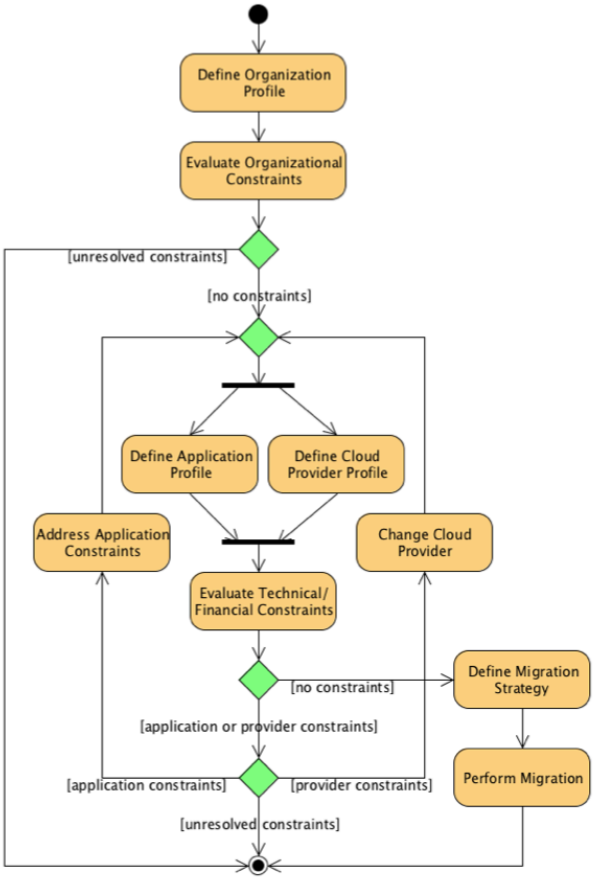


Figure 1, Step-by-step workflow [2]

Each of these activities are explained briefly in the upcoming subsections [2]:

1) Determine Organization Profile

The purpose of this section is to create a profile for the company in order to analyze constraints that might influence cloud migration and selection. For instance, these questions can be used in order to define company profile [2]:
What is the reason of the company for deciding to migrate to the cloud? Is it beneficial for the company to migrate to cloud? Is the company has some legal restrictions for the locations of its data? How experienced are IT personnel? How the company obtains and assigns its computing resources?

2) Evaluate organization Limitations

The aim of this process is to assess serious factors, which may hinder the company from migration. Therefore, an introductory evaluation is conducted in order to discover potential limitations, which are based on company's profile. Below examples may be used in order to identify possible constraints [2]:

- Insistence of employees for not migrating to the cloud, because they have fear of being dismissed after migration to the cloud.
- Lawful limitations for the physical location of data (i.e. government confidential data, which must be kept within national frontiers).
- Possibility of unauthorized access to important business data by third parties such as cloud provider.

3) Determine Application Profile

In this activity, characteristics such as usage and technical aspects of application are investigated, which can have influence on migration to the cloud. Here are some of the questions, which can be used in order to define these characteristics [2]:
What are the key characters of the application? How many users access the application? From which locations, users access the application? What times in a day the demand of usage is high and low? What is the required cost for running and maintaining the application? What architecture is used for building the application? What type of file system and database are used in the application to handle data? What operating system and environment is needed to run the application? Does quality-of-service (QoS) need to be precise? What is the least hardware configuration needed? What is the traffic usage of the application (sending and receiving)?

4) Determine Cloud Provider Profile

In this activity, a profile should be created for potential cloud providers in order to check whether it can satisfy constraints that company encountered. Below questions can be used in order to create cloud provider's profile [2]:
What kinds of services are offered by the cloud provider (i.e. platform as a service (PaaS), infrastructure as a service (IaaS) or software as a service (SaaS))? What type of resources such as virtual machines, development environment and storage space, is offered by the cloud provider along its service models? Is there any Service Level Agreement (SLA) offered from the cloud provider? How the prices are calculated (i.e. per hour reserved, per hour on demand or per bidding)? What security mechanisms are provided by the cloud provider? What other services are provided by the cloud provider (i.e. monitoring, auto-scaling, backup)? Does the provider give permission for accessing its operational logs for forensic or auditing purposes? What kinds of support services are offered by the cloud provider (i.e. phone call, email, online chat)?

5) Assess Financial and/or Technical constraints

The aim of this activity is the obedience between company profile, application profile and cloud provider profile. In this process seven main limitations are evaluated, which are including: organization limitations, communication limitations, security limitations, financial limitations, availability limitations, performance limitations and suitability limitations. Each of the above-mentioned limitations should be evaluated in the same context. As one can see in figure 2, these limitations are not totally independent from each other [2]. For instance, if company has some financial constraints then the performance it receives from cloud provider might be low, because if there is low budget for migrating to the cloud then company might not rent virtual servers as much as needed.

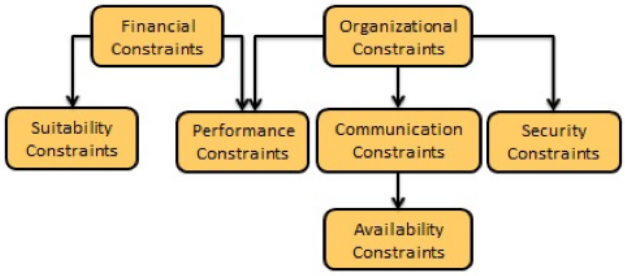


Figure 2, Limitation affect diagram [2]

6) Devise Application Constraints

In this section, any identified limitations from previous activity are tried to be eliminated. This can be done by increasing or reducing migration scope or by changing the application itself. In migration scope, only a part of the application component is moved to the cloud or additional components are moved along with the application. In changing the application, some codes can be modified if it is possible. After these modifications, application profile should be updated in order to restart the evaluation cycle of constraints. This cycle continues until there are no more

constraints or the developer reaches to a point that decides to abort migrating to the cloud due to some unresolvable constraints [2].

7) Change Cloud Provider

The aim of this activity is to change the cloud provider to another provider in order to resolve constraints that encountered in the previous activity evaluation. For instance, constraints that are due to operational cost can be resolved by finding a provider that has cheaper price. Similarly, if the constraint is due to the physical location of cloud provider regarding legal issues, a provider, which is inside boundaries, can be selected [2].

After the selection of new cloud provider, in order to restart the constraint evaluation cycle, a new cloud provider should be created. Same as devise application constraints stage, the evaluation cycle continues until the developer finds no more constraints or aborts the migration [2].

8) Determine Migration Strategy

If there are no more constraints that prevent application migration to the cloud, then this activity should be executed. Different guidelines such as SOA (Service Oriented Architecture) migration can be used to plan migrating to the cloud. Moreover, these questions can be used in order to plan the migration [2]:
What kind of activities should be accomplished in order to perform the migration? What strategies are most suitable for the migration?

9) Perform Migration

At this stage, the company should accomplish the real migration of application to the cloud based on strategy, which was defined in the previous activity [2].

III. Evaluation Of Benefits

A. Benefits of cloud models

There are three different models in cloud computing. In this section these three models and their benefits are discussed. In addition, some general benefits of cloud will be discussed.

1) Infrastructure-as-a-Service & its Benefits

Infrastructure-as-a-Service (IaaS) is related to servers, physical storage, networking components and processing capability, which enables the use of Software-as-a-Service and Platform-as-a-Service, or utility-like service for customers [3]. Thus, cloud computing mostly is self-managed by the customer itself on-premise private cloud.

Gibson, et al., described some of the highlights of IaaS [4]. The study included a project to construct hybrid cloud or community environment for the team of international physicists. These physicists needed high computational power for simulating and developing different models. Using the existing processors and resources, made it difficult for these physicists to operate. In order to conquer these challenges the team developed a cloud infrastructure project based on IaaS model. In order to reduce latency and improve end-user experience for users that are in different countries, nodes were located in different regions of the world [4].

The IaaS had some benefits for this team of physicists, which can demonstrate that how cloud computing can have benefit for communities. Here are some of the advantages and benefits that team of physicists gained from IaaS [4]: The nodes that these physicists were working on, were geographically scattered in order to reduce latency and improve the end user experience that lived in different regions and countries around the world. Moreover, due to having cluster, necessity of purchasing individual workstations every time was eliminated. For the maintenance point of view these physicists no longer needed to maintain the hardware, since the cloud provider handled this job. Therefore, as a result this group of scientists had access to enormous resources in order to run their simulations instead of waiting for grid computing.

2) Platform-as-a-Service & its benefits

Platform-as-a-Service (PaaS) is a platform for developing on a cloud through the network. PaaS provide required tools for developers to build web applications without any required tools installed on their own space. Microsoft azure, SalesForce.com and Google app engine, are some examples of PaaS providers [3].

PaaS is built on top of IaaS, thus it has many benefits, which are same as IaaS, such as [4]:

- Dynamic resource allocation
- Hardware virtualization
- Utility computing
- Low investment cost
- Reduced setup, maintenance and administration time
- High availability
- Reduced processing time by running the application through parallel processors

3) Software-as-a-Service & its benefits

Software-as-a-Service (SaaS) provides service or software to users via network. Both vertical and horizontal market software are provided to the users through Service Level Agreement (SLA). Some examples of vertical SaaS are specialized software such as management information system, Accounting software and Customer relation Management system. Moreover, some examples of horizontal SaaS are search engine, subscription management software, office suits and mail servers [3].

SaaS pricing model is based on pay per use, which generally are calculated based on customization costs, training costs, user's license and user's support. These costs are described in SLA, which defines base on pay on demand

[2]. SaaS can have many advantages, as it can relate to cost saving and budgeting for a company [4]. One of the major benefits of SaaS in deploying applications is low initial investment cost on hardware, software and staff [4]. According to Hurwitz and Associates, SaaS solutions offered 64% of saving over 4 years compare to on premise solution [6]

B. General Benefits of Cloud

Khaje Hosseini, et al., [5] assessed some benefits of cloud. In this section we will discuss some of these benefits.

According to Khaje Hosseini, et al., there are three different types of benefits: Technical, Financial and Organizational benefits [5].

a) Technical

- Response time is reduced because of extensive computational resources. For instance, when execution of a series of programs takes 1000 hours with one computer, the same job can be done in one hour using 1000 computers with the same price [4].
- At anywhere, anytime with any device (i.e. laptop, tablet, mobile) computational resources can be accessed, which facilitates cooperation between users and also facilitates maintenance and application support.

b) Financial

- Costs are decreased because of more efficiency in operations and less infrastructure maintenance. Moreover, the economy of scale, which can be attained by cloud providers, can reduce the costs.

c) Organizational

- With cloud, IT personnel do not need to care about hardware maintenance anymore; therefore they can focus more on value-added activities.
- Opportunity for the company to propose more products or services to the users in order to increase the level of their interest.

IV. Evaluation Of Risks

In the previous section three different cloud models, and some general benefits of cloud have been discussed. In this section risks of IaaS, PaaS, SaaS and some general cloud risks will be studied.

A. Risks of cloud models

1) Infrastructure-as-a-Service Risks

Security is an issue in IaaS, especially because other models are operating on top of IaaS. In a case when there is a shared environment and one company is hosting many other companies' data, all the parties may be at risk of privacy or security incidents. For instance, virtualization of the hypervisor gives access to hardware's physical resources. If hypervisor gets compromised, it is feasible to capture memory contents, virtual network traffic, and any other types of communications, which are under its domain [4].

In order to conquer above mentioned security issue, assigning roles to employees, using detailed logging, and also exerting security principle of minimum privilege is good inception [4].

2) Platform-as-a-Service Risks

One of the challenges of PaaS customers is that developers do not develop their applications on new platforms due to the rapid growth of these platforms and also uncertainty of developers about the future. However, these anxieties should be reduced by the time providers get popular [4]. According to Gibson, et al., currently, it is feasible to use additional APIs and middleware in order to develop the application on provider-independent platform, which gives the user this option to select cloud provider that their application can be deployed, such as Cloud Foundry and OpenShift [4].

Compatibility is the second concern that may be confronted when using PaaS. Different PaaS providers may have different types of language, middleware, database or APIs software, which make it difficult for users to choose the right platform or to switch to another platform [4].

Like IaaS, security is a big issue in PaaS. Public cloud is not like enterprise infrastructure, and it restricts customers from securing their data. In order to operate effectively, platforms must enhance privileges. Thus, the PaaS provider must strictly limit these privileges so no consumer can get access to another consumer's platform, data, memory or network traffic [4].

3) Software as a Service Risks

In SaaS while accessing data at anywhere and anytime is convenient and decreases the need for carrying sensitive data, however, a non-secure endpoint can be a high risk [4].

According to Gibson, et al., in order to reduce the risks that consumer may encounter using SaaS solution, they have to ask these questions when searching for a provider [4]:

- Which SaaS personnel have full access to the database?
- Is client data separated?
- What kinds of security controls does the provider use?
- Is data encrypted?
- What are the SLAs (Service Level Agreements)?
- What kinds of data are saved in audit logs?

B. General cloud Risks

Khaje Hosseini, et al., assessed some potential risks of cloud. They divided risks into five categories: Organizational, Legal, Security, Technical and Financial. In this section we will briefly discuss some of these risks and an approach to alleviate the risk [5].

a) Organizational

- Staff productivity is decreased during the migration because staff would have this anxiety that they might be dismissed after the migration.
  Alleviation approach: Company should make sure that experts do not get dismissed.
- Difference between existing error handling methods and cloud provider error handling methods. In case of occurring error, there is a limited response from the organization due to lack of information or no access to the cloud's error report data or vulnerability information.
  Alleviation approach: Should check the cloud provider's SLA and confirm that it has precise error classification systems and reporting technique. For instance, what is reported? To whom it is reported? And how quick it is reported?

b) Legal

- No agreement on data confidentiality rules. For instance, cloud provider can access data without permission.
  Alleviation approach: Should use encrypted data storage and transfer.
- Unable to use traditional software licenses, because the licensing agreement was based on per-seat or per-CPU.
  Alleviation approach: Check all the software license agreements.

c) Security

- Vulnerability of browser can become more important.
  Alleviation approach: Make sure to update browsers time to time.
- Denial of sevice attack can make the resources unavailable.
  Alleviation approach: Use tools, which monitor network.

d) Technical

- Performance is not what was expected. For instance, input/output and network data transfer latency or CPU clock rate.
  Alleviation approach: Should use benchmarking tools to examine the performance of cloud before making any decisions. In order to reduce network latency or transfer rate, use physical disk shipping, and to deal with CPU clock rate rent more Virtual Machines or higher spec ones.
- Collaboration issues between clouds due to incompatibility between cloud providers' platform.
  Alleviation approach: Use cloud middleware to solve collaboration issues.

e) Financial

- Actual cost might be different from the expected cost due to inaccurate resource estimates; providers increase their price or inferior performance because of over-utilized servers, consequential need more resources than expected before.
  Alleviation approach: In order to estimate the cost accurately, monitor existing resource usage and use estimation applications.
- Augmented cost because of complex integration. Incapability to decrease costs because of the unrealizable reduction in system support personnel.
  Alleviation approach: Explore system integration issues and prevent migration of extremely interconnected system primarily.

V. MIGRATE TO THE CLOUD OR NOT

Migrating to the cloud or not is not a question, which can be answered easily. Therefore, based on what discussed and described earlier, companies have to analyze everything from top to down and go through every details in order to gather all the necessary information to decide whether it is beneficial for them to migrate or not.

For some companies it might be very challenging and costly to migrate their legacy software to the cloud, because they might need to reengineer the whole application in order to be able to migrate to the cloud. Therefore, with all of the challenges, they should have long-term evaluation in order to decide to migrate or not. On the other hand, for some companies it might be easier to adapt cloud computing.

VI. GLOBAL CLOUD NETWORKING SURVEY

According to Cisco survey, which was held on 2012, if the companies have only one choice for moving an application to the cloud the first choice would be storage, followed by Enterprise Resource Planning (ERP), Email and collaboration. But in reality, when they are asked which applications are already moved, or they have plan for moving them to the cloud in the next year, most of IT

decision makers named Email and Web services, followed by storage and collaboration solutions such as instant messaging and web conferencing [7].

For those companies that are on process of migrating to the cloud, availability/reliability of cloud application was mentioned as one of the highest network challenges for hindering a successful implementation of the cloud. Below figure shows percentage of these challenges [7].

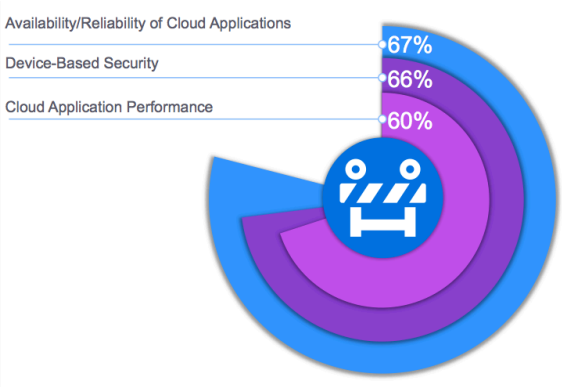


Figure 3, Network Challenges of Migrating applications to the cloud (Percentage)

On 2012 only 5% of IT companies were able to migrate at least half of their applications to the cloud, while by the end of 2012 it was expected that this number increase to 20% [7].

VII. Conclusion

Cloud computing has become very popular due to all the benefits that it has to offer. Many companies find it interesting and beneficial for them to migrate their software on the cloud, however, they find it challenging. Companies that are interested to migrate to the cloud should perform some researches to analyze what steps they need to take in order to discover all of the constraints and resolve them. They have to take all the benefits and risks into account before making any critical decisions.

VIII. Refrences

[1] Saripalli, P.; Pingali, G., "MADMAC: Multiple Attribute Decision Methodology for Adoption of Clouds," Cloud Computing (CLOUD), 2011 IEEE International Conference on, pp.316,323, 4-9 July 2011

[2] Beserra, P.V.; Camara, A.; Ximenes, R.; Albuquerque, A.B.; Mendonca, N.C., "Cloudstep: A step-by-step decision process to support legacy application migration to the cloud," Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2012 IEEE 6th International Workshop on, pp.7,16, 24-24 Sept. 2012

[3] Bibi, S.; Katsaros, D.; Bozanis, P., "Application Development: Fly to the Clouds or Stay In-house?," Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on, pp.60,65, 28-30 June 2010

[4] Gibson, J.; Rondeau, R.; Eveleigh, D.; Qing Tan, "Benefits and challenges of three cloud computing service models," Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference on, pp.198,205, 21-23 Nov. 2012

[5] Khajeh-Hosseini, A.; Sommerville, I.; Bogaerts, J.; Teregowda, P., "Decision Support Tools for Cloud Migration in the Enterprise," Cloud Computing (CLOUD), 2011 IEEE International Conference on, pp.541,548, 4-9 July 2011

[6] Amazon, "Advantages of SaaS Based Budgeting, forecasting & Reporting". Available at: http://aws.amazon.com/

[7] Cisco Global Cloud Networking Survey. Summary and analysis of results, 2012. Available at: http://www.cisco.com/en/US/solutions/ns1015/2012_Cisco_Global_Cloud_Networking_Survey_Results.pdf

Cloud Provider Interoperability and Customer Lock-in

Mirva Toivonen
Department of Computer Science
University of Helsinki
Helsinki, Finland
mirva.raman@cs.helsinki.fi

***Abstract*—Cloud providers (IaaS, PaaS and SaaS) have their own platform implementations and they use different implementation languages and modeling for implementing the same features. This incompatibility can lead to customer lock-in where customers cannot switch cloud provider without major extra re-adjusting like application rewrite. The switching cost may lock cloud consumer in so that they have to keep using cloud providers services if they want to avoid paying substantial switching costs. Lock-in is a business strategy that helps cloud providers to differentiate in tightening competition in cloud market. However lock-in deter organizations adopting cloud technology. Different standardization and abstraction layer solutions have been developed for increasing interoperability. Interoperability solutions like abstraction layers change market demand towards more open direction. Market demand and the ability to attract more customers are creating more pressure on cloud providers for supporting interoperability. In this paper I describe interoperability issues that leads to vendor lock-in and present brief overwiev on current standardization efforts and conversion technology. Finally a meta cloud example is presented.**

Keywords-component; interoperability; data migration; customer lock-in; cloud provider;

I. INTRODUCTION

Cost savings, power savings and increased agility in software deployment are some reasons why enterprises should go to the cloud. Using cloud infrastructures and platforms is convenient because services on demand offers high flexibility and pay as you go pricing offers low costs.

In this paper cloud consumers are users like developers or organizations, anyone who use cloud computing services. Cloud providers offers cloud computing services through Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service services (SaaS). IaaS is a computational infrastructure as a service that is based on virtual computers. PaaS is a software development platform service and SaaS are applications that are delivered to customers in the form of services over the internet [2].

Providers have their own incompatible platform implementations and they use different implementation languages and modeling for implementing the same features. Incompatibility can lead to a lock-in situation where customers want to change cloud platform, but can not do that because the lack of interoperability. This paper have short overview on interoperability solutions and reasons why providers promote their own formats.

The rest of the paper is structured as follows. In the second section I will briefly explain the key concepts and section three presents some thoughts of the lock-in as a business strategy. Fourth section present an overview of interoperability problems. Section five presents main features of interoperability standards. Abstraction layer solutions and an example of meta cloud is presented in section six. Section seven concludes the paper.

II. MOTIVATION

Interoperability means the ability to exchange information between two systems and the ability to operate on it. In cloud context information can be changed between two clouds or between organizations own private cloud and public cloud or a mixture of these. G. Lewis [12] points out that in cloud computing community the term interoperability is often used as in the meaning of portability, the ability to move system to another platform or the ability to bring a system back in to organization. In this paper interoperability refers to the ability to exchange information, operate on it and move data from one cloud platform to another.

Lock-in refers to a situation where customer is dependent on vendors products or services. In a lock-in situation switching vendor without paying substantial switching costs is not possible. Xiaoguo et al. [11] stated that substantial costs to switch between software systems can force a customer to continue to use products and services from a particular vendor. This means that customers have to keep using cloud providers services if they want to avoid re-design and adjusting to new cloud environment. Switching cost is the significant effort in adjusting to new cloud environment in the form of system re-design, re-deployment and data migration. As Open Cloud Manifesto [10] describes, bringing a system back in-house or porting system to another cloud provider will be difficult and expensive.

Lock-in and interoperability are related to each other because the lack of interoperability leads to lock-in problem. Lock-in by definition is caused by interoperability issues. Lock -in can happen if organization's system is designed to use some particular cloud provider platform. Each platform

supports different languages and different libraries that may not be incompatible. Vendor's can also license their software under exclusive conditions [11].Silva & Lucredio [2] take Google App Engine as an example. Google App Engine is a PaaS provider that have specific programming style and it's own way to manage data. This provides great elasticity for applications inside the engine but porting application into different cloud provider may not be easy. Unfortunately they do not explain in greater detail what kind of difficulties porting systems into different cloud brings. Besides this low level API access Google App Engine provides access to their data stores through standard APIs as well.

It is desirable for customers point of view that the ability to change provider is supported. Customers may in the worst case scenario lose their data or the provider might go out of business. Armsbrust et al. [1] took Linkup example from the year 2008 where the online storage service lost 45% of it's customer data. Linkup went out of the business after that incident.

Article [1] lists some less dramatic but still harmful risks that consumers may suffer from, which could be price increases, reliability problems and the lack of service quality like provider outages and failures. Substantial switching costs force customer to use some particular cloud service. Locked-in customers need to tolerate these risks. These risks may prevent some organizations from adopting cloud solutions and prevent wider industry adoption.

One interoperability benefit for customers - besides avoiding the lock-in risks - is is that customers are able to compare and choose between providers. Also the use of multiple clouds or hybrid clouds becomes possible when interoperability is supported.

III. Lock-in as a Competitive Advantage

Vendors support incompatible cloud platforms because it is a way to differentiate and it gives providers competitive advantage.

J. Bitzer [14] studied the role of product heterogeneity in software competition between commercial and open source software (OSS) with a simple economic Launhardt-Hotelling model. Their research question was why some proprietary software support the development of open source software while others refuse any support. The result was that the long-run survival of the existent software depends eventually on the heterogeneity between the existent software and the new product. The heterogeneity helps to cover application's fixed substantial development costs. Refusing to support OSS is rational for proprietary vendors, because it maintains the heterogeneity of their products relative to OSS. This kind of thinking may be behind cloud providers business logic. They do not want to support open standards because they want to maintain their differentiation benefits. Implementing own libraries, application softwares and platforms cloud providers gain more powerful features [12] and give differentiation advantage. Harmer et al. [8] wrote: "It is in the interest of providers to have their own APIs as this simplifies their development task, fitting perfectly their business model and their implementation". Differentiation enables providers to implement powerful features, innovate and enhance their services. Providers want to implement as powerful platform they can and give customers more value by doing so.

Yu [18] have different opinion on whether customer value is added by not supporting inteoperability: cloud converter technology adds more value- added services to consumers because fully realized interoperability adds pressure for cloud providers to offer more value- added services to attract consumers. "Each provider will continue giving their strong points into play based on their technical advantages, in order to stand out in competition"[18]. In other words value added services help to capture market share.

Interoperability solutions may lose some of theirs optimality. For example Ranabahu et al. [13] proposed semantic Domain Specific Language (DSL) development approach in supporting higher levels of cloud interoperability. The code generated via templates was functional, but perhaps was not optimal for given platform. G. Lewis [12] noted that the standard API makes applications more portable, but offers less control and less provider specific features than the low-level API.

Conversion technology is welcome because it attracts more consumers and more profits for cloud providers [18]. Cloud providers are beginning to realize that customers want more interoperable cloud services. By supporting interoperability cloud providers can potentially attract more customers, because customers want to move freely between cloud providers [18].

Governments are increasingly supporting interoperability and discouraging vendor lock-in strategies. This is an outer force that changes market demand towards more open and interoperable products. As an example of this Zhu et al. [11] gives a law (passed in 2006) that French Parliament passed that requires vendors of digital music player and online music services to open their technical standards and become completely interoperable. The aim for that law was to protect artists copyrights. The law was targeted towards Apple's digital music market, where music downloaded from iTunes was only playable on its iPod music players. Also technological devices or applications for increasing interoperability are changing the structure of a market [18]. Different conversion models give consumers tools for adjusting the tradeoff between overhead expense and vendor mobility.

Lock-in is not always a bad thing as long as everything goes well, platform works well and supports organization's long term business goals. The benefits of more powerful implementation could be more attractive than being able to switch provider. If the platform's implementation fits perfectly to the organization's business requirements the tradeoff between being locked-in and interoperability might be bigger for lock-in.

Whether to support interoperability depends on the key values of application or a system. For example if applications key value is in the data they collect or expose,

lock-in situation is a high risk [3]. That is because if each cloud platform supports different languages and different libraries that may not be incompatible, the portability will be limited.

IV. Interoperability Solutions

Loutas et al. [6] did a systematic review of semantic cloud computing interoperability papers. Interoperability solutions were divided into three category: standards, interoperability frameworks and abstraction layers. Standards try to unify differences between providers. The aim for standardization is common understanding in how clouds basic entities like resources, services and APIs are represented among cloud providers. Abstraction layers hide cloud providers underlying technical details and differences. Framework efforts emphasize different parts or directions of interoperability which vary from unified management, standardized APIs to layers of abstraction.

Interoperability problems can be divided into three leves:

- technical interoperability like incompatible virtualization implementation or programming code
- semantic interoperability where two systems understand and express the same information differently.
- organizational interoperability, business needs for cloud providers and customers are fullfilled.

Cloud community sets semantic interoperability between clouds as a high priority for cloud computing. Semantic models and techniques are utilized in solutions that try to solve interoperability problems [6].

Interesting finding [6] was that standardization, frameworks and semantic cloud solutions agree on using standardized APIs, common management models, and utilization of common marketplace, broker or abstraction layer. That means that standardization efforts, interoperability frameworks and cloud model solutions all employ similar strategies for achieving interoperability. Therefore two requirements for interoperable cloud should be met:

- a set of standardized cloud models should be developed. These models describe cloud elements like SLAs, computing resources and APIs. They give an abstracted view of these cloud entities and hides the differences among cloud providers.
- a standardized cloud API should be created and supported.

Standardized APIs and standardized cloud model solutions can be seen as fundamental requirements for achieving cloud interoperability because all three solutions categories give them high importance in cloud interoperability [6].

V. Standardization

First standardization efforts have been made for solving technical and service interoperability issues in the cloud. Bozman and Chen wrote [19] in 2010 that "being able to pick up a VM and move it is a primary first step to cloud portability". This first step was taken a year later when Open Virtualization Standard (OVF) was approved in 2011. OVF is a standard for system portability. Other efforts have been made in data portability and in cloud data management. Cloud Data Management Interface (CDMI) standard has been approved for data portability and cloud service standards like Open Cloud Computing Interface (OCCI) and Cloud Data Management Interface (CDMI) have been approved.

Standardization projects can be divided in two groups: standardizing parts of cloud solution and standardizing how parts work together [12]. This is consistent with Loutas et al. [6] paper which divide semantic interoperability issues in two categories:

- to compatibility conflicts, which can be solved with a set of standardized cloud models and
- to co-operation with another cloud system, which should be solved with standardized APIs.

Standards like IEEE P2301 and IEEE P2302 describe fundamental cloud entities. IEEE P2301 enables portability by grouping different options of the cloud elements into logical profiles. IEEE P2302 define topology, protocols, functionality and governance to support cloud to cloud interoperability and federated operations.

Distributed Management Task Force (DMTF) develops and maintains system management standards. DMTF and Open Cloud Computing Interface (OCCI-WG) are standards for interaction and management efforts.

- DMTF standardizes interactions between cloud environments, specifications for delivering architectural semantics and implementation details for interoperable cloud management (between providers and consumers)
- OCCI-WG allow development for interoperable tools for deployment autonomic scaling and monitoring. It have protocol and API for management tasks and remote management API for IaaS based services, suitable for also PaaS and SaaS services.

Table I and Table II from NIST Cloud Computing Roadmap [16] summarizes available standards and Standards Developing Organizations for service interoperability and data- and system portability. Tables include approved standards which are approved by standard development organizations and available for public use. Market acceptance means widespread use by many groups and de facto or de jure market acceptance of standards-based products or services. De facto and de jure acceptance means that the standard is in widespread use but not yet officially established.

I Interoperability Standards Mapping

Categorization	*Available Standards and SDO*	*Status*
Service Interoperability	Open Cloud Computing Interface (OCCI); Open Grid Forum	Approved Standard
	Cloud Data Management Interface (CDMI); Storage Networking Industry Association, SNIA	Approved Standard

	IEEE P2301, Draft Guide for Cloud Portability and Interoperability Profiles (CPIP), IEEE	Under Development
	IEEE P2302, Draft Standard for Intercloud Interoperability and Federation (SIIF), IEEE	Under Development

II PORTABILITY STANDARDS MAPPING

Categorization	*Available Standards and SDO*	*Status*
Data Portability	Cloud Data Management Interface (CDMI); SNIA	Approved Standard
System Portability	Open Virtualization Format (OVF); DMTF	Approved Standard Market Acceptance
	IEEE P2301, Draft Guide for Cloud Portability and Interoperability Profiles (CPIP), IEEE	Under Development

VI. ABSTRACTION LAYERS

Standardization appears to be good solution to address the interoperability issue [7], however several articles [7,8,12,13] stated that standardization in PaaS and SaaS services is not likely to happen in next few years. Instead of standardizing all cloud providers, one solution is to allow multiple cloud usage with abstraction layers between cloud provider system and customer's system. Abstraction layers [3,4,5,8,9,15] hide differences between cloud providers. Abstraction layers help to ignore the provider being used, their utility model and their API which enable the use of hybrid clouds. The advantage is that the abstraction layers bring immediate results compared to the slow standard development. Abstraction layers are user friendly and cloud provider friendly because it allows providers to freely define their own cloud policies and users are free to choose a cloud provider [5].

Hybrid cloud means that customer deploy services from multiple clouds. One hybrid cloud application is cloud bursting where an application runs in a private cloud or data center and uses public cloud when there is a high demand spike for computing. Cloud captures the extra tasks that cannot be easily run in the internal system. In cloud bursting the customer pays only for the extra compute resources when they are needed.

Some abstraction layer solutions are Meta Cloud[4], abstraction layer [8], RACS proxy for cloud storages [9], Mosaic of Clouds mOSAIC [15] and cloud broker for any cloud client [3]. Cloud application solutions for interoperability are Model Driven Engineering (MDE) based application [5] and semantic solution based on DSL application development [13].

Yu [18] represents a high level model for conversion technology which consists of three components: interface, abstraction layer and management tools. Interface migrates applications and data from customer's interface to a converter that converts the standard to the target cloud's standard. Abstraction layer isolates virtualized services from the infrastructure in abstraction layer, reducing dependence on ecosystem in the cloud. Management tools operates on application and data in different clouds through unified and standardized management interface. Figure 1. from [18] represents a high level model for conversion technology.

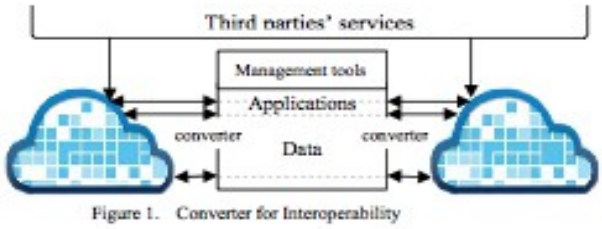


Figure 1.

Meta Cloud abstraction layer [4] is an example of solving interoperability problems with conversion technology. The aim for Meta Cloud is to find an optimal combination of cloud services, to allow applications to run anywhere and support runtime migration. Meta Cloud utilize existing tools and concepts like libcloud, fog and jclouds libraries, resource templates like Amazon Cloud Formation and TOSCA and automated deployment tools like Opscode Chef.

Developers create cloud applications using meta cloud development components:

- *meta cloud API*, which consists of abstraction libraries like libcloud, fog and jclouds. It incorporates design time and runtime components.
- *resource templates*, which define concrete features that the application requires. Developers use resource templates for describing cloud services they need and constraints for costs or geographical distribution. Cloud services could be CPU, memory and dependencies and communication relations between components. Resource templates are created by developers by using DSL.
- *migration and deployment recipes* which describe automation process. Migration handles migration of an application at runtime and deployment recipe handles package installation, starting required servies, managing package and application parameters and establishing links between related components.

Runtime migration and automated application life-cycle management is supported with knowledge base, resource monitoring, and provisioning strategy:

- *Knowledge base* stores information of resource templates and migration or deployment recipes.

- *Resource monitoring* gets QoS statistics from the cloud proxy and enables meta cloud to decide whether to provision new instances of the application or migrate parts of it.

- *Provisioning strategy* matches application's cloud service requirements to actual cloud service providers. Provisioning strategy makes a list of possible cloud service combinations that are ranked based on QoS information of expected QoS and costs in knowledge base. Provisioning strategy uses resource templates for determine the applications optimal deployment configuration.

The mediator part is Meta Cloud proxy which is deployed with the application and run on the provisioned cloud resources. Proxies expose the meta cloud API to the application, transform application requests into cloud-provider-specific requests, and forward them to the respective cloud services. Proxies also send QoS to resource monitoring which send the information forward to knowledge base.

Figure 2. from [4] demonstrate the conceptual meta cloud overview.

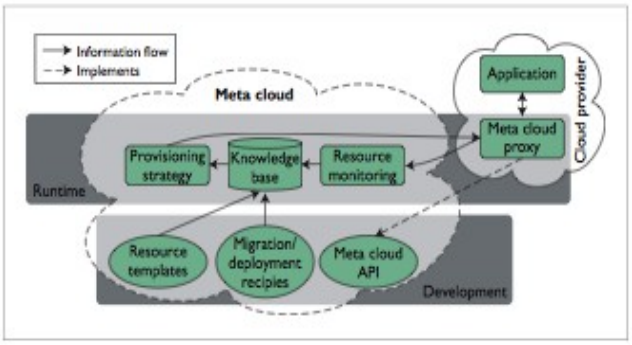


Figure 2.

VII. Conclusion

Cloud providers use their own implementations and notations for differentiation purposes. While new cloud vendors come to cloud markets, differentiation help providers to survive in the tightening competition between cloud players. It seems that proprietary cloud providers had little desire for opening their systems because differentiation is considered as a competitive advantage. However different interoperability solutions like abstraction layers and standardization efforts are slowly changing the market demand towards more open and interoperable direction. This means that cloud providers have to respond to this demand and support interoperability if they want to attract customers.

There are theoretical standardization efforts and more practical abstraction layer efforts for solving interoperability. The difference between these two is that standardization efforts try to change differences between providers towards more unifying and homogenized way while abstraction layers want to hide underlying technical details and differences and allow the use of multiple cloud. The aim for standardization is common understanding in how clouds basic entities like resources, services and APIs are represented among cloud providers. Both theoretical works, like standardization efforts and practical works, like abstraction layer solutions, agree on steps to be taken for achieving semantic interoperability. For exchanging information and being able to understand it semantic interoperability solutions are common for describing cloud artifacts.

For achieving interoperability a set of standardized cloud models for describing cloud entities like SLAs or computing resources should be developed. Standardized cloud API should be created and supported for managing cloud resources and usage. Standardization is in a good beginning for standardizing technical incompatibility issues. This is a good start for cloud services who are moving towards dynamic systems where location, negotiation, provisioning and instantiation of cloud resources occur at runtime.

References

1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "A View of Cloud Computing." Commun. ACM 53, 4 (April 2010), 50-58, 2010.

2] Da Silva, E.A.N., Lucredio, D., "Software Engineering for the Cloud: A Research Roadmap" Software Engineering (SBES), 2012 26th Brazilian Symposium on, pp.71-80, 23-28 Sept. 2012.

3] E. Maximilien, A. Ranabahu, R. Engehausen, L. Anderson, "Toward cloud-agnostic middlewares." Paper presented at the Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, pages 619-625, 2009.

4] B. Satzger, W. Hummer, C. Inzinger, P. Leitner, S. Dustdar, "Winds of Change: From Vendor Lock-In to the Meta Cloud," IEEE Internet Computing, vol. 17, no. 1, pp. 69-73, Jan.-Feb., 2013.

5] J. Miranda, J. Guillén, J. Murillo, and C. Canal, "Enough about standardization, let's build cloud applications." In Proceedings of the WICSA/ECSA, Companion Volume (WICSA/ECSA '12). ACM, New York, NY, USA, pp. 74-77, 2012.

6] Loutas, N.; Kamateri, E.; Bosi, F.; Tarabanis, K., "Cloud Computing Interoperability: The State of Play," Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on , vol., no., pp.752,757, Nov. 29 2011-Dec. 1 2011.

7] T. Dillon, C. Wu, E. Chang, "Cloud Computing: Issues and Challenges", Advanced Information Networking and Application (AINA), 2010, 24th IEEE International Conference, pp. 752-757, Nov. 29 2011- Dec.1 2011.

8] T. Harmer, P. Wright, C. Cunningham, and R. Perrott, "Provider-Independent Use of the Cloud," in The 15th International European Conference on Parallel and Distributed Computing, 2009, p. 465.

9] H. Abu-Libdeh, L. Princehouse, H.Weatherspoon, "RACS: a case for cloud storage diversity". In Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10). ACM, New York, NY, USA, 229-240, 2010.

10] The Open cloud manifesto: http://www.opencloudmanifesto.org/Open%20Cloud%20Manifesto.pdf Spring 2009.

11] K. X. Zhu, Z. Z. Zhou, "Lock-In Strategy in Software Competition: Open-Source Software vs. Proprietary Software", Information Systems Researchs, Volume 23, Issue 2, Pages 536-545, 2012.

12] Grace A. Lewis, "Role of Standards in Cloud-Computing Interoperability" IEEE, 46th Hawaii International Conference on System Scieneces, pp. 1652-1661, 2012.

13] Ajith Ranabahu, Amit Sheth, “Semantics Centric Solutions for Application and Data Portability in Cloud Computing”, 2nd IEEE International Conference on Cloud Computing Technology and Science, pp. 234-241, 2010.

14] J. Bitzer, “Commercial versus open source software: The role of product heterogeneity in competition”, Economic systems, Volume 28, Issue 4, pages 369-381, December 2004.

15] B. Di Martino, D. Petcu, R. Cossu, P. Goncalves, T. Mahr, M. Loichate. “Building a Mosaic of Clouds” In Euro-Par 2010 Parallel Proces Workshops, volume 6586 of Lecture Notes in Computer Science, pages 571–578. Springer Berlin /Heidelberg, 2011.

16] M. Hogan, F.Liu, A. Sokol, J. Tong, “NIST Cloud Computing Standards Roadmap – Version 1.0”, (Special publication 500-291), NIST: National Institute of Standards and Technology, Gaithersburg, 2011.

17] P. Harsh, F. Dudouet, R. G. Cascella, Y. Jégou, C. Morin,“Using Open Standards for Interoperability: Issues, Solutiond and Challenges facing Cloud Computing”, Proceedings of the 2012 8th International Conference on Network and Service Management, p. 435 2012.

18] Z. Yu, “Cloud Computing – Conversion Technology for Interoperability”, Fourth International Conference on Multimedia Information Networking and Security, pp.179-182, 2012.

19] J. Bozman, G. Chen “Cloud Computing: The Need for Portability and Interoperability”, IDC, sponsored by Re Hat. Inc. August 2010.

Comparision of different approaches to evaluate cloud computing services

Rakesh Pandit
Department of Computer Science
University of Helsinki
Helsinki, Finland
rakesh.pandit@cs.helsinki.fi

***Abstract*—There has been a large increase in the number of enterprises offering various cloud services since last 5 years. Good deal of research has been happening to evaluate these services based on different criteria. Even though, the progress in this area is substantial, the related research has been done in bursts and there is still no consensus on a fixed set of metrics or frameworks that represent various evaluation aspects. This paper is a systematic review to compare three efforts by Li, Brien et al., Li, Yang et al. and Garg et al. The approaches taken by all three, to consolidate different matrics into a single framework or catalogue, in order to help users and providers to evaluate/rank services, is different and unique. This paper will discuss these approaches and different goals these efforts try to address. This discussion also includes the use of Goal Question Matric paradigm to simplify the comparision.**

***Keywords*-Cloud computing; Quality of Service; Measurement;**

I. Introduction

Cloud services, be it Infrasture as a Service (IaaS) or Software as a Service (SaaS) or Platform as a Service (PaaS), are on rise. Cloud service providors are attracting customers to move computation and storage requirements to cloud. This provides several advantages, including low initial cost of infrastructure and maintainence, reliability and cost benefits on computation as well as storage. As cloud services reach more application and customers, it has become important to evaluate them based on different attributes and requirements. This is not only true (obvious reasons) for customers, but also important for service providers to improve services. For customer this evaluation is important, and allows them to do a cost benefit analysis for initial migration and later service utilization. It also allows customer to select between range of service providers offering similar services, based on customer requirements and future growth. Some service providers may provide good services for I/O intensive applications and others for CPU intensive applications. So, even for single customer it may make sense to use two different services. Even if customer is locked up with a service provider, there is a need to periodically verify and compare whether the quality of service is same or not, or whether it can get a better deal from other service providers. There are two major sources of metrics available, which can used to evaluate different cloud service offerings. One of the source is Service Level Agreements (SLAs) signed between service provider and customer, other source is academic research publications.

There has been lot of research in cloud service metrics to evaluate different aspects of these services. There have been some efforts to create catalogues or frameworks to assemble all of these matrics to create a single place of reference or converge the effort. But all of these efforts are still happening in bursts and they haven't converged yet. This paper takes up three recent research efforts towards creating frameworks and catalogues to converge different metrics. These efforts are useful for customers as well as service providers. It helps customers to evaluate different services based on their requirements, by referring to a single catalogue or framework. In addition it also helps service providers to evaluate their services by comparing them to similar services offered by different providers. Moreover, it converges all the efforts going into evaluating different metrics, for different attributes, and for different requirements. Three efforts we will be discussing in this paper have been done by Li, Brien et al.[1], Li, Yang et al.[3] and Garg et al.[2]. Li, Brien et al.[1] has targetted commercial service providers directly and developed a catalogue to evaluate services based on comparision to different services already available in market. Garg et al.[2] on the other hand have developed framework keeping user/customer presepective in mind. Li, Yang et al.[3] have developed a framework which takes into consideration both customer requirements as well as provider's service quality.

In this paper we will discuss different approaches taken by all three researches to come at catalogues and frameworks, goals they try to address, intersections between them, existing coverage i.e cloud service providers these catalogues have been run against. There are number of challenges these efforts need to overcome while doing research on framework or catalogue. Metrics need to cover attributes which are not only obvious and relevant to different customers, but also simple to compute, so as to keep measurement cost low[10]. Second important challenge is to select set of attributes which are common and most relevant to the goals framework wants to address. There are number of attributes frameworks may have to give up to save on development time and reduce complexity of framework.

The remaining paper is divided into 4 sections. Section II

discusses different approaches taken by three efforts. Section III discusses the metrics and frameworks in those efforts. Section IV does a comparision using GQM paradigm. Section V has conclusion.

II. Different approaches

In this section we discuss about the approaches these three researches have taken, to come up with a catalogue or framework. Research done by Li, Yang et al.[3] targets both customers/users and cloud service providers. They target at providing information related to how costly different services will be to different customers. In addition their metrics targets to provide performance comparisions specifically targetted at applications running on cloud. This includes computation, networking (both intra and inter) and storage. For service provider, their framework doesn't not just aim to provide performance results, but also reasons because of which it performs bad. Their research focus is to keep performance and cost mertrics uniform for different services providers. This limits it to fixed set of services which are common in industry i.e. they don't aim at covering services which specially target small set of applications. This has couple of advantages. It keeps metrics fair and increases coverage. In addition, it keeps the cost of research and measurement low. The idea here is to measure different service providers periodically, making sure that service provider's user policies are followed even while taking the measurements. This makes it possible that measurements are realistic, and metrics do not depend upon factors which violating usage policies, and thereby preventing it from being widely usable. Covering every service provider would still be costly and time consuming. Making a good judgement on that is also important, but the research doesn't include any suggestions on that account. Despite that they have tried to keep important aspects related to deployment (their solution - ClouCmp[3]) in mind, while makes it a good candidate for different customers and service providers.

Second effort by Li, Brien et al.[1] tries to come up with a catalogue by targetting only commercially available cloud services. Their research covers evaluation of existing metrics for common commercially available services. This resulted in dividing all metrics based on three aspects: performance, security and economics. Based on this division, they have planned their catalogue around these three aspects. Performance involves communication between users and services or between services internal to cloud provider for a specific client. In addition performance also includes computation, which is important for CPU intensive applications. Performance metrics also covers mermory both non-persistant as well as permanent storage. Certain application just need temporary (cache) memory for fast access but others need storage for permanent data. In addition to memory, computation and communication, they propose certain metrics to give an overall picture of service performance. For economics, two areas which are covered for metrics are: elasticity and cost evaluation. Elasticity refers to extension or reduction of resources allocated for a customer. Other factor related to economics is cost, which doesn't only play part in direct expenses. Li, Brien et al.[1] have taken other aspects of cost into account e.g. incremental cost effectiveness, time related and performance related cost effectiveness, etc.

Effort by Garg et al.[2] has entirely based on user view and user's requirements. It takes into consideration different attributes such as accountability, agility, cost, performance, assurance, security, privacy and usability into account. Many of these attributes are non-functional i.e it is difficult to assign values to them e.g security. Garg et al. develops a framework which takes user requirements into account and based on those filters service providers. It then ranks these service providers based on best fit, which is in turn based on attribute based metric set. Ranking is based on weights associated with different attributes depending upon user requirements.

III. Comparison of frameworks and catalogues

In this section we discuss in details metrics taken up in all three cases and compare them. We divide them into four categories and look at coverage of all three studies in discussion.

A. Performance

A service provider provides computation power to user via virtual instances. These virtual instances are useful for executing customer application or storing customer information either for shorter or longer period. Most of cloud service providers have more then one level of performance based services available. Different levels have different resources allocated i.e. some levels of service will have faster CPUs, more cores, and good storage capacity. The transition between these level can be manual or on demand. In later case customers are moved automatically to higher levels depending upon running application requirements.

Li, Yang et al.[3] in Cloudcmp use three metrics to compare computation power of different service providers. These are benchmark finishing time, cost and scaling latency. Note that cost is an economic metric, but here it is referance is to cost of running benchmarks. Benchmark finishing time uses commonly used benchmarks for CPU usage and other metrics. There are various challenges for running these benchmarks e.g different services providers provide a closed environment to run application code. These closed environments have limitations to number of cores, memory usage and time for completion. So, common benchmarks used for desktop computors need to be updated to meet these constraints. Because running a benchmark involves paying cloud service provider, this makes it an interesting parameter to do a quick comparision among different cloud service provider using cost-performance ratio. Li, Yang et al.

made sure to keep these modified benchmarks simple and these benchmarks, based on service provider, can also use multicores if available. Similicity has an advantage that it gives realistic comparision for cost-performance evalation for small budget and within different levels of services offered by same provider. Cloudcmp also has metrics for scaling latency. This metric is useful to measure scaling of resouces based on application usage. This is a good metrics for users to figure out how different service providers behave with different scaling strategies and whether it would be better for application to use another strategy to minimize the cost and not pay in terms of performance. Cloudcmp makes it possible to differentiate between provisioning latency and booting latency. Provisioning latency is the time delay taken by service provider to allocate resources (e.g. virtual machines) to users were as booting delay is the time it takes for assigned resouce to be ready for usage. These are important differences and can play vital role in decisions taken by customers. These are also useful for cloud providers to make sure customers know that allocation of a resouce taking booting time. It also allows service providers to compare resource allocation methodology with other service providers.

Li, Brien et al.[1] have divided performance into four categorises: communication, computation, memory (cache) and storage (persistant). Every category is further divided on the basis of capacity. These divisions for each of the category are based on: speed, availability, latency, reliability and throughput. This division is based on cloud services being offered by some common service providers. In addition to this division scalability and variability is also taken as part of performance metrics. Communication involves transfer of messages between intra cloud services or between exyernal clients (from customer/user end) to exported services for consumption. These messages can be network level i.e TCP/UDP messages or application level MPI messages. Both of them have been taken into consideration. These metrics also allow customers to get a hint about how much of total time has been spent in intra and external communication, and hence whether customers can adapt their application for better gain and savings. Computation evaluation is done to check whether CPU intensive applications can perform better in cloud instances (virtual guest). Transaction speed and latency are important capacity attributes taken into picture here. For temporary memory (cache) performance this catalogue involves certain metrics which can be useful for customers to check response time with different hierarchies of memory provided by different service providers. Persistant storage is another performance category coverd by this catalogue. Accessing persistant data on cloud services takes longer time then temporary data. For accessing small files or small data, the overhead for different services can vary and thereby effect applications. Methods for accessing data from cloud service providers can be divided into three parts: downloading blobs i.e. big chunk or queries related to tables e.g GET, PUT etc. or queuing operation e.g add, remove or insert data. Storage evaluation metrics provides metrics to differentiate these requests and provide relevant readings for them. Interestingly this catalogue also includes plotting of histogram for HTTP/GET requests which is useful to check throughput during a particular time slot. In addition to communication, computation, and memory (both temporary and persistant) this catalogue also provides certain performance metrics useful for providing overall performance of cloud service provider. One of these matrics provides aggregiate performance by running a fixed set of applications with known functionality and resouce consumption on different service providers. Another being performance based on requests which consume different resources and hence give a weighted indicator. Weight is derived from resource consumed by request in question. This set of overall performance indicators are useful for getting a big picture on obvious attributes for common requirements and expectations of a customer. Scalability and Variability are useful indicators but rather then numbers tables and charts give a quick indication of service performance for both of these capacity attributes. This catalogue considers charts and tables as metrics as well and for both of this capacity attributes plots graphs and charts based on scaling and variation in resources consumed by application or provided by service provider.

Garg et al.[2] has created SMICloud a framework which ranks different service provides based on different users and customers but still gives a general view useful for service providers as well. Performance includes service response time, sustainability, data center performance per energy units, suitability (which can be considered as performance metric), accuracy, availablity, stability, adaptability, scalability, throughput and efficiency. Some of these attributes are non-fucntional, but Garg et al. has tried to find closed possible quantifiable term. Service response time seems to be an obvious choice. It includes time when client/customer issued a request, time after which resouce is ready for usage, including boot time and even IP allocation time if applicable. It is measured till the time requestor can start installing application. SMICloud includes sustainability based on carbon footprint. It may not be a useful metric from customer or service providers point of view, but is an important metrics not covered by other efforts. SMICloud proposes to use different available calculators or famous metrics for calculating the carbon footprint. Data center performance per energy metrics takes into account IT infrasture energy usage and equipment utilization. It also includes calculation of renewable energy usage by service provider. It will help service providers to look into energy saving options and compare there consumption details with other service providers if available. There are two types of user requirements related to sustainibility: essential and non-

essential. Sustainability metrics takes into account both of these while calculating metrics value. Garg et al. calculates accuracy based on how much service provider's services are deviating from expected values as agreed in SLA's signed with user/customers. This is applicable for computation resources because they are usually quantifiable and agreed on in SLAs. SMICloud proposes to calculate transparency as effect of service changes on applications running in service. The effects can be replaced with frequecy of changes as well. Availabilty is defined as time for which user/customer can access a given service and reliability on the other hand is defined as mean time between failures based on assurances by service provider. Stability is deviation in resouce value from normal and mean of rate of change over a fixed period of time. Adaptability is the time service provider takes to adjust its resouces or offerings based on client/customer requests. It is calculated in terms of time taken to reach at new state which meets new demands. Scalability is the ability of a service provider to handle many requests at same time as demand grows.

B. Meomory

Cloud services provide memory has part of their offerings. It could be in the form of temporary memory e.g cache, used by applications to do normal processing or it could be persistant storage to store data for fixed time i.e longer durations. There are three types of storage offerings offered by service providers these days[3]: storage for blobs, table storage for structured data and queues for messages. Blobs store unstructured data e.g long files which are downloaded and uploaded to service providers. Table storage involves records in simple tables with support to simple queries and not complex queries e.g join or group by queries. Third type of storage is message queues which stores messages and passes it to different instances[3]. Interfaces for these storage services are stable mostly and these can be accessed over HTTP tunnels. Storage services provide high availability and reliability of data but the consistency levels are not strong enough[3]. Consistency refers to the fact that a read followed by a write will result in latest results.

Li, Yang et al.[3] uses three types of metrics for comparing cost and performance related to memories. These are response time, cost per operation and time to consistency. For an operation, from the time it is issued and till the time response is received back we compute the time interval and use it as operation response time. The operations are common operations for different types of storage solutions offered by different service providers. For example, for table operations are get, put and query, for queue operations are send and receive and for blob operations are upload and download. As mentioned earlier, usually service providers implement HTTP tunnels to offer storage APIs. Li, Yang et al.[3] have developed programs which use persistant HTTP connections to these APIs and call these APIs. Persistant connections help reduce latency by not establishing connection everytime query has to be sent. There is more to it then seems at first. Li, Yang et al. vary request size to measure latency vs throughput. They are also run simultaneous benchmarks at same time to check limitations of throughput. Cost per operations is an important metric which allows customers to compute cost-effectiveness for different services specifically for storage and choose best amoing them. It uses billing API to compute cost for services. Different services provides charge based on different metrics e.g some charge on amount of data passed in communication were as others charge based on CPU cycles spent for query processing. Time to consistency is another important metric which measures the time between a data element is written to the time it is available for consistant reads. Usually services don't offer consistency between different data centers, and this study limits itself to single data provider. This metric is useful for customers who want strong consistency for some data critical for operation of application. The test run is simple, first write something to storage via API and then repeatedly read at regular intervals to come up time it takes for results to match.

Garg et al.[2] has certain hybrid metrics which measure storage related metrics, but nothing dedicated to storage/memory as such. As an example throughput and efficiency metrics, accuracy, availability and data center performance per energy are related to memory as well. Accuracy measures degree to which service provider obeys with promises in SLA. These involve memory related promises as well, including both temporary and persistant memory. DPPE (Data Center Performance per Energy) includes storage capacity in its energy computation. Cost metrics also involves cost of storage in addition to computation and network bandwidth.

Li, Brien et al.[1] divide memory evaluation into two groups. First group evaluates temporary memory and second evaluation metrics takes care of persistant storage. Temporary memory (e.g cache) can be used to access momory for day to day operations of an applications. This memory isn't permanent but needs to be fast. Random Memory Update rate, Mean Hit Time, Memcache Get/Put/Response Time, Intra node scaling, Sharp performane drop are some of the metrics used for temporary memory performance. As an example Sharp performance drop is useful to find cache boundries[1]. Intra node metrics on the other hand is used to find cache contention by using scalability evaluation metrics. It uses different load under different CPU cores to check execution. Permanent storage performance evaluation metrics are more common though. Li, Brien et al. use one byte data access rate, benchmark I/O opertion speed, Blob/Table/Queue IO (similar to Li, Yang et al. above) operation speed, Blob/Table/Queue IO, Page Generation Time(s), I/O Access Retried Rate, Benchmark I/O bit/Byte Speed etc for storage evaluation[1]. One byte data access rate

is useful for calculating transaction speed. This is important to know the overhead assosiated with transfer of files.

C. Network

Network performance measurement is important part of cloud service validation. Our interest is specially intra-network communication between different instances of customer applications running in cloud or among applications to other shared cloud services[3]. The communication between intra datacenter instance of application is different then between two separate data centers. Communication between two separate data centers isn't important, but communication with data center and shared services is relevant because usual rates/charges for outside communication are same[3]. Intra datacenter infrastructure differs between different service provider i.e. switches, router hardware, cables on intra cloud network vary. Hence, evaluation of communication is important. Communication also plays a key role in cloud instances and clients/customer softwares. Li, Yang et al.[3] path capacity and latency to compare intra cloud communication. Path capacity is measured by measuring throughput for TCP traffic as TCP protocol is mostly used for communication. It allows one to figure out data throughput and congestion rate in the intra network. It also allows customers to figure out the configuration of internal network. For external communication to client/customer applications, cloud services divert traffic to instances which are very near to the point from which initial request was sent. Li, Yang et al. use optimal wide area network latency to compare this kind of communication[3]. This latency is useful to calculate the actual transmission latency between customer and cloud service provider when there is optimal mapping i.e. the customer has been directed to nearest instance of cloud provider. Li, Brien et al.[1] has metrics for both IP level as well MPI messages for application level. Max number of transfer sessions, packet loss frequency, correlation between runtime and communication time, TCP/UDP/IP transfer delay, MPI Transfer Delay, connection Error Rate, Proble Loss Rate, TCP/UDP/IP transfer speed, MPI transfer speed are metrics included into communication evaluation metrics[1]. As with other metrics in catalogue composed by Li, Brien et al. communication evaluation metrics is also divided into five capacity attributes and they are: transaction speed, availability, latency, reliability and data throughput. Garg et al.[2] metrics are hybrids which include network related parts in metrics such as data center performance per energy, stability, cost, accuracy, availability etc.

D. Others

Garg et al.[2] defines certain metrics which don't fall under computation, network and memory. Usability, interoperability, adaptability are few of them. Usability is defined as average time a user learns to use cloud services, install, remove and update application software on it. It includes operating on cloud service, installation of dependencies and software, learning features and other related documentation[2] etc. Interoperability is defined as possibility of communication between different instances hosted on different cloud providers. It is useful for migration or communcation between different applications. Adaptability is defined as service providers ability to provide new resouces based on request. Garg et al.[2] has used four assesment criteria to come up with useful and practical metric measurements. These are correlation, practical computability, consistent and discriminative power.

Li, Brien et al.[1], also mentions security evaluation metrics like communication latency over SSL, Discussion on SHA-HMAC, Discussin using Rish List, General discussion etc. There are no formulae for most of them but reseach assumes discussion as metrics as well. It also discussion elasticity and cost in economic evaluation metrics. Elasticity includes boot time, total acquisition time, suspend time, delete time, total release time etc.

Metric	SIMCloud	CloudCmp	Catalogue Li, Brien et al.
CPU	[ok]	[ok]	[ok]
Communication		[ok]	[ok]
Memory		[ok]	[ok]
Scaling Latency	[ok]	[ok]	[ok]
Benchmark cost		[ok]	
Sustainability	[ok]		[ok]
Suitability	[ok]		
Availability	[ok]		[ok]
Stability	[ok]		[ok]

Figure 1. Performance metric comparision based on GQM paradigm

Metric	SIMCloud	CloudCmp	Catalogue Li, Brien et al
Network Reliability	[OK]*	[OK]	[OK]
Latency		[OK]	[OK]
Network Throughput	[OK]*	[OK]	[OK]
Availability	[OK]*		[OK]

Figure 2. Network metric comparision based on GQM paradigm. [OK] marked * are special cases where metrics are more abstract and don't have a simple difinition.

IV. COMPARISION USING GQM PARADIGM

Goal Oriented Measurement (GQM) paradigm[10] is a process improvement method in which measurements are laid down to address completeness, consistency and suitablitly of a process. Figure 1 compares three research efforts based on goals related to performance of cloud services. These goals are CPU, communication, memory

Metric	CloudCmp	Catalogue Li, Brien et al
Cost		[OK]
Consistency time	[OK]	
Query/Data pull request		[OK]
Benchmark cost	[OK]	[OK]

Figure 3. Memory metric comparision based on GQM paradigm

(for processing), scaling latency, benchmark cost etc. Based on comparision it can be easily seen that catalogue by Li, Brien et al. seems to meet most of these goals were as SIMCloud a near second. This trend continues if we compare three research efforts based on network metrics figure 2 e.g reliability, latency, throughput and availability. Catalogue by Li, Brien et al. outshines other two even in memory metrics comparision figure 3. As stated in earlier section , all three efforts have taken different directions and it can easily seen if one compares them based on GQM paradigm. Definitions of some of the available metrics is very abstract and this comparision isn't really perfect, but it gives a broad pictures.

V. Conclusion

The number of cloud service providers has increased and the corresponding type of services offered have increased manifold since last few years. It is important for customers to use a good source of metrics but consistently same source of metrics to evaluate these services. The paper tries to compare three different approaches taken to assemble metrics and hence aims to converge the research work. Future work will involve evaluating further these metrics frameworks and catalogues based on GQM paradigm[10] and come up with a single unique metrics which uses a broad approach to prevent research in bursts.

References

[1] Z. Li, L. O'Brien, H. Zhang, and R. Cai, "On a catalogue of metrics for evaluating commercial cloud services," in *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on*, sept. 2012, pp. 164 –173.

[2] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012 – 1023, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.future.2012.06.006

[3] A. Li, X. Yang, S. Kandula, and M. Zhang, "Cloudcmp: Comparing public cloud providers," *Internet Computing, IEEE*, vol. 15, no. 2, pp. 50 –53, march-april 2011.

[4] R. Krebs, C. Momm, and S. Kounev, "Metrics and techniques for quantifying performance isolation in cloud environments," in *Proceedings of the 8th international ACM SIGSOFT conference on Quality of Software Architectures*, ser. QoSA '12. New York, NY, USA: ACM, 2012, pp. 91–100. [Online]. Available: http://doi.acm.org/10.1145/2304696.2304713

[5] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 931–945, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2011.66

[6] L. C. Briand, C. M. Differding, and H. D. Rombach, "Practical guidelines for measurement-based process improvement," *Software Process: Improvement and Practice*, vol. 2, no. 4, pp. 253–280, 1996.

Analysis of the Availability of Amazon Web Services' Cloud Infrastructure Services

Santeri Paavolainen
Department of Computer Science
University of Helsinki,
Helsinki, Finland
Email: santtu@iki.fi

***Abstract*—With the rising importance of cloud infrastructure services as basic building blocks used for creating large-scale distributed software systems comes also the need to understand why they fail, how they fail and how often do they fail. A lack of accurate information on availability of cloud infrastructure services means that any predictions on availability made during system design are based either on *ad hoc* estimates or based on service level guarantees provided by cloud vendors themselves.**

This paper provides an analysis of known failures in one cloud infrastructure provider, Amazon Web Services, and from this availability estimates for overall availability, availability of EC2, EBS, RDS and ELB and S3 services globally as well as the availability of aggregate of all analyzed services by geographical area.

The results are based on 12 months of publicly available failure information for Amazon Web Services. The resulting availability rate for the analyzed infrastructure service is between 99.947% and 99.981% depending on the assumptions made with failures with incomplete data. Also a wide variance in failures between geographical regions was observed with the largest total duration of failures in us-east-1 region and the lowest in us-west-2 region.

I. Introduction

In recent years the use of cloud computing has increased dramatically [1]. Cloud computing offers many inviting features such as service scalability and availability that were difficult to reach without substantial capital investments in hardware. Now cloud-based virtualized computing, networking and storage services are widely available to customers globally on demand and paid on a pay-by-use payment model [2].

Cloud computing has also given rise to new concerns over security, availability and data durability among others [3, Table 2]. Not all concerns necessarily turn out to be real problems – but even unfounded concerns affect expectations of cloud services and can hinder their cost-effective and timely adoption.

Concerns about cloud service availability have a good soil to grow in. During the initial period of cloud computing there were several highly publicized outages by different vendors (for an early list see [4]). Although cloud service outages are occurring more or less regularly, their relevancy to customers need to be considered carefully. Early cloud service failures predictably affected a large portion of customers as the absolute number of cloud service users was small.

It is natural to expect that cloud services will keep failing also in the future. Other systems such as electricity grids we normally consider as highly reliable have outages. There is no reason to assume that cloud services - both software and hardware - would be any different. Thus the question is not whether cloud services would fail – they will – but what is the actual impact to customers that failures will have?

There has been a lot of research on cloud computing quality metrics such as computing expected resource quality and stability of cloud computing systems [5]–[7] as well as applying reliability engineering techniques and their effects when used on real or simulated cloud computing environments. However to the best knowledge of the author there is no research into actual availability of cloud computing infrastructure services based on historical outage events. The lack of fact-based availability data noted also by Iosup et al. [8].

Reliability engineering and fault tolerant system architectures [9], [10] are used to guard against localized infrastructure failures. Fault-tolerant system design is considered a must feature for any non-trivial service. In practice reliability engineering techniques on systems built on cloud infrastructure services does face a practical problem when trying to estimate the amount of resources needed to attain some desired availability goal (conversely the same problem applies when trying to estimate availability of a system based on available resources). Without accurate knowledge of availability of the underlying cloud infrastructure one must either build an excess of redundancy into the system (at a cost), or face a possibility of over-estimating availability of the finished system (at a potential increased cost).

The goal of this paper is to estimate availability of Amazon Web Service's core infrastructure services (EC2, EBS, RDS, ELB and S3, for description of these services see [11]) based on publicly available historical information. Amazon Web Services was chosen as the evaluated cloud infrastructure provider because its dominant market position which makes its failures highly visible and publicized. In particular, this paper focuses on estimating service availability on the granularity of *availability zones*.

The rest of this paper is arranged so that Section II describes sources of failures in cloud infrastructure and how these manifest themselves visibly to customers. Section III describes sources of data used in the study and how data is processed and analyzed. Section IV discusses results of the analysis and Section V describes potential sources of errors in the data and

its analysis. Finally related work is presented (Section VI) before concluding (Section VII).

II. Failures in the Cloud

A. Failure Sources

There has been a large amount of research into failures of conventional computer systems and server virtualization [12] [7]. There is no reason to assume these results would not apply equally to the hardware used to build cloud computing systems. A similar argument can be made when looking at failures and availability of whole data centers [13].

Cloud infrastructure also creates new failure models that were non-existent in earlier computing systems. Undheim et al. [14] and Jhawar et al. [15] point out that the *control plane* responsible for provisioning and management of cloud infrastructure resources becomes a new potential source of errors. While cloud-based systems with slowly changing resource needs may choose to ignore the possibility of control plane problems this is not the case for all systems. For example systems with a variable usage pattern can be severely degraded or even completely disabled when not being able to scale up when needed [16].

B. Scope of Outages

Highly available systems are based on redundancy. Fault tolerant design distributes redundant components over different *failure domains* which isolate collections of resources from common failures. In earlier system design failure domains were created by physical separation and use of multiple power and network circuits. Cloud computing abstracts these physical attributes away thus customers cannot know the physical relation between any two computing resources. For example, a custom cannot determine whether a power circuit failure would affect multiple resources. Earlier designs could look at multiple failure domains – failure of a single server, failure of a rack, failure of a power domain and failure of a whole data center. These physical aspects of a data center organization are not visible when designing systems operating in cloud infrastructure services.

Cloud infrastructure providers mitigate the loss of physical visibility by introducing new abstractions that provide guarantees on isolation from common failure models. The new failure domain abstraction provided by Amazon Web Services is *availability zone*, where the guarantee is that resources located in different availability zones will not have correlated failures (power outages, network cabling problems etc.). This is attained by ensuring that any data center includes resources assigned only to a single availability zone[1].

From a cloud customer's point of view a cloud resource failure can manifest as an inability to access and use the resource or as inability to provision and manage the resource[2].

[1]Failure of a single data center can affect less than 100% of resources in the availability zone since a single availability zone may span multiple data centers.

[2]The resource may also be corrupted or permanently lost, but *durability* of the service is more difficult to assess, and is not included in the scope of this paper. This paper focuses on transient failures.

These failures may be caused by many different root causes acting on both individual server and availability zone levels:

1) Server hardware and virtualization software failures, affecting only cloud servers located in the failed machine.
2) Network failures, which can affect anything down from a single virtual server to a whole data center. These may be caused by physical network component failures, cascading network software failures, cabling losses, environmental effects etc.
3) Power failures, affecting anything from a single server or a rack of servers up to whole data centers.
4) Control plane software failures, which by definition are more likely to affect whole availability zones *or* whole regions. Control plane problems have a possibility to propagate through multiple availability zones either via data cascades – or administrative decisions intended to prevent spread of control plane data corruption as in [17].

This paper focuses on availability zone failures. The reason is that any fault tolerant computing system must be first and foremost tolerant against single server failures. Barroso et al. [13] observe that a single server will fail between 1.2–2 times a year, meaning that a cluster of 2000 servers will see an average of 10 server failures daily. When a system is tolerant of server failures it is worthwhile to look at the next failure domain, e.g. availability zone failures.

III. Methods and Data Sources

A. Failure Data Sources

Amazon Web Services provides a publicly accessible *Service Health Dashboard* [18] showing both current and historical information of up to five weeks on health of its services on a per-region granularity. Amazon Web Services does not publicly provide a long-term archive of notices on the service dashboard but it is possible to retrieve historical records via RSS aggregation services such as Google Reader [19].

In addition to the historical service health dashboard data that was retrieved from via Google Reader for this research, web searches were made to collect publicized Amazon Web Services failures. This information was later used to cross-correlate and verify the accuracy of the dashboard information.

A single dashboard RSS message consists of RSS source identifier (identifying service and region), publication time and a textual summary of the event (plus some additional information not relevant to this study). Below is a sample of a message published on February 16th 2013 relating to ELB service in the us-east-1 region:

> *3:27 PM PST*
> *Between 11:55 AM and 14:50 PM PST ELB experienced increased provisioning and back-end instance registration times in the US-EAST-1 region. The issue has been resolved and the service is operating normally.*

The summary text does not have a fixed format, and while some common conventions are followed the messages are not easily machine-parseable.

Data for the study was retrieved on February 22, 2013 and contained 1149 RSS messages dated from March 4, 2011 to February 21, 2013. From this data only those that were published between February 1, 2012 and January 31, 2013 (12 months) were chosen, and further narrowed down to contain messages only for EC2, EBS, RDS, ELB and S3 services. This subset contained 546 messages dated from February 13, 2012 to January 31, 2013.

B. Message Analysis

Selected messages were manually analyzed and broken down into individual events containing fields as shown in Table I. Amazon Web Services typically publishes multiple messages relating to a single event. A single message or a chain of messages may describe an event that affects one or more services. The message data analysis resulted in 172 distinct *events* affecting different regions and services. For some events more detailed information provided by Amazon Web Services was used in addition to information in the service health dashboard messages.

Not all of the messages could be cleanly categorized. Most common problems were:

- **Unknown affected availability zones.** Only 49 out of the 172 events had explicit information of the number of affected zones (if explicitly given, was always one[3]). Events without explicit information on affected availability zones were categorized as "not specified" – a value which is later handled specially (see below).
- **Non-quantifiable event impact.** Hard numbers on impact of events are rare in messages. Qualitative descriptions were common, such as "small number of instances", "small number of RDS instances", "some customers" and "some requests". Although some events did quantify a functional impact of the event (listing affected API endpoints), the amount of resources affected was quantified only once ("7% of the EC2 instances" in [20]). Due to lack of quantifiable event impact no attempt to was made to estimate failure impacts on smaller granularity than a single availability zone.
- **Value judgments.** For some events – for S3 in particular – it is difficult to quantify whether an event is a failure (customers incapable of using the service) or a degradation. S3 has two forms of access, with programmatic API access primarily used for resource management and HTTP GET interface primarily used to serve static data to end users. Software written using APIs must take potential API errors into account, already turning elevated error rates to a service degradation. Browsers do not retry a failed GET, so elevated error rates would cause user-visible errors for some end users.
  In this analysis S3 such ambiguous events are primarily categorized as degradations.

[3]The ELB event of December 24th, 2013 can be deducted reliably to have affected all five availability zones in the us-east-1 region, but again, this isn't explicitly stated by Amazon Web Services in dashboard messages related to the event.

Since the smallest failure domain considered in this analysis is a single availability zone, all events that have smaller impact (for example "some instances") are classified as failure of an availability zone. This leads to conservative estimates on the cloud service availability. Without quantitative information on the portion of resources affected by an event it is not possible within the scope of this paper to estimate speculate on the actual impact on all resources within the affected availability zone.

Two significant changes in the Amazon Web Services infrastructure occurred during the time period chosen for analysis. In September 2012 the number of availability zones increased from two to three in ap-northeast-1 (Tokyo) region. This change has been taken into account in service time calculation. In November 2013 the ap-southeast-2 region (Sydney) became available for general use. The Sydney region has been excluded from analysis on the basis that it was in customer use only for the last quarter of the time analyzed.

IV. Results

A. Overview of Events

In total, there was 546 messages published between February 13, 2012 and January 31, 2013. These messages corresponded to 61 failures and 111 service degradations. Summaries of failure event counts and durations by region and service are shown in Tables II and III. Further failure classification based on affected functionality (core, control and network) are shown in Figures 1 and 2.

From these graphs it is quite obvious that the us-east-1 region had more failures than all other regions combined. The reason for this is not known. Without more information it is possible to only speculate on the reason – it may be that us-east-1 could be overrepresented because it simply has more hardware than other regions.

As shown in Figure 2 EC2 failures are most common. EC2 failures are also dominated by network failures. This may be contrasted to lack of network failures for ELB and S3. This is probably due to physical constraints and implementation of different services[4].

From outage event durations (see Figure 2) we see that failures are dominated by core service failures. Although network failures dominate (34 network failures vs. 10 control and 18 core events), their combined duration is smaller (1d 11h 43m for network, 3d 19h for core and 1d 16h 57m for control). In practice this means that although network failures are most common, they have the least total impact due to shorter average failure duration (mean duration is 1h 03m for network failures, 5h 03m for core failures and 4h 05m for control failures).

B. Event Duration Distribution

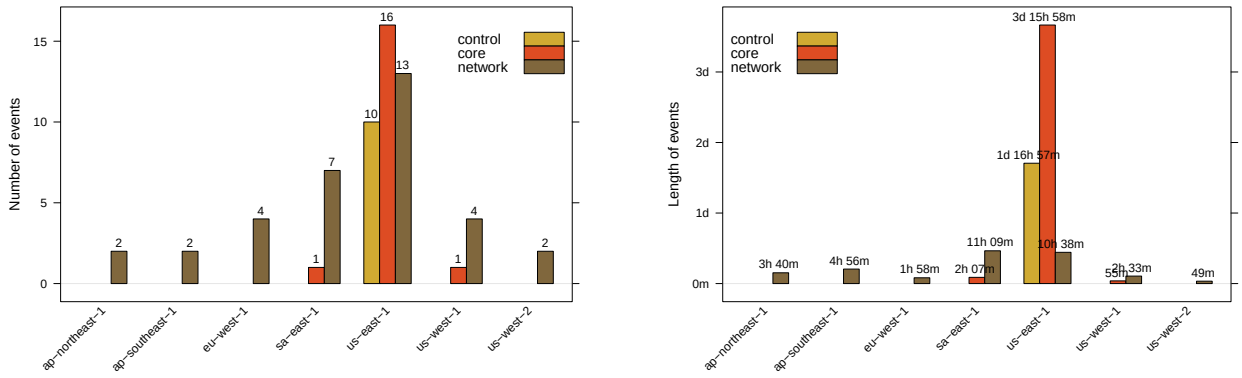
Figure 3 shows cumulative probability density over event duration. Half of the failures were shorter than 1h 06m

[4]ELB and S3 are inherently distributed services, whereas EC2 and RDS instances themselves are not distributed and thus more prone to be affected by network failures.

TABLE I
CATEGORIZED EVENT FIELDS

Field	Contents	Description
Event type	*degradation* or *failure*	Event type[a]
Region	*ap-northeast-1*, *ap-southeast-1*, *eu-west-1*, *sa-east-1*, *us-east-1*, *us-west-1* or *us-west-2*	Affected region[b]
Service	EC2, EBS, RDS, ELB or S3	Service that was affected[c]
Functionality	*control*, *core* or *network*	What was the affected area of the service[d]
Affected zones	Number or *not specified*	Number of affected zones if given
Event start	Date and time	Either explicitly from message, or publication time of the first message for the event
Event end	Date and time	Either explicitly from message, or publication time of the resolution message

[a] Degradations are events which cause the service to be usable, albeit at either increased latency, smaller performance, or with occasional retryable errors.
[b] These are correspondingly Tokyo, Singapore, Ireland, São Paulo, Northern Virginia, Northern California and Oregon.
[c] If a message described an event that affected multiple services, each service was recorded separately.
[d] These correspond to different types of failures. All cloud services interact with customers via a network, which is also clearly indicated as failed element in messages. Each service has a core which is what it actually does, e.g. for EC2 it is computation capability (being able to run OS and applications), for EBS being able to read and write data without corruption etc. Control means the mechanisms to provision, deprovision, monitor and manage core resources.



(a) Number of events

(b) Duration of events

Fig. 1. Number of events and their total duration by region. For each region the failed functionality is shown as a separate bar.

and 95% were less than 10h 19m long. Failure duration distribution exhibits a long tail, with the maximum event duration of over 22 hours.

C. Prorated Downtime and Total Service Time

We'll mark duration of an event $e \in E$ as d_e, the event time as t_e, the region affected as r_e, the number of services affected s_e and the number of zones affected as z_e. Each event occurs in some region $r \in R$, each region has set a set of services $S_r(t) \in S'$ (where S' is the set of all services) and zones $Z_r(t)$. Note that we consider the number of services in a region and number of zones as a function of time. The time under study is $t \in [t_0, t_1]$ so that $\forall e \in E : t_e \in [t_0, t_1]$. Thus, the prorated impact p_e is:

$$p_e = d_e \cdot s_e \cdot z_e \tag{1}$$

Conversely the service time $T_{r,s}$ for region r and service s is (where $\tau_{r,s}(t)$ is 1 if service exists in the region at time t and 0 otherwise):

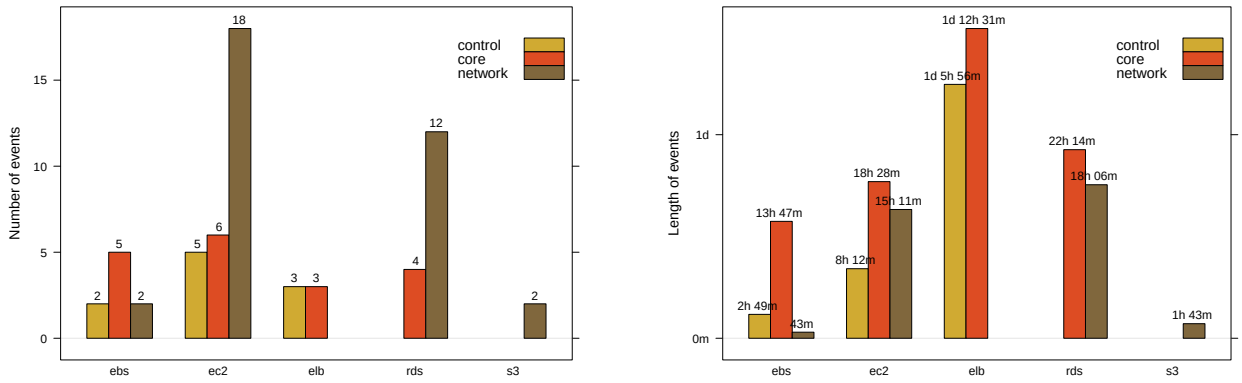
$$T_{r,s} = \int_{t_0}^{t_1} Z_r(t) \cdot \tau_{r,s}(t)\,\mathrm{d}t \tag{2}$$

and similarly for a single service over all regions $T_{*,s}$ and for all services in a single region $T_{r,*}$:

$$T_{*,s} = \sum_{r \in R} \int_{t_0}^{t_1} Z_r(t) \cdot \tau_{r,s}(t)\,\mathrm{d}t \tag{3}$$

$$T_{r,*} = \int_{t_0}^{t_1} Z_r(t) \sum_{s \in S_r(t)} \tau_{r,s}(t)\,\mathrm{d}t \tag{4}$$

These equations can be simplified as $\forall s \in S', r \in R, t \in [t_0, t_1] : \tau_{r,s}(t) = 1$ applies for the whole study period.



(a) Number of events

(b) Duration of events

Fig. 2. Number of events and their total duration by service. For each service the failed functionality is shown as a separate bar.

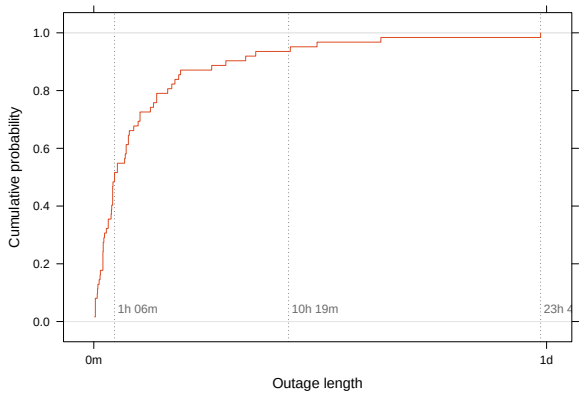


Fig. 3. Cumulative probability density for event duration. The three vertical lines correspond to 50%, 95% and 100% probabilities.

Similarly from the problem statement the set of services analyzed is closed: $|S'| = 5 \Rightarrow \forall r \in R : \sum_s = |S'| = 5$. Using these shortcuts we can determine the total service time for all regions and services:

$$T_{*,*} = |S'| \sum_{r \in R} \int_{t_0}^{t_1} Z_r(t)\,\mathrm{d}t \tag{5}$$

The total prorated downtime $P_{r,s}$ for region r and service s is:

$$P_{r,s} = \sum_{e \in E} p'_{r,s,e} \tag{6}$$

where $p'_{r,s,e}$ is defined as

$$p'_{r,s,e} = \begin{cases} p_e & \text{iff } s = s_e \wedge r = r_e \quad (7a) \\ 0 & \text{otherwise} \quad (7b) \end{cases}$$

Finally prorated downtime for any single region $P_{r,*}$ and service $P_{*,s}$ are defined:

$$P_{r,*} = \sum_{s \in S'} \sum_{e \in E} p'_{r,s,e} \tag{8}$$

$$P_{*,s} = \sum_{r \in R} \sum_{e \in E} p'_{r,s,e} \tag{9}$$

$$\tag{10}$$

The total prorated downtime over all regions and services degenerates to a sum of prorated downtime over all events:

$$P_{*,*} = \sum_{e \in E} p_e \tag{11}$$

Prorated availability A is based comparing prorated downtime against the total service time (from [10, p. 38], modified to use terms defined above):

$$A_{r,s} = \frac{T_{r,s} - P_{r,s}}{T_{r,s}} \tag{12}$$

For this analysis we are interested in availabilities of all services in different regions ($A_{r,*}$), availabilities of different services over all regions ($A_{*,s}$) and the total availability ($A_{*,*}$).

D. Region, Service and Total Availability

As noted earlier, a lot of the events do not explicitly specify the number of affected zones. Two sets of availability estimates are made with two different assumptions that events not specifying the number of affected zones either:

Case 1) affected **all** zones in the region, or
Case 2) affected one zone.

The first case is over-cautious and thus gives a lower bound availability estimate. The second case is likely to be more realistic based on the assumption that any large-scale (affecting multiple availability zones) outage would have gained a lot of media attention. No such news, *ergo* such outages are very unlikely to have happened. This may be an over-optimistic position, but it gives a higher bound for the availability estimates.

The zone availability estimates are shown in Tables V and IV (for services and regions, respectively). Availability of the EC2 service is 99.949–99.977% (low–high). ELB's widely different availability values (99.843–99.963%) between models is based on a single which began on December 24th, 2012 and lasted for over 22 hours and which didn't specify the number of affected zones for all affected functional areas.

Regional availability estimates show that the availability of us-east-1 region is lowest of all regions: 99.803–99.936% versus 99.998–99.999% for us-west-2 region. Availabilities of other regions were between.

The total availability $T_{*,*}$ is 99.947–99.981%.

Deriving service availability statistics for individual regions is not meaningful due to the small amount of events in some regions. Tokyo, Singapore and Oregon all had only two failures. Without long-term statistics it is not possible to determine whether these were caused by excellent data center management, or by combination of fewer customers and sheer luck.

V. Error Sources

Since this work relies on public, not easily quantifiable data produced according to Amazon Web Services' internal operating procedures, which are executed by people working in (presumably) busy situation, there are a lot of identified potential sources errors. Those sources of errors are listed below:

- **Missing messages**. It is possible that Google Reader does not have a complete archive of Amazon Web Service Dashboard messages.
- **Underreporting of incidents**. Amazon Web Services might not be reporting all problems, for example if the number of affected instances or customers is below some threshold.
- **Overestimating the impact of incidents**. In this analysis even the smallest failure or degradation is considered to impact at least one zone. On one incident in us-east-1 region the amount of affected instances was 7% of the region capacity and not 20% as could naively be assumed (us-east-1 has 5 zones).
- **Inaccurate event duration**. Amazon Web Services does provide explicit event start and stop dates for some events, whereas for some events these are determined from message timestamps. Biases in these timestamps would cause event duration to be incorrectly estimated.
- **Message categorization errors**. Should elevated S3 API error rates be a failure or degradation?

VI. Related Work

There has been little scientific publications related to cloud service availability from the viewpoint of customers. On the other hand there are lots of papers about *some* empirical aspect of cloud computing and some of these papers do actually report aspects of cloud service availability and reliability. Some of these are summarized below:

- Palankar et al. ran 215112 API operations on S3 and reported a total of 35 errors (timeouts or error codes) implying a 99.984% success rate for S3 API calls [21].
- Tung and Kang analyse availability between hot-hot and detect-restart systems [22]. They additionally include information about the *mean time to provision* a replacement server which is quite close to mean time to repair (MTTR) that can be used for estimating system availability.
- Ostermann et al. have benchmarked EC2 instances for performance, providing also time to provision new systems [23]. They also note being incapable of provisioning new EC2 instances for a whole day.

Cloud vendors themselves naturally have detailed knowledge of their service history. Similar records may exist with cloud service aggregators and heavy cloud infrastructure service users such as Rightscale, Netflix and Zynga. Also some public services reporting public cloud availability information exist such as CloudSleuth. With publicly available information it seems that these services monitor aggregate network reachability of different servers. This provides more accurate information on server level availability, but does not directly translate relate to estimating availability zone failure rates.

VII. Conclusions and Future Work

In this paper I have analyzed 12 months of Amazon Web Service's service health dashboard data. The result of availability for a single availability zone of 99.947–99.981% is categorized as between "well managed" and "fault tolerant" according to [10] and is equivalent to either availability of Tier 2 or Tier 3 data centers according to [24]. There is large variability between regions and services with some regions meeting 99.99% availability or better.

These values can be used in estimating availability of a service deployed in *a single or multiple availability zones* but the limitations of this study and the presented values must be also understood. In particular this paper does not consider *server-level failures* which occur due inherent unreliability of the computing hardware in use. The values presented in this paper are meaningful only in context where the service is already fault tolerant against single server failures.

The original service dashboard data contains messages for several other services (CloudFront, Auto Scaling, CloudWatch) that were excluded from this analysis. Service degradation probabilities were calculated from the analyzed data, but omitted from this paper for brevity.

It would be interesting to combine the estimates presented in this paper to observed cloud server failure rates in a large-scale cloud system availability simulation. This kind of model could be used to evaluate the effect of different availability estimates on operational costs and service availability.

References

[1] L. Columbus, "Cloud computing and enterprise software forecast update, 2012," http://www.forbes.com/sites/louiscolumbus/2012/11/08/cloud-computing-and-enterprise-software-forecast-update-2012/, Nov. 2012. [Online]. Available: http://www.forbes.com/sites/louiscolumbus/2012/11/08/cloud-computing-and-enterprise-software-forecast-update-2012/

TABLE II
NUMBER OF FAILURE EVENTS BY REGION AND SERVICE

Type	Service	Region							Total
		ap-northeast-1	ap-southeast-1	eu-west-1	sa-east-1	us-east-1	us-west-1	us-west-2	
Failure	ebs	0	0	1	1	7	0	0	9
	ec2	1	0	1	5	17	4	1	29
	elb	0	0	0	0	6	0	0	6
	rds	1	2	2	2	8	0	1	16
	s3	0	0	0	0	1	1	0	2
Total counts		4	39	2	2	5	8	2	62

TABLE III
DURATION OF FAILURE EVENTS BY REGION AND SERVICE IN MINUTES

Type	Service	Region							Total
		ap-northeast-1	ap-southeast-1	eu-west-1	sa-east-1	us-east-1	us-west-1	us-west-2	
Failure	ebs	0	0	30	127	882	0	0	1039
	ec2	110	0	29	289	1901	162	20	2511
	elb	0	0	0	0	3987	0	0	3987
	rds	110	296	59	380	1546	0	29	2420
	s3	0	0	0	0	57	46	0	103
Total lengths		118	8373	220	49	208	796	296	10060

TABLE IV
PRORATED DURATION OF FAILURES BY REGION

Region	Service time	Unknown number of zones is all zones[a]		Unknown number of zones is one zone[b]	
		Down time	Availability	Down time	Availability
	(days)	(minutes)	(percent)	(minutes)	(percent)
r	$T_{r,*}$	$P_{r,*}$	$A_{r,*}$	$P_{r,*}$	$A_{r,*}$
ap-northeast-1	4275	220	99.996	220	99.996
ap-southeast-1	3660	357	99.993	296	99.994
eu-west-1	5490	234	99.997	118	99.999
sa-east-1	3660	1053	99.980	796	99.985
us-east-1	9150	25941	99.803	8373	99.936
us-west-1	5490	514	99.993	208	99.997
us-west-2	5490	147	99.998	49	99.999
Total	37215	28466	99.947	10060	99.981

[b] In this model, events without number of affected zones are assumed to affect all zones in the region.
[c] In this model, events without number of affected zones are assumed to affect one zone.

TABLE V
PRORATED DURATION OF FAILURES BY SERVICE

Service	Service time	Unknown number of zones is all zones[a]		Unknown number of zones is one zone[b]	
		Down time	Availability	Down time	Availability
	(days)	(minutes)	(percent)	(minutes)	(percent)
s	$T_{*,s}$	$P_{*,s}$	$A_{*,s}$	$P_{*,s}$	$A_{*,s}$
ebs	7443	1715	99.984	1039	99.990
ec2	7443	5456	99.949	2511	99.977
elb	7443	16855	99.843	3987	99.963
rds	7443	4017	99.963	2420	99.977
s3	7443	423	99.996	103	99.999
Total	37215	28466	99.947	10060	99.981

[a] In this model, events without number of affected zones are assumed to affect all zones in the region.
[b] In this model, events without number of affected zones are assumed to affect one zone.

[2] P. Mell and T. Grance, "The NIST definition of cloud computing," *National Institute of Standards and Technology (NIST)*, 2009, mel09.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, p. 50–58, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1721654.1721672

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

[5] B. Wei, C. Lin, and X. Kong, "Dependability modeling and analysis for the virtual data center of cloud computing," in *2011 IEEE 13th International Conference on High Performance Computing and Communications (HPCC)*, Sep. 2011, pp. 784 –789.

[6] Q. Guan, C.-C. Chiu, and S. Fu, "CDA: a cloud dependability analysis framework for characterizing system dependability in cloud computing infrastructures," in *2012 IEEE 18th Pacific Rim International Symposium on Dependable Computing (PRDC)*, Nov. 2012, pp. 11 –20.

[7] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proceedings of the 1st ACM symposium on Cloud computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, p. 193–204. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807161

[8] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2011, pp. 104 –113.

[9] P. O'Connor, *Practical Reliability Engineering*. John Wiley & Sons, Jul. 2002.

[10] E. Bauer and R. Adams, *Reliability and Availability of Cloud Computing*. Wiley-Blackwell, Sep. 2012.

[11] Amazon Web Services, "Products & solutions," http://aws.amazon.com/products-solutions/, 2013. [Online]. Available: http://aws.amazon.com/products-solutions/

[12] W. E. Smith, K. S. Trivedi, L. A. Tomek, and J. Ackaret, "Availability analysis of blade server systems," *IBM Syst. J.*, vol. 47, no. 4, p. 621–640, Oct. 2008. [Online]. Available: http://dx.doi.org/10.1147/SJ.2008.5386524

[13] L. A. Barroso and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–108, Jan. 2009. [Online]. Available: http://www.morganclaypool.com/doi/abs/10.2200/S00193ED1V01Y200905CAC006

[14] A. Undheim, A. Chilwan, and P. Heegaard, "Differentiated availability in cloud computing SLAs," in *2011 12th IEEE/ACM International Conference on Grid Computing (GRID)*, Sep. 2011, pp. 129 –136.

[15] R. Jhawar and V. Piuri, "Fault tolerance management in IaaS clouds," in *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*, Oct. 2012, pp. 1 –6.

[16] A. Cockcroft, "A closer look at the christmas eve outage," Dec. 2013. [Online]. Available: http://techblog.netflix.com/2012/12/a-closer-look-at-christmas-eve-outage.html

[17] Amazon Web Services, "Summary of the december 24, 2012 amazon ELB service event in the US-East region," http://aws.amazon.com/message/680587/, Dec. 2012. [Online]. Available: http://aws.amazon.com/message/680587/

[18] ——, "AWS service health dashboard," http://status.aws.amazon.com/, 2013, ama11c. [Online]. Available: http://status.aws.amazon.com/

[19] Google, "Google reader," http://www.google.com/reader/, 2013. [Online]. Available: http://www.google.com/reader/

[20] Amazon Web Services, "Summary of the AWS service event in the US east region," http://aws.amazon.com/message/67457/, Jul. 2012. [Online]. Available: http://aws.amazon.com/message/67457/

[21] M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon s3 for science grids: a viable solution?" in *Proceedings of the 2008 international workshop on Data-aware distributed computing*, ser. DADC '08. New York, NY, USA: ACM, 2008, p. 55–64, ACM ID: 1383526.

[22] T. Tung and J. Kang, "Characterizing service assurance for cloud-based implementations: Augmenting assurance via operations," in *SRII Global Conference (SRII), 2012 Annual*, Jul. 2012, pp. 21 –28.

[23] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of EC2 cloud computing services for scientific computing," in *Cloud Computing*, ser. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, D. R. Avresky, M. Diaz, A. Bode, B. Ciciani, and E. Dekel, Eds. Springer Berlin Heidelberg, Jan. 2010, no. 34, pp. 115–131. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-12636-9_9

[24] W. P. Turner IV, J. H. Seader, and V. E. Renaud, "Data center site infrastructure tier standard: Topology," *Uptime Institute*, 2010.

Impact of Cloud Computing on Global Software Development challenges

Inna Smirnova
Department of Computer Science
University of Helsinki
Helsinki, Finland
inna.smirnova@helsinki.fi

Abstract - **The paper introduces challenges in Global Software Development and application of Cloud computing platforms as a solution to some problems. Global software development (GSD) means geographical distribution of development teams across the globe that might lead to economical benefits and opportunity of 24-hours development process. However, GSD has to deal with many challenges such as coordination, communicational barriers, geographical distances and time differences. Based on existing literature the paper gives an overview on Cloud computing as a form of IT-outsourcing that is believed to help to overcome some GSD problems. Cloud's components Infrastructure-as-a-service, Software-as-a-service, Platform-as-a-service and Tools-as-a-service along with their benefits for global software development will be presented as main resolves. Additionally, the paper will present some real examples of GSD prototypes for the cloud.**

Cloud computing; global software development; GSD challenges (key words)

I. INTRODUCTION

Nowadays software development is carried out globally. New development teams consist of developers who are distributed across the globe with location even at different continents. Boundaries between countries in modern software development are disappearing, and software development process is becoming globally oriented, multidiscipline and cross-cultural. This new approach is called Global Software Development (GSD). GSD's popularity among IT businesses is growing rapidly because of potential economical benefits as development outsourcing to China or India might make software development process much cheaper and faster [8, 5].

Although costs' benefits seem to be so attractive, GSD brings some serious challenges for development teams. Challenges referred to geographical distances, different time zones, unified coordination problems, cultural misunderstanding and communicational barriers will be overviewed in the paper. There are few recommendations which could eliminate negative effects of these problems such as overlapping working hours, video conferences with support of asynchronous communication, some face-to-face visits. But none of the suggestions solves challenges fully and in an effective way.

This paper will present Cloud Computing paradigm and its components as a solution to defined GSD problems. Cloud can provide to development teams the whole set of computing resources and tools (Infrastructure-as-a-Service, Tools-as-a-Service), fully equipped platform for applications' development (Platform-as-a-Service), some existing apps and software components (Software-as-a-Service), huge storage capacities with effective data management services on a pay on demand basis [5, 7, 10]. Cloud facilities are believed to overcome some GSD challenges such as lack of powerful resources, geographical distances and data exchange delays between different development sites [5]. Using of cloud components might eventually reduce software development costs and consuming time that could lead to resources' saving for the company and improvement of software quality for the end customers. However, using of Cloud Computing brings some risks and difficulties including some political law restrictions [2]. Cloud advantages and risks for GSD will be described in this paper.

Currently there are some research and development of prototypes in the area of application of cloud entities in software development process. Last chapter of the paper will give some real examples of GSD prototypes for the cloud which have already found successful application in software development [9].

II. GLOBAL SOFTWARE DEVELOPMENT AND ITS MAIN CHALLENGES

In today's world economy and cooperation national boundaries are getting lost and trade market is becoming global. It refers to almost all industrial areas including software products market. In order to answer to this tendency, software development process is changing and becoming worldwide oriented and multicultural. This modern stage in software development is called Global Software Development or GSD in short. GSD means the development process where developers are geographically distributed across the world and working together at one project. GSD format now is considered as the most popular model for software development that suits very well to medium or large size organizations [4].

The reasons for GSD universal popularity include potential significant economic benefits for businesses and fulfilling current global trade needs [4]. GSD can allow to companies to decrease software development costs that leads to reduction of software prices for end customers. Also it provides the opportunity for 24/7 sun-followed development that might save development resources and allow to get faster product releases in some projects. At the same time, there may be some obligations for companies for using GSD model. Lack of trained, experienced, highly qualified developers enforces IT companies to use outsourcing as a way to hire best developers. Customization needs also may require locating development offices close to end customers in order to fulfill their requirements in a better way and discover market better. Moreover, country level law regulations might be a reason for GSD. For instance, some specific governments do not allow selling product locally unless the development process is held in that country [8, 5].

As a result, GSD is believed to facilitate business growth and make product to appear faster at the market place. Furthermore, GSD for company is an opportunity to use resources in other geographical regions. That's why it is dramatically popular nowadays to outsource development to India or China – countries with quite low economy and low costs in general [5]. Even so, with

all the potential benefits global development raises some serious challenges [6].

Mockus et al. [8] present these major GSD problems:

- Differences in provided infrastructure at different development sites including lack of computing resources or network connection capabilities.
- Coordination and management problems. Development teams are quite independent at different sites; they have their own development process strategy that generates some project goals' misunderstanding. Because of time zones and lack of project related discussions (especially face-to-face meetings) work can be overlapped partly, and cost of software will be increased as a consequence. Large geographical distribution of developers' teams results in absence of formal unified development process and hard process monitoring.
- Communicational problems are considered as principal and most difficult to solve challenges of GSD. Cross-cultural communicational barriers appear because development teams consist of developers of different nationalities, cultures, background, mother tongues. Developers at different continents have different work experience, they participated in distinct IT projects, had other training, style of software implementation, business and software development culture [5]. Furthermore, lack of face-to-face synchronous communicational contacts with remote colleagues makes this barrier even stronger.
- Finding a right remote person who is responsible for needed particular expertise results in delays in software product development.
- Working hours of distributed development teams are often not overlapped, so developers have to use only asynchronous communicational tools such as emails, chats, forums, etc. However, it is hard to be absolutely sure if email will reach recipient in time and will be answered soon, so additional time for waiting for the reply is wasted. Moreover, in urgent situations only using asynchronous communication is impossible and can fail the whole project [8].
- Information and knowledge should be distributed in an effective way between development sites without any delays or losses. Large amounts of data should be exchanged, then processed and stored properly [6].

Altogether, according to Hashmi et al. [5] GSD challenges refer to global distance that can be divided into four main dimensions:

1) Geographical distance as a result of development countries distribution.
2) Cultural distance and misunderstanding.
3) Linguistic distance. Mother tongues of developers are different, speech speed and manner can vary a lot, so even talking in common for IT industry English language can be problematic.
4) Time distance.

The next chapter of the paper will present Cloud Computing paradigm and its components as possible main resolves for described earlier Global Software Development challenges.

III. APPLICATION OF CLOUD COMPUTING IN GSD

Cloud computing paradigm recently became quite common useful tool in software development. Cloud is presented as an abstract infrastructure run by many servers that can store apps of cloud users, resources, services and be always accessible by end users on demand. Cloud computing is described as a model of delivery of all cloud facilities to end users via internet on a "pay-as-you-go" base [9, 5]. Cloud computing can provide to developers services, platform for software development, computing infrastructure. Cloud providers own virtual machines which can be located anywhere in the world, and facilities are always available for consumers via internet. Developers can use the cloud services and pay only for the time they actually used resources. Another advantage is that developers can access the cloud from any location in the world. So cloud computing can assist GSD and make some challenges less noticeable [3].

Depending on type of provided resources the cloud can include different components such as Platform-as-a-Service, Infrastructure-as-a-Service, Software-as-a-Service or Tools-as-a-Service which can give potential benefits to GSD. The next section of the chapter will present the overview of above mentioned cloud computing components with their advantages for Global Software Development.

A. Components of cloud computing and their benefits

Hashmi et al. [5] describe **Infrastructure-as-a-Service** (IaaS) cloud computing component as a cloud facility that gives to end users access to the set of hardware resources, including their administration and maintaining. So IaaS offers a whole infrastructural platform with hardware resources in outsourcing. With IaaS benefits all the development work might be done on virtual machines which are run by servers at cloud provider's site. Machines collect all the computing facilities needed for development team [2]. In the issue, company can save money on computing resources, their administration and maintaining. Moreover, all the teams included in GSD project will have a scope of powerful resources of the same kind, so the challenge of infrastructure differences between teams might be overcome.

Platform-as-a-Service (PaaS) gives to the user an access to full platform with development environment where user can build his own application [3, 10]. With using PaaS: resources and infrastructure are available; maintaining and system updating are carried out by cloud computing providers; testing environment with real life industrial simulation is available anytime and can be easily accessible by any team member. So everything for software design and implementation, including programming capacities, is offered by the cloud. Consequently, development process speed is getting higher. Moreover, delivery to end customers is easily done from the cloud via World Wide Web. As a result, development costs and time might be reduced sharply [3, 5].

Actual software development ordinarily consists of following steps: design, developing and implementation, testing, deployment and monitoring. In GSD some of the actions are offshored and distributed randomly across the globe, so it might be hard to interpret and integrate all the software components when they are collected finally at onsite development team. Especially it is problematic when infrastructure and resources are distinct within involved teams. With PaaS we can achieve standardization of the development tools, services and processes. PaaS allows getting portability that leads to no need for special effort in integration and synchronization of software parts [10].

Also PaaS provides an online always available storage in the cloud that can be used for communication between offshoring teams. PaaS allows shared source code development, information about whole project history (with all the changes), discussion forums. Additionally, developers get common programming tools for development, automatic system updating, and administration of work environment [10].

Based on Stankov and Datsenka [10] all major PaaS benefits for GSD can be presented in following Table 1.

TABLE 1. GSD COMPONENTS AND PAAS CHARACTERISTICS [10]

GSD element	PaaS characteristics
Development	Common programming languages, shared source code, project history, automatic system updates
Delivery	
Capacity management	Cloud's scalability, powerful resources, opportunity for many development teams to be involved simultaneously
Availability	Always accessible on demand, 24/7 product delivery to end customers
Financial management	Pay-as-you-use base
Collaborative environment	Discussion forums and social networking inside the cloud, safe cloud's storage for data and knowledges

Today cloud component PaaS is getting more and more popularity among software developers. There are some currently existing commonly used industrial platforms such as Google AppEngine, Microsoft Azure Platform, Force.com, Heroku, Bungee Connect. All the platforms have similar characteristic, but use slightly different approaches to software development, e.g. different programming languages offered – some use Java and Python, some use Ruby [10].

Software-as-a-Service (SaaS) component means that fully working ready applications are stored on web servers and can be accessible by developers and end users through WWW online [3, 5]. Thus development teams can use some already existing software components for their own product development. It may conduct to software development costs' elimination, avoiding duplication work. With SaaS reuse of software as a main goal of lean development is achieved [5].

SaaS is aimed to reach centralized, unified, coordinated development platform for development teams with all the services available at delivery time. So with SaaS there are no delays because of local synchronization, there is no need for special onsite coordination. Also, local customization at developer's site is provided by the cloud service. Altogether, SaaS can be a good optimization of technical tools for distributed developers. SaaS concept is a fine-drawn solution for availability, accessibility, data synchronization GSD challenges. It helps to achieve one standardized app environment for immediate software development without need for initial set up of new software and administration at GSD multi sites [7].

Chou et al. [1] make an economical analysis of using SaaS for software development process in IT companies. Chou et al. describe SaaS as a new model for software delivery that is based on service oriented approach. The research describes next benefits of outsourcing with SaaS:

- Costs reduction. Costs of needed resources are minimized because of usage of already existing apps and tools rather than developing everything from the initial step.
- Project budget optimization. Firm can invest money saved on resources and network administration into realization of other business strategy goals.
- Scalability. Cloud SaaS providers allow immediate, almost unlimited scaling up the number of users at customer's site. It leads to the fact that more customers could get a desired developed application, and IT company could get more profit as a result.
- SaaS services can be physically located anywhere and still will be accessible easily virtually over the network. So it is a nice potential tool for overcoming geographical distances of global outsourcing.

One of GSD challenges is lack of development tools and strong variation in tools at different development sites. Moreover, different development stages need different tools which are not needed at other steps, such as requirements management tools, testing tools, design tools, or data transferring tools. If cloud could provide **Tools-as-a-Service** (TaaS), this challenge might disappear. So TaaS cloud component can provide appropriate tools for effective development, collaboration and easier process coordination [4].

Chauhan and Babar [4] emphasize following TaaS advantages for Global Software Development. Some software development stages need specific tools and some tools are used just once for a short period of time. However, unfortunately organization still has to keep all the tools' set during whole project development period. Also organization might have projects in completely opposite areas with wide variety of distinct tools needed. With TaaS company could pay only for exact usage of specific tools, and if more new tools are needed they are easily offered by TaaS providers. So development process might become more convenient and time for finding particular resources could be saved. TaaS can be especially beneficial for small organizations with small sized occasional software development projects [4].

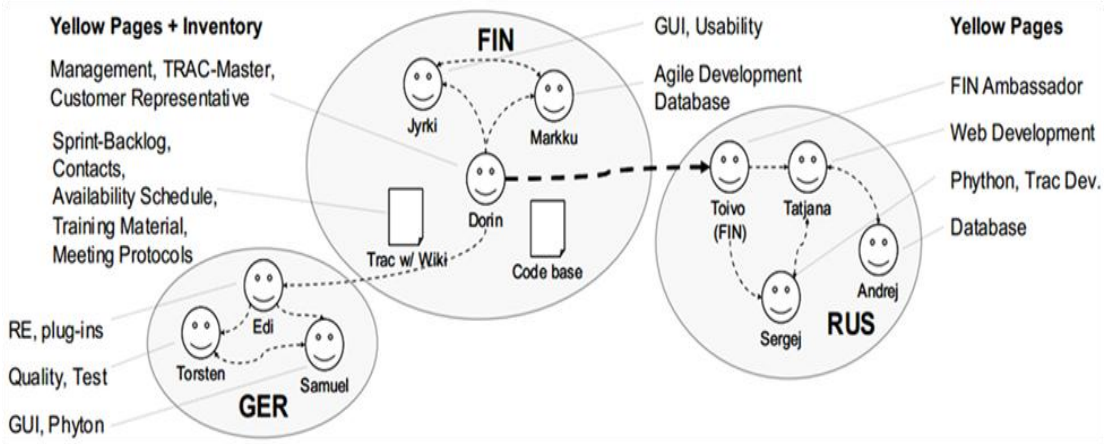
TaaS can provide a data storage with huge capacity and effective data transfer management. There are some countries' law regulations saying that some data is prohibited to be moved outside country territory. Using TaaS, data can still be located inside the country physically, but at the same time it can be accessible from outside locations. So data law problems for GSD are believed to be minimized, and there are less product development delays possible.

TaaS could be used as a collaboration tool within GSD teams. Usually asynchronous communication, e.g. emails, chats, forums, does not work well enough. There is no visualization, it is impossible to trace the whole work chain of another developer. Moreover, it is very difficult to find a right person to contact for asking a question about specific detail. So consequently it is hard to observe a stage of the whole project. TaaS services can provide visualized tools with big interactive map and all data flow within project [4]. One of the approaches could be drawing a FLOW map that was proposed by Stapel et al. [11]. FLOW map (Fig. 1) presents an interactive map with all the GSD participants, information about each developer including his contacts and area of expertise – yellow pages, data flow exchanged between parties. As a result of using TaaS as a method for effective collaboration and communication, developers get more awareness, knowledge about project and more trust between development sites. That might lead eventually to optimization of software development process and product quality [4].

Even so, in order to be a beneficial for GSD teams TaaS should satisfy to some requirements. Chauhan and Babar [4] define the following set of essential requirements. First of all, multi-tenancy refers to the fact that many organizations and even many people in one organization might be using the same tool. So services and tools should be isolated from each other. Safety for using and storing private data is essentially important for developers.

TaaS should be able to support different versions of the tool. Some development teams might require different versions of one particular software tool than others, so TaaS providers need to maintain all the history of software releases. Moreover, TaaS should be able to combine and integrate tools used at different development steps, so end users will be able to get full complex equipment.

FIG. 1 FLOW MAP EXAMPLE [11]



Developers might need an access to tools by devices of different type such as laptops, mobile phones with different operating systems like Android or iOS. So the cloud should be able to provide synchronization services. Furthermore, requirements of different organizations for the same tool can vary dramatically. Quality expectations for different projects might be different. So TaaS services should support flexibility.

Next section will sum up all the beneficial outcomes of using cloud computing outsourcing model for Global Software Development projects.

B. *Outcomes with global development with outsourcing in the cloud*

The past section has already introduced principal benefits of cloud components for software developers and GSD teams particularly. In total, the biggest potential impact of cloud computing in GSD could be the elimination of development costs and resources' savings. Additionally, the cloud may reduce problems with providing full infrastructure for each GSD site. It also provides scalability as it can serve in an isolatable base large amount of users simultaneously with guarantee of data security and privacy that is essential for big GSD teams [3].

Table 2 gives a clear summary of Cloud Computing characteristics and their advantages for developers.

Cloud Computing paradigm can offer the architectural solution for organizing operative work for many separate GSD teams together at once. Whole cloud space could be virtually divided, and different GSD teams might be connected to one private cloud through which all the information is exchanged, stored, and even locally translated with multilingual services. This architectural cloud model helps to reduce main geographical and temporal GSD problems. Moreover, project data safety and reliability occur in addition [5].

Hashmi et al. [5] point out a nicely summarized GSD challenges and requirements for related cloud computing components in order to overcome some GSD difficulties:

1) Coordination and management in GSD format are quite challenging because of negative issues of geographical and time distances. It could also affect on product delivery to end customers. However, with cloud usage product delivery is independent from development process, and developed software product can be accessible all the time without special coordination actions from development teams.

TABLE 2. CLOUD COMPUTING CHARACTERISTICS [5]

Virtualization, scalability and performance	Cloud providers run services for users on virtual machines. So they can effectively accommodate more users on their infrastructure if there is high services' demand.
Reduced costs	Pay-as-you-go model might significantly reduce investments, especially on computing facilities and their administration.
Infrastructure	Cloud provides fully equipped infrastructure to developers that is always accessible. At the same time cloud provides an infrastructure for easy delivery of new implemented apps to end consumers.
Multi-tenancy support	Cloud users' threads are isolated from each other and privacy is guaranteed.

Furthermore, with using IaaS GSD teams can get unified functional resource capabilities between all included development teams, so the administration efforts will be minimized. Safe data storage provided by the cloud is also aimed to help overcoming of data and knowledge management problems [5, 6].

2) Geographical distance - principal GSD challenge - can be supported by using PaaS cloud services. PaaS grants to developers a platform for apps development and shared resources. That hopefully might lead to speeding up product development cycle without any delays for software installation at every distributed GSD site.

3) Data and knowledge exchange within GSD teams. Cloud storage and project memory can assist GSD in maintaining all the project activities, changes, tasks completing. With cloud it is becoming pretty transparent to see on what particular stage the whole project is currently. Furthermore, project monitoring may become simplified. Altogether, it results in creating more trust between GSD members.

4) Main technical advantage for GSD from the cloud could be an easy testing procedure. Usually in software development this is a quite costly and important phase. Cloud can provide large scalability and possibility of testing on real industrial systems. That advanced cloud based testing might lead to better software product quality, less future bugs' appearance and increasing of company's profit [5].

However, together with cloud computing favors for Global Software Development, some cloud outsourcing risks show up. The next section of this chapter is devoted to the discussion of those risks.

C. Risks and challenges of cloud computing for GSD

Based on Clemons et al. [2] cloud computing risks for software development can be viewed in three main dimensions:

1) **Opportunism risks**

Shirking risk means that user's payment and cloud provider's efforts are not equal, e.g. provider might blame slow network capacity, but it is actually the own fault cloud provider.

Poaching and intellectual property's stealing refers to the fact that user is not sure if cloud provider uses his data in a secure and fair way. Cloud end user cannot be completely sure about privacy while using cloud services.

Vendor's renegotiation of contract - sudden provider's lock-in and saying new prices for the continuation of services' usage. As app was created at one specific vendor's platform, user cannot easily move his app to another platform in order to overcome paying new high prices. [2]

2) **Technological development risks**

Functionality risk. There is no advance knowledge if applications created with using PaaS services and in traditional onsite way will integrate with each other. Also, there is no guarantee that apps implemented at different cloud platforms can be synchronized. It at the same time refers to data transfer across the national borders. Some laws and special countries' regulations might not allow outside data moving. So full functionality of the cloud in the future is absolutely unknown for now.

Political risk includes an employment situation in whole IT industry after using cloud facilities. Cloud's popularity may lead to future no need for system administrators, network maintainers, or low level software development personnel in general. In all, the situation of unemployment crisis in the industry might occur.

Project risk. Development team cannot be sure how the fully integrated combined software product will work after development process finishes. The main question is if it is going to be synchronized properly at customer's site or not?

Financial risk. The principal point with using cloud computing facilities for GSD purposes is if the cloud is beneficial and profitable for the company and development project at the end or not. For small, new firms cloud benefits are pretty obvious because they will eliminate costs on resources, personnel, and allow releasing software product faster. But for large already established businesses no one can predict the economical advantages of the cloud for organization's profit. [2]

3) **Special risk** of cloud paradigm for GSD refers to *cross border litigation*. If something goes wrong with cloud services, in what country the court should be happening – provider's, actual cloud service's location or end user's? This risk leads to need for standard cross-national law about cloud services, including data storing and transferring regulations [2].

IV. EXPERIMENTS OF GSD PROTOTYPES FOR THE CLOUD

At the present time combination of areas of Global Software Development and cloud computing in real software development are in a focus of many researches.

Pavan [9] and his research group present some very interesting and useful examples of commercial prototypes of GSD format with cloud platforms. The first prototype they developed is called *Compile Server Farm*. For large software projects with parts at different world sites compilation process might take a lot of time while team just has to wait. After compilation there are may be some bugs, and since fixing them, long compilation cycle can repeat again. Besides, compilation of large projects requires development team to have powerful server machines or even clusters which are quite expensive to have and maintain. So Pavan et al. propose to use cloud services for compilation of the software projects. Their presented experimental prototype helps to speed up Compile-Test cycle using Hadoop and Condor methodology. Testing prototype showed that compilation time for the project in average reduced quite dramatically from 150 minutes to 80 minutes [9]. So cloud computing paradigm for compilation might be very beneficial for large GSD projects.

Another example of the prototype created by Pavan's group is *Online Storage Cloud*. Online storage with big capacity is an essential entity for GSD as the process usually contains huge amount of data to store and exchange between parties. All the data present themselves unified, correct, and shared global knowledge within whole organization. Knowledge is a fundament of organization. So project knowledge needs a good infrastructural storage that should be scalable, reliable and effective for data flow coordination [6, 9]. Cloud based storage is provided online, so there are no problems for GSD in access it from different remote locations. Data transfer via cloud is easier and much faster than usual emails, for example. Cloud storage capacity is huge and can be aimed for terabytes of information with guarantee of high reliability level. Pavan group's prototype presents to developers a secure private storage service with good performance characteristics that is delivered via Internet. In addition, cloud based storage can allow streaming media data, so prototype could be used for stimulating face-to-face video calls or conferences [9].

Furthermore, Pavan's group designed a *Lab Any Where (LAW)* prototype for Online Virtual Lab services. For big sized companies with many employees job training is very important, especially for new coming workers or completely new projects. Traditional way of training delivery via physical classes with lectures does not work with GSD model where teams and lecturer can be located too far from each other. Moreover, it is pretty expensive to organize those training sessions, especially on regular basis. Online e-learning solution might be quite convenient, but the experience of developers with real software systems is always missing. For effective training results employees should have a real interaction with software systems which they are going to work with later on. LAW prototype is cloud based online training platform that provides comprehensive technical training and testing scenarios via internet. As a result, training might become rapid and based on actual interaction with the software system. It could make the educational process for developers more efficient generally [9].

Presented examples show the variety of areas of application of cloud computing in software development. It leads to the conclusion that research field of Global Software Development based on cloud advantages in the nearest future might be in a process of progress evolution.

V. CONCLUSION

In this paper, the outcomes of using Cloud Computing paradigm in Global Software Development (GSD) were considered. GSD concerns the model of software development with developers physically distributed around the globe. GSD format is believed to bring many benefits to software development process such as resources' saving, reduction of software costs, speeding up the release of end product. However, together with advantages GSD has to deal with some negative parts of geographical development distribution. According to Hashmi et al. [5] GSD global challenges can be divided into four principal dimensions: geographical distance, cultural misunderstanding, linguistic distance (because of differences in mother tongues of developers) and time distance (different time zones at multi sites).

As a way to support GSD model and overcome its main difficulties, Cloud Computing paradigm was outlined in current paper. Cloud provides scalable storage facilities, software

development platform and tools, existing apps and other services that can be accessible by end users over the internet any time at pay-as-you-use basis. IaaS, PaaS, SaaS and TaaS components of cloud computing were described in a paper as possible effective solutions for main GSD barriers. Table 3 sums up potential advantages of the cloud for GSD major challenges.

Summing up in all, cloud computing paradigm seems to be quite a nice resolve for developers involved in GSD projects. Principal outcomes of using cloud based software development include: almost no investment into infrastructure and minimum investments into office hardware/software working tools; saving of costs because of "pay-as-you-go" cloud principle; no need for maintaining IT resources in the office; cloud services are easy to use because they are highly locally customized; management system is unified; safe online knowledge storage is provided; cloud facilities are easily accessible for every developer [3]. Moreover, software product delivery to end customers might be effectively done via internet from the cloud. So benefits of cloud computing can lead to improvement of the whole organizational development process and product quality. Though there are certain difficulties and requirements for the cloud usage in GSD, the advantages seem to be still bigger.

As a result of quite intensively growing interest in the area of GSD and cloud based development, many research groups are trying to develop real life working GSD format prototypes for the cloud. Last section of the paper gave an explanation of interesting examples of successfully implemented GSD cloud based prototypes, particularly Compile Server Farm, Online Storage Cloud and Lab Any Where [9]. It leads to the withdrawal that Cloud Computing approach has its wide potential application in Global Software Development. And nowadays it is on the continuous focus of the research in Software Engineering field.

REFERENCES

[1] Chou, D. C. and Chou, A. Y., Software as a Service (SaaS) as an outsourcing model: an economic analysis. *Proc. SWDSI'08*, 2007, pages 386-391.

[2] Clemons, E. and Chen, Y., Making the Decision to Contract for Cloud Services: Managing the Risk of an Extreme Form of IT Outsourcing. *Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS)*, 2011, pages 1-10.

[3] Cocco, L., Mannaro, K. and Concas, G., A Model for Global Software Development with Cloud Platforms. *Proceedings of 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, 2012, pages 446-452.

[4] Chauhan, M. A. and Babar, M. A., Cloud infrastructure for providing tools as a service: quality attributes and potential solutions. *Proceedings of the WICSA/ECSA 2012 Companion Volume*, 2012, pages 5-13.

[5] Hashmi, S. I., Clerc, V., Razavian, M., Manteli, C., Tamburri, D. A., and Lago, P., Nitto, E. D. and Richardson, I., Using the Cloud to Facilitate Global Software Development Challenges. *Proceedings of Sixth IEEE International Conference on Global Software Engineering Workshop (ICGSEW)*, 2011, pages 70-77.

[6] Huzita, E. H. M., Leal, G. C. L., Balancieri, R., Tait, T., Cardoza, E., Penteado, R. and Vivian, R. L., Knowledge and Contextual Information Management in Global Software Development: Challenges and Perspectives. *Proceedings of IEEE Seventh International Conference on Global Software Engineering Workshops (ICGSEW)*, 2012, pages 43-48.

[7] Martignoni, R., Global sourcing of software development-a review of tools and services. *Proceedings of Fourth IEEE International Conference on Global Software Engineering (ICGSE)*, 2009, pages 303-308.

[8] Mockus, A. and Herbsleb, J., Challenges of global software development. *Proceedings of Seventh International Software Metrics Symposium*, 2001, pages 182-184.

[9] Pavan, Y., Ramaseshan, R., Gayathri, B., Karthik, M., Divya, C., Global software development with cloud platforms. *Software Engineering Approaches for Offshore and Outsourced Development*, 2009, pages 81-95.

[10] Stankov, I. and Datsenka, R., Platform-as-a-Service as an Enabler for Global Software Development and Delivery. *Multikonferenz Wirtschaftsinformatik*, 2010, pages 555-566.

[11] Stapel, K., Knauss, E. and Schneider, K., Using FLOW to Improve Communication of Requirements in Globally Distributed Software Projects. *Proceedings of Collaboration and Intercultural Issues on Requirements: Communication, Understanding and Softskills (CIRCUS)*, 2009, pages 5-14.

TABLE 3. GSD CHALLENGES POTENTIALLY SUPPORTED BY CLOUD services [5]

GSD challenges and issues	Negative impact on software development	Facilitating GSD using Cloud's components
Geographic: distance, time, data transfer, infrastructure	Communication delays Project development delays Costs increases	Dynamic data exchange, always accessible project related knowledges, PaaS common shared development environment with all infrastructure, SaaS provided working software components
Cultural: lack of trust, face-to-face communication, unequal distribution of work	Poor management Work duplication Project delays Costs increases	TaaS and SaaS provide coordination tools, discussion tools, fair distribution of tasks, work monitoring tools, localized and translated services
Linguistic: knowledge transfer, frequency of synchronous communication	Ineffective project management Misunderstanding of tasks and problems' solution Release delays Loss in project quality	SaaS provides multilingual support, tasks' threads isolation, advanced communication stream possibility, project data history
Time zones differences at multi sites: less project visibility, duplication work risk	Project delays Loss in software quality Loss of some project knowledge	Cloud storage provides a guarantee of not losing data, information is always accessible

Cloud-based Testing: Opportunities and Challenges

Yanhe Liu
Department of Computer Science
University of Helsinki
Helsinki, Finland
yanhe.liu@cs.helsinki.fi

***Abstract*—Recent years, we are experiencing an explosion of cloud computing as a new generation of Internet service. More and more computing resources, software services and platforms are migrating to cloud, which also leads a great opportunity in providing more effective and scalable testing methods and tools as well as testing services (TaaS). However, testing has its own characteristics differing from other parts of software engineering, which requires new schema and techniques for offloading. This paper offers a clear overview of concepts, features, and challenges in cloud testing. Furthermore, this paper summarizes and reviews different products and solutions of cloud testing and testing environment as a service (TEaaS), and provides some possible research directions of cloud-based testing.**

***Keywords*—cloud computing, software testing, cloud tesing, test as a service**

I. Introduction

Recently, cloud computing has been one of the hottest topics in IT industry since it is changing the way of offering Internet service and computation resources. Today, we are experiencing an explosion of cloud computing as a new generation of Internet service. More and more companies, including some leading ones such as Google, IBM and Amazon, are offering service or infrastructure based on cloud techniques.

Software testing, as one of the most important but labor-intensive parts of software development, can also benefit from cloud computing. Since cloud computing provides on-demand resources and services with high elasticity and compatibility, research focusing on cloud testing, or sometimes called as Testing as a Service (TaaS), receives more and more attention recently. Cloud testing system is a new generation of software testing system. The software tests in this system are executed and analysed in the cloud-based environment.

One of the primary advantages of cloud testing is its cost-efficiency in managing and maintaining testing environment. Users can expand their testing resources on demand. For example, users do not need to buy multiple devices for large-scale reliability tests. They can run their tests in the cloud with devices provided by vendors. In addition, it is easier to simulate different hardware failures in a cloud environment. However, testing in the cloud is not free. For service vendors, testing in the cloud is executed in an online way where virtualization and dynamical configuration of testing resources are crucial, but difficult to implement. For the customers, safely and effectively migrating test cases or scripts to the cloud also requires special skills. Cloud testing brings challenges with opportunities and benefits together.

Although some organizations and companies have already provided cloud-based testing services such as cloud load testing and web application testing, cloud testing or TaaS is still a new topic for software engineering. When people talk about cloud testing, many points are still not very clear:

- What is cloud testing? What are the most important and distinct features of it?
- What are the major differences between software testing based on cloud and the conventional one not based on cloud?
- When should we use cloud testing? What are the major benefits of cloud testing? Is there any disadvantage of using it?
- What are the special issues and challenges of cloud testing?
- What are the typical structures of cloud testing system? What are the current practices?

In this paper, we present some basic concepts and features as well as distinct requirements of cloud testing and aim to answer these questions we list. To have more profound understanding of cloud testing, this paper introduces some practical cloud testing systems and compares them with legacy automated software testing systems. The rest part is organized as follows. Section II reviews some related research work. Section III discusses the concepts of cloud testing, including definition, distinct features and benefits. Section IV describes some typical testing systems based on cloud computing, and analyses the main differences between them and conventional software testing systems. Primary issues and challenges of cloud testing are discussed in Section V. Finally, we conclude in Section VI.

II. Existing Research Work

With the development of cloud computing, there are more and more published papers focusing on cloud-based software testing. Most of the literatures can be divided into two different groups: The first group mainly discusses basic concepts and standards of cloud testing and TaaS systems, and the second group concentrates on practical system design and implementation.

The papers [1], [2], [3], [4] propose some cloud testing systems. Hanawa et al. [3] develop a cloud-based testing system with fault injection. Lian Yu et al. [2], [4] try to optimize TaaS systems by clustering similar test tasks and reuse the testing environment. Jerry Gao et al. [5] discuss major benefits and requirements of cloud testing. Parveen et al. [6] focus on the cost of migrating software testing to the cloud.

III. Understanding Cloud Testing

A. What is Cloud Testing?

Cloud testing refers to the techniques of executing and managing software testing on a cloud-based environment [5]. There are two basic forms of cloud-based software testing: testing in a cloud, and testing toward a cloud [1]. The main difference between them is whether the testing focuses on validation of the quality of a SaaS application in the cloud. Testing in a cloud highlights getting test resources from cloud to test different software application no matter whether it is a service based on the cloud, while testing toward a cloud mainly runs software testing to assure the quality of a SaaS application in the cloud.

Unlike conventional software testing, cloud testing has some unique testing features and requirements:

1) On-demand Service: This is one of the most important features and requirements for most of the cloud computing techniques. Cloud testing provides consumers with more flexibility of using computing resource. Users can choose exactly what and how much they need [1], and they only need to pay for what they use every time.

2) Cloud-based Testing Environment: All the computing resources and test beds as well as testing platforms are provided and allocated automatically based on cloud infrastructure. The test cases are executed in the cloud-based environment. The vendors of cloud testing need to manage the testing resource by virtualization.

3) Service Level Agreements (SLAs): The cloud testing services are provided to different users with diverse but well-defined service-level agreements [5]. These agreements are always designed by different vendors and considered as an aspect of the testing service, such as data confidentiality, system reliability and user privacy.

4) Multi-tenant: The cloud testing service is always provided to multiple consumers [7]. Different from conventional internal test systems, the cloud service is designed for sharing by many different users.

B. Benefits

There are some distinct advantages when cloud-based testing is used. First of all, the cloud-based testing can reduce costs. Customers of cloud testing use virtualized computing resources in the cloud and share testing infrastructure on demand without buying required hardware and software. In addition, vendors of cloud testing offer testing platforms (networks, operating systems and hardware) for different test tasks. Users can easily execute tests without paying much attention on configuration and installation of test beds. The test environment can be set up quickly.

In addition to the cost efficiency, cloud testing can provide better load scalability and testing performance. It is easy for cloud testing system to generate and produce scalable test load utilizing parallel and concurrent computing as required to test and evaluate the performance of large systems. Improving performance by using concurrent computing is also an important characteristic of cloud-based techniques.

Furthermore, it can be easier to set up the cloud testing platforms for very large distributed systems. The cloud testing systems provide interfaces for automatically environment construction, which helps users to set up and rebuild large-scale test platforms conveniently. By using techniques of virtualization, the configuration time for setting test beds for large systems could also be shorten.

IBM illustrated some major advantages when using their cloud computing system in small business division [5]:

- capital and licensing cost can be reduced more than 50% by using virtualized technique.
- Testing setup time is shorten.

C. When to Migrate Test to Cloud?

Before starting to use cloud testing, we first need to consider when it is more appropriate to use, or whether cloud testing is suitable for the task. Though the advantages of cloud testing are considerable, it is not suitable for all the conditions. Users need to compare the benefits and trade-offs between a local test environment with cloud system before testing.

The first thing users need to consider is the testing efficiency. Generally speaking, an internal test environment is smaller and easy to build up, and there is no extra cost for test uploading and management. This testing environment is very suitable for testing some small applications. In contrast, a cloud-based testing system may have better performance for testing large distributed system. It is also more suitable to simulate complex user load of the real world.

The test case independence is also an important aspect for choosing test environment. Cloud testing may shorten the executing time of a large number of test cases by concurrent testing. However, this is possible only if the test cases or test tasks uploaded to the cloud are independent from each other. It is more appropriate to use cloud testing for highly independent test cases [6].

In addition to the independence of test cases, the environment requirement affects the testing execution. Typically the cloud environment is hardware-standardized. The cloud testing vendors always only provide the most common hardware resources and software environment to the users. Applications requiring special devices or particular hardware architectures may be not appropriate for cloud testing. Complex test cases which depend on some specific tools and libraries are also not good candidates for TaaS. However, some Web application tests are easy and appropriate to migrate to the cloud nowadays because of their low environmental dependency on specific hardware or software.

Moreover, users need pay attention to privacy and safety. By now, privacy and safety are still open questions for cloud-based techniques, and customers have to consider these questions and their risks before testing.

Tauhida Parveen et al. [6] suggest some features to consider when migrating software test to the cloud. They also show that some applications such as the GNU Compiler Collection (GCC) with complex environmental dependencies are not very appropriate to test in the cloud.

IV. Practical Solutions of Cloud Testing System

In this section, we introduce some typical structures and schemes for implementing cloud testing. First, we demonstrate the basic workflow of cloud testing. Then we introduce a cloud testing system with fault injection. Finally we explain the cloud testing system developted by Lian Yu et al. [2] which focuses on task clustering.

A. Workflow

Sometimes cloud testing is also called as testing as a service (TaaS), which receives much attention recently because of its cost-efficiency and load scalability. The concept of TaaS was introduced by Tieto in Denmark in 2009 [5]. A typical workflow of TaaS is displayed in Figure 1.

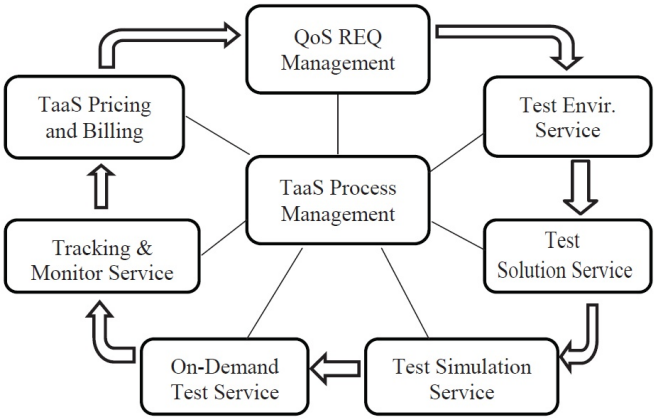


Fig. 1: General Workflow of TaaS [5]

The typical workflow of TaaS consists of the following steps and service modules:

1) TaaS Process Management: This is the main logical controller of TaaS systems. It provides process control and the management of test tasks.

2) QoS Requirement Management: This component manages the predefinition of software testing QoS requirements. There are modules which are used to assure test quality in this component.

3) Test Environment Service: This module is used for creating virtual testing environment by dynamically allocating cloud computing resources on demand. This part is very important for cloud-based techniques, which affects the efficiency of the whole testing system.

4) Test Solution Service: TaaS system can generate test cases and test suites automatically by using this module. In addition, it is responsible for scheduling the test tasks for the system.

5) Test Simulation Service: The module simulates testing inputs in the virtual system and use these simulations to reflect varieties of real conditions. For example, specific faults can be generated in the cloud testing system to simulate real fault scenarios. This component can not only simulate specific conditions, but also establish necessary environment with user data.

6) On-demand Test Service: By using this module, the system executes test cases for the customers on demand as the requirements and schedules.

7) Other Components: There can be some assistant modules in a TaaS system. For example, tracking and monitor module allows system to record the behaviors and results of the executed tests, and the pricing and billing module enables the customers to pay based on their testing workload.

B. D-Cloud

D-Cloud is developed by Toshihiro Hanawa et al. [3] mainly for testing parallel and distributed systems at first. It can be considered as a typical cloud testing system which accelerates software testing by executing test cases simultaneously using virtual computing system. D-Cloud consists of three different parts: virtual machine nodes providing computing resource, resource controller which manages and allocates testing resource for different tasks, and testing configuration controller, which interprets test scenarios to detailed hardware and software requirements. The test configuration of D-Cloud is written in a specific format of XML file to describe the test scenario and resource requirement, and the test cases can be provided in separated files written in scripting language. The structure of D-Cloud is shown in Figure 2.

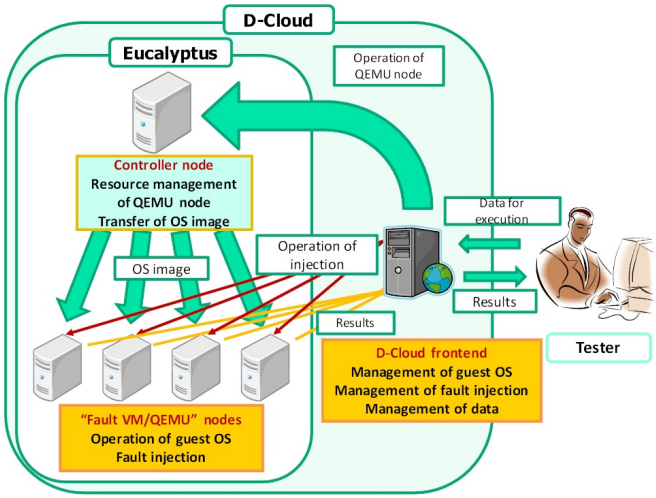


Fig. 2: Structure of D-Cloud [3]

D-Cloud constructs its virtual computing system by using QEMU (short for Quick EMUlator), which is an open-source software providing hardware virtualization. The on-demand software testing is executed in the QEMU virtual machines with specific emulated hardware and different operating systems. In addition, D-Cloud can emulate hardware fault by using QEMU.

In order to allocate testing resources dynamically and efficiently for performing a number of test cases simultaneously, D-Cloud uses Eucalyptus [8] to manage virtual machine resources flexibly. The system can transfer and allocate various operating system images to different virtual machine on demand as well as initial and close these virtual machines. The resource allocation is transparent for the test customers.

Before allocating testing resources dynamically, D-Cloud system first interprets user-defined testing configuration file to specified hardware and software requirements, and then uses these requirements for resource management. The configuration is written in XML and can be used to generate various resource descriptions. Table I lists some machine definition which can be considered as some basic hardware configuration of the testing platform. Table II is the descriptions of software environment in D-Cloud system.

TABLE I: Machine Definition Used in D-Cloud [3]

Element name	Meaning
machine	Delimiter for definition of the hardware environment
name	Name definition of the hardware environment
cpu	Number of CPUs
mem	Size of memory
nic	Number of NICs
id	ID of the used OS image

TABLE II: System Definition Used in D-Cloud [3]

Element name	Meaning
system	Delimiter for definition of the software environment
name	Name of the software environment
host	Delimiter of the testing host
hostname	Name of the host
machinename	Name of the used machine element
config	Designation of the configuration file

The basic workflow of D-Cloud is similar to what we describe in part A of section IV. The system first receives the test configuration and test scripts from the users, and parses the configuration file to allocate testing resources and set up testing environments. Then the test cases are executed concurrently. In addition, D-Cloud can simulate some kinds of system faults by using virtual machines.

C. Cloud Testing System with Task Clustering

As mentioned before, cloud testing system is cost-efficient for the customers because the customers can use virtual resources and testing environment as requested without really buying them. However, preparing separated but similar testing resources for several different users in a short time is still expensive for the system vendors. If the testing environment can be reused by several customers who have similar testing requests, the system can be more cost-efficient. Lian Yu et al. [2] develop an optimized cloud TaaS system with task clustering. The architecture of this system is shown in Figure 3. It contains five layers: test service frontend, test task manager, testing resource manager, testing layer and testing database.

1) Test Service Frontend: This is the interactive layer between users and the TaaS system where users can use different tools and methods to interact with the remote cloud testing system, give commands on what to test and transfer test cases as well as collect testing results.

2) Test Task Manager: In order to optimize the cloud testing system, Lian Yu et al. [2] develop a test task manager. This layer checks whether different test tasks can be clustered and executed together in one testing environment, and how these tasks can be scheduled in an appropriate order if they are clustered. This manager consists of several major components such as the module of checking test task capability and the module of clustering test tasks.

3) Testing Resource Manager: This layer is responsible for allocating test resources and creating test environment according to the requests processed by test task manager. It monitors hardware resources and manages all the virtual machines used for testing. This layer also contains some different modules, such as virtual machine controller and resource monitor.

4) Testing Layer: This layer is responsible for executing test cases on virtual machines.

5) Testing Database: This is the layer where test tasks, test cases, and testing result are stored.

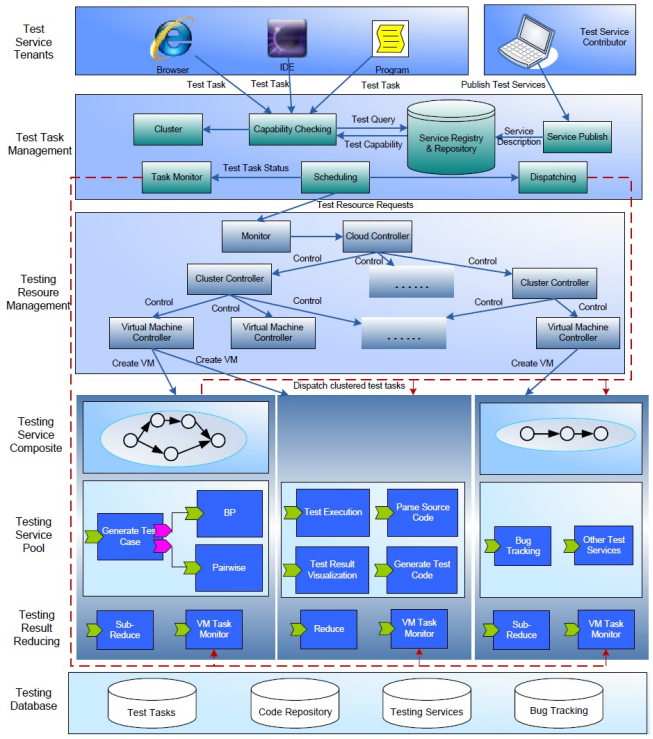


Fig. 3: Structure of Cloud Testing System with Task Clustering [2]

The test task manager is the unique component in this cloud testing system, which performs test task aggregation. Every test task defined by users has its test requirements on the executing environment and platform. The Table III is an example of the test requirements on software.

TABLE III: Software Requirements Used in Test Task Manager [2]

Element	Requirement
Operating System	Unix
Web Server	Tomcat
Database	MySQL
Compiling Tool	jdk

The test task manager first matches these tasks with specific environment which can fulfill the requirements by using the

definition given by the users, and then maps different tasks with similar requirements to one clustering platform. After clustering, the task manager make decisions on which sequence should be used for executing these different tasks. For handling this, the user should also give description on the priority of the tasks.

D. Main Features of Cloud Testing Systems

Though the two cloud testing systems we illustrated in our paper use different structures and have different characteristics, they have one basic feature in common which can also be considered as the distinct characteristic of TaaS systems: Both of them have a resource manager for dynamically allocating testing resources for multiple users simultaneously as requested. Different from conventional automated testing system, TaaS or cloud testing system should offer testing environment and platforms on demand. A cloud testing system can be considered as the combination of PaaS and SaaS: it provides not only the testing resources and environment as the platform, but the automated testing as a service. Some research only focus on the platform part, and they call this kind of service as test environment as a service (TEaaS).

In the Table IV, we list some famous cloud-based testing providers.

TABLE IV: Examples of Cloud-based Testing Providers

Cloud Provider	Testing Services	Features	Service Costs
Soasta	Performance testing of web and mobile application	1. Open for some cloud platforms and infrastructures like Amazon EC2 2. Support Mobile Touch test	Pay as you test
BlazeMeter	Performance testing of web applicaiton	Quick start	Pay as you test
Zephyr	Scalable platform for manageing testing cycle	Integrated with the JIRA platform	Buy the software licence

V. Challenges and Needs

With the development of cloud computing and network techniques, more and more companies start to offer their testing systems based on the cloud. However, there are still some challenges which hinder the utilization of cloud testing. In this section we first discuss these challenges, and then propose some needs and requirements for the cloud testing.

A. Challenges in Cloud Testing

1) Tesing Security: Security is one major concern in all kinds of cloud systems, including current SaaS, PaaS and cloud testing system. How to upload data to the cloud and use the remote computing resources safely and privately has already become a hot topic in the field of software engineering. For cloud testing system, designers need to pay more attention to the test data management, which is one of the biggest and critical open questions in TaaS [9]. Some issues which engineers have to handle are listed here:

- Is it safe for the users to test applications in a third-party cloud system? How can we assure that the testing and migrating processes are safe to use?
- What should the vendors of cloud testing do to protect and assure users' privacy in the cloud? What kind of mechanism are needed?
- What should the vendor do after the testing is finished? Do they need to delete all the test data and the software applications?
- What are the security standards for cloud testing?

The source code of test cases and applications can be confidential by either executing the test in some private clusters or providing specific and strong protections for the customers.

2) Environment Construction: One of the biggest differences between conventional testing systems and cloud testing systems is on-demand environment construction. The cloud testing system should allocate testing resources and build up the testbed automatically as required. Though current TaaS systems we mentioned in Section IV have the ability to construct the testing environment on demand, they can not guarantee high cost-efficiency for the system. To implement setting up test environment on demand in a cost-effective way, the cloud testing system needs to understand more about test tasks with specific techniques. Some PaaS techniques can be used here for enhancing the efficiency of construction.

3) Integration and Interoperability: Nowadays, many companies provide their own cloud testing systems. However, there is no standard on interaction interfaces or system structures. Different companies use different user interfaces and techniques to initialize the cloud testing, and engineers and customs must deal with the interoperation of different applications in the cloud based on different APIs. Extra costs are needed if users want to migrate their tests to other different cloud testing systems. There is a lack of well-defined interfaces and standards for interactive protocols between different cloud systems or vendors.

4) Scalability: The cloud testing system could be more cost-effective when concurrent and parallel computing techniques are applied. However, the first thing to use concurrent computing for cloud testing is to find appropriate ways to scale automated test systems [10]. When the system is very large, new challenges arise. The TaaS systems we introduce in Section IV can execute automated tests on demand, but if there are thousands of computers to manage and a large number of users together, the resource allocation and test monitor will become difficult. New techniques are needed for management the testing resources dynamically in a large scale.

B. Needs in Cloud Testing

To use cloud testing effectively, some requirements of optimization are needed in current TaaS systems.

First of all, the cloud testing systems should have some standard service level agreements. Recently, the cloud testing and SaaS systems still use different standards and provide diverse types of service level agreements (SLAs), which is inconvenient for customers and bad for interoperation over different clouds. Defining some standards for all the cloud testing vendors is beneficial not only for the customers, but also for the vendors themselves. The customers can design their test cases based on more detailed standards, and the vendors

can pay more attention on performance and reliability of their system. The specific standards could include the following aspects:

- What kind of security and privacy should a commercial TaaS product offer?
- How should the cloud testing systems deal with the uploaded applications and test cases?
- What is the minimal security requirement for transmitting test cases from users to the cloud systems? What kind of security protocols are needed?

For enhancing software testing integration in TaaS systems, innovative cloud testing models are also needed to provide more inter-operational APIs. For example, there is no uniformed protocol or API for communication crossing clouds. What we need right now are some integration models supporting crossing operations.

In addition, new large scale test platforms will be more important in the near future. With the development of mobile applications and high speed networks, the scale of Internet services is expanding. For testing this kind of services, a test system supporting large scale test loads and dynamical resource allocation becomes very necessary.

Moreover, we need more diverse testing environments. For minimizing the cost and simplifying the resource management, vendors of cloud testing always provide standardized hardware and software resources. However, sometimes uses need some special devices or particular hardware architectures. For example, the developers of applications of mobile phones may wish to test their applications on several different devices to guarantee compatibility, and the testing cost can be reduced if many different developers share to use the same remote devices together. This can be a new opportunity for cloud testing or other cloud platform companies.

VI. Conclusion

Cloud computing is a popular research field which not only brings opportunities to software testing, but also raises challenges. With the development of networks and software engineering, more researchers and companies start to test software in the cloud. However, cloud testing has no clear meaning or definition although it has been proposed for several years. In this paper, we presents some major features and benefits as well as the trade-offs and challenges of cloud software testing aiming to obtain better understanding about it. In addition, we study two types of practical schemes for cloud testing and analyse their major characteristics and structures. Moreover, this paper also discuss some basic requirements of effective cloud testing systems.

We hope that this paper can answer some basic questions about the nature of cloud testing systems, and provide some hints for efficient cloud testing and TaaS system design.

References

[1] J. Wu, C. Wang, Y. Liu, and L. Zhang, "Agaric – a hybrid cloud based testing platform," in *Proceedings of the 2011 International Conference on Cloud and Service Computing*, ser. CSC '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 87–94.

[2] L. Yu, W.-T. Tsai, X. Chen, L. Liu, Y. Zhao, L. Tang, and W. Zhao, "Testing as a service over cloud," in *Proceedings of the 2010 Fifth IEEE International Symposium on Service Oriented System Engineering*, ser. SOSE '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 181–188.

[3] T. Hanawa, T. Banzai, H. Koizumi, R. Kanbayashi, T. Imada, and M. Sato, "Large-scale software testing environment using cloud computing technology for dependable parallel and distributed systems," in *Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops*, ser. ICSTW '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 428–433.

[4] L. Yu, X. Li, and Z. Li, "Testing tasks management in testing cloud environment," in *Proceedings of the 2011 IEEE 35th Annual Computer Software and Applications Conference*, ser. COMPSAC '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 76–85.

[5] X. B. Jerry Gao and W.-T. Tsai, "Cloud testing - issues, challenges, needs and practice," *Software engineering: an international Journal (SeiJ)*, vol. 1, no. 1, p. 923, 2011.

[6] T. Parveen and S. Tilley, "When to migrate software testing to the cloud?" in *Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops*, ser. ICSTW '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 424–427.

[7] K. Incki, I. Ari, and H. Sozer, "A survey of software testing in the cloud," in *Proceedings of the 2012 IEEE Sixth International Conference on Software Security and Reliability Companion*, ser. SERE-C '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 18–23.

[8] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 124–131.

[9] L. Riungu-Kalliosaari, O. Taipale, and K. Smolander, "Testing in the cloud: Exploring the practice," *IEEE Softw.*, vol. 29, no. 2, pp. 46–51, Mar. 2012.

[10] G. Candea, S. Bucur, and C. Zamfir, "Automated software testing as a service," in *Proceedings of the 1st ACM symposium on Cloud computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 155–160.

Continuous Deployment of Software

Ville Pulkkinen
Department of Computer Science, University of Helsinki
Helsinki, Finland
ville.p.pulkkinen@helsinki.fi

***Abstract*—Continuous integration practice has gained popularity in the industry in the last decade. In the continuous integration practice all changes the developer commits to VCS are automatically and continuously tested for integration problems. Some have taken this practice even further to include also automated acceptance testing. The most extreme practice is to automate the whole process so that the deployment to the production environment is done automatically if tests pass. This is called the continuous deployment.**

Goal of this article is to i) describe the current state-of-practice and to ii) find out what is the current status of the continuous deployment in the software engineering research. I also will iii) point out issues that continuous deployment introduces to the software development. Lastly I briefly iiii) present how do the current cloud-providers support the continuous deployment strategy.

The concept of the continuous deployment as well as difference to the contionuous delivery and the continuous integration strategy is provided. Some views of the continuous deployment strategy from the industry will be presented. Lastly we will have short review of how current cloud-providers are supporting the continuous deployment strategy.

***Keywords*-D.2 Software Engineering, D.2.0.c Software engineering for Internet projects, D.2.6. Programming Environments/Construction Tools D.2.6.e Programmer workbench, D.2.16 Configuration Management D.2.16.f Software release management and delivery**

I. Introduction

The Internet, network infrastructure and cabable browsers have enabled cloud software to be a very popular approach to provide different kinds of services to users. At the same time the popularity of Lean software development approach and similar Agile methods put great emphasis for customer collaboration and faster feedback [1].

A Software that is provided via web browser, such as the Facebook or the Google Docs, allows deployment of a new software build to users without requiring any significant effort from the user. Page refresh might be enough to get the latest version available to use and the user doesn't probably even notice of the upgrade made between the normal interaction. Also these web-based solutions enables the collection of usage data continuously and in real-time. This data could be used to observe and to analyze the performance of the system or as an input for product development to support decision making for next steps in development project. This way developers can concentrate to the implementation of features that are important for users and hence reduce waste by not further implementing the features that are not used. Reducing waste is one of the principles of Lean software development. Also other principles of Lean put great emphasis on the fast customer feedback. Hence there is a vision of developing a software by a help of fast customer feedback loops by experimenting. With this approach it could be faster to find out what are the real needs that customers have for the software. The Continuous deployment is suggested to be the enabling technique towards this approach [2].

Behind all of these thoughts is one goal: to do more successful software development. To do successful software development, you must provide valuable software to the customer and getting the value to the customer as fast as possible without compromising the quality.

II. Background

The idea behind the continuous deployment has evolved from the continuous integration which was first presented by Martin Fowler in the year 2000 [3]. The concept of the continuous delivery and the continuous deployment is described by Humble et al [4] and the distinction is explained in more detail in blog post made by Humble [5]. Also one example of the continuous delivery in the cloud is provided by Birkner [6].

A. Terminology

Continuous integration is one of the practices in the agile software development where every team member check in to a centralized repository their work every time a new change or a task is completed [7]. After the developer has checked in the latest changes, a continuous integration system tries to compile the source code artifacts. If the build process is successful and a new build is available, the continuous integration system runs unit and integration tests. By *a build* as an object we mean an executable artifact compiled from a source code. If some tests fail the continuous integration system will inform, for example via email, the developer immediately which tests have failed. The idea behind continuous integration is to detect the problems that occur at integration test phase as soon as possible and to deliver the feedback as fast as possible to the developer.

Continuous delivery is about making sure that the software that is under development is always production ready -

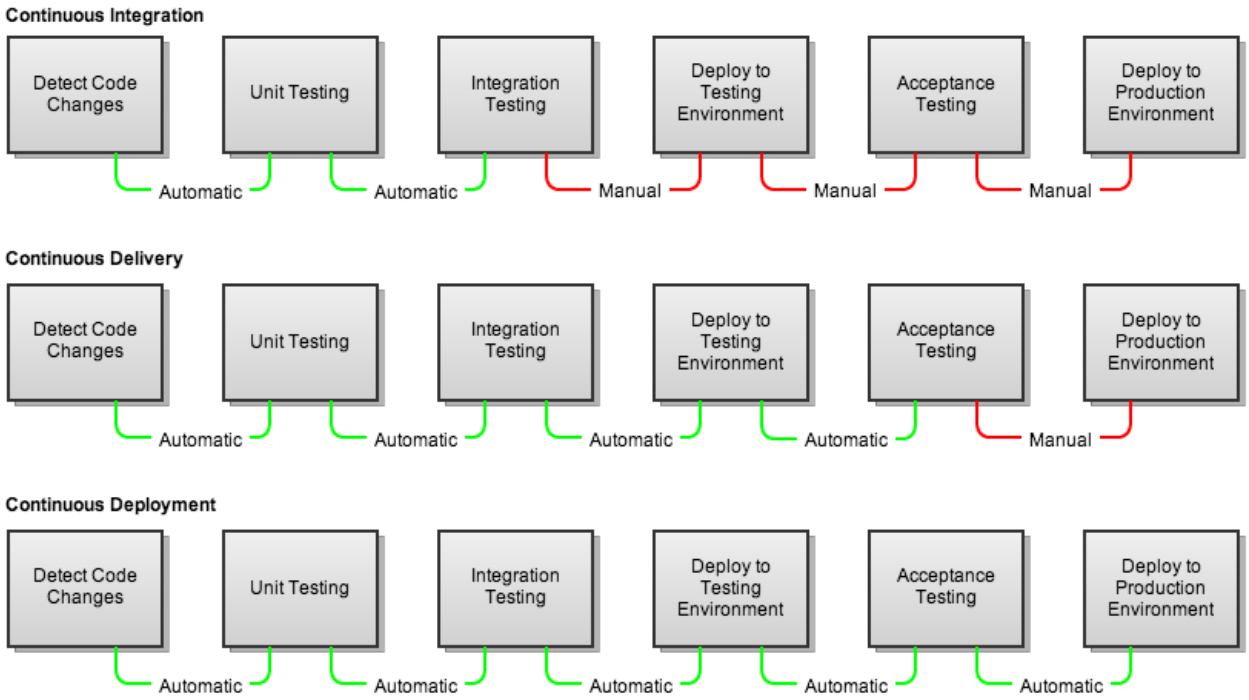


Figure 1. Continuous Integration, Delivery and Deployment

hence it is always ready to be deployed to the production environment [5]. The Continuous delivery consists of the continuous integration phase including also automated deployment to testing environments for automated acceptance and performance testing [4]. Releasing a new software build is in hand of a business personnel, not solely a developer. With continuous delivery there is possibility to do for example some manual exploratory testing or user assurance testing before the application is deployed to the production environment. Also some other inspections can be done. For example related to some software quality testing standards.

In *continuous deployment* every change that passes your automated tests will be deployed automatically to the production [4] environment. In this kind of practice all test phases should be fully automated including acceptance tests that cover functional and non-functional requirements. It is clear that successful continuous deployment depends on automated tests that are very good quality. It is more radical approach than continuous delivery.

In Figure 1 there is presented the differences of these three techniques. A Box is describing the process to take place and the line between the boxes is describing if the step between the processes is automated or not. The Author got inspiration for the figure from a visualization from Sundman [8].

If you have constraints on compliance, then approvals are required for deploying a new build to the production environment [4]. In that case you could not deploy continuously to the production environment with the help of an automated build pipeline. The same goes with everything around the code that is required to be done manually before the deployment is made to the production environment and thus cannot be automated. For example the creation of product support documentation.

Release vs. deployment. In continuous deployment one could also see a distinction with *a release* of a software product and *deploying* some *build* of the software to the production environment [9]. Deploying a build in continuous deployment might occur multiple times a day, but a release might be more business oriented event. For example announcement of some set of new features when they are already deployed to the production environment, hence are already available.

In this paper the Author cites many references that are related to the continuous delivery because there are also discussion about the continuous deployment. The continuous deployment is seen in those contexts as the most extreme and advanced technique [10].

B. Tools

There are several tools that are quite essential to achieve the continuous delivery and deployment. Heart of all is the continuous integration server which orchestrates the

whole process of building code into executable software artifacts, testing those artifacts and deploying a new working build to different environments. The Continuous integration server communicates with the revision and source control management systems also known as version control system (VCS) to checkout the source code and to trigger a new build process. New build process is triggered when some commit is made to VCS. The Continuous integration server notices changes from the VCS usually by polling it frequently.

Testing is done with the help of automated test suites which are covering all phases from the unit testing to the quality assurance phase including functional and non-functional tests. Also the database changes should be managed and applied automatically with the help of database change management tools. Environments and infrastructures for the testing and also for the production environment could be configured and setted up automatically with configuration management systems, altough not always required. Last but not least the build activity is usually automated with build tools and dependecies to external libraries are managed with dependency management systems.

All these tools are involved in an automated build pipeline process. Examples of these kind of tools are in Table I.

III. Related Work

The concept of continuous integration is greatly explained by Abdul et al [11]. Main points of continuous integration practice is to reduce the manual effort in the software build process and to reduce integration problems that usually occurs when different parts of software from different developers are integrated rarely. Referenced article [3] in the paper by Abdul et al [11] originates to the year 2000 and was rewritten in the year 2006 [3].

Canizzo et al [12] reports of an experience of extending the continuous building of software with the help of a continuous integration server to include also performance and robustness testing. Although the automated build system and the automated robustness and performance test framework took some time at the beginning to setup, benefits on later resulted in better maintainability. It was also easier to convince the customer by showcasing these test results for non-functional requirements on demand. The main challenges with automation of robustness and performance testing were the set up of a reasonable testing environment and coping with the concurrency of the actual system within tests. As the Authors of the paper noted it is crucial that the actual success criteria are defined for automated testing to take place. Other benefits are also mentioned in the paper such as writing of more robust code.

As noted by Jiang et al [13] the regression testing phase might be a bottleneck towards the continuous integration and hence also to the continuous deployment. The Authors also mention that test prioritization is used to overcome this. On the other hand this could reduce the overall code coverage and hence might lower the quality of delivery. As a result from the conducted experiment, the paper suggests that randomized prioritization outperforms all of the other examined approaches.

Continuous SCRUM presented by Agarwal [14] is described by the Author to enable continuously deploying software to production. The model he presented describes a weekly scheduling for releasing of a software with a three parallel scrum teams, thus not continuously. Also where the continuous deployment is described by Humble [4] as a practice where developer is done with a task when the feature is deployed to the production environment, Agarwal described that *"as and when each developer completes an individual work-item they mark its status as completed (in the Tracker application) after performing their round of UT (unit-testing)."*.

While there should also be a process model that supports the continuous deployment, the Continuous SCRUM Agarwal presented could be only adopted to the continuous delivery strategy where the automated deployment to the production environment is ommited. That is because of the nature of SCRUM development process model which relies on scheduled releases.

IV. Research Method and Approach

The Author studied practices from the industry by reading the book Continuous Delivery by Jez Humble and Dave Farley and reading multiple articles, blog posts and other web articles from the Internet related to the continuous delivery and deployment. Also to study the current state-of-art of the continuous deployment, articles related to the continuous deployment, delivery and integration were searched from the IEEEXplore digital library and via the Google search.

Approach of this paper is a development of a software which is provided to users via browser and is deployed to a cloud environment such as PaaS. Issues related to the continuous deployment of software that is supposed to be installed to a users local machine is ommitted completely.

V. Continuous Deployment from practicioners

A. Why continuous deployment?

Assumed benefits of continuous deployment are widely presented in the blog posts and articles on the Internet and literature from practicioners. Advocates are motivated mainly by the idea of fast feedback from system and from users. The Author noticed that there is some differences of how the terms continuous deployment and continuous delivery were understood. It seems like the terms got mixed up at times and there were no clear understanding of the terms.

Automated build pipeline was one of the main benefits of the continuous deployment. To achieve the continuous

Type of tool or software	Explanation	Examples
Version Control System (VCS)	To manage and revision source code changes. Essential to establish an automated build pipeline.	Git, Mercurial, SVN, CVS
Continuous Integration Server	To automatically run integration tests when code changes are introduced via VCS. Orchestrate automated testing and deployment.	Jenkins, Hudson, Atlassian Bamboo, CruiseControl, Teamcity
Software Configuration Management	To automatically setup and configure environment and infrastructure for example testing and production purposes. Also provides support for scalability.	Puppet, Chef, Salt
Automated Test Suites	To run automated tests for unit, integration, acceptance testing phases	Junit, JMeter, Cucumber, Selenium, FitNesse
Database Change Management	To automatically apply database changes related to current build of software. Also for tracking changes and to apply rollbacks.	DbDeploy, Liquibase, FlyWay, ActiveRecord
Build Tool and Dependency Management System	To automatically build the software and manage dependencies to other libraries	Apache Ant+Ivy, Rake, Apache Maven, Gradle

Table I
ESSENTIAL TOOLS FOR CONTINUOUS DELIVERY AND DEPLOYMENT

deployment, a fully automated deployment process from a commit to a production environment, must be implemented. Automated process is repeatable and trackable. Repeatability and trackability will help to track the defects in the process and by that reducing the errors made by human [4].

Continuous deployment is said to shift **more responsibility to developer**. You as a developer know that if tests pass, the code will be at the production environment possibly in few minutes [15]. Also the same author states that the continuous deployment including the continuous integration will force developers to deploy features in small pieces. That will ensure the iterative development and getting the feedback for deployed features fast [15].

The Author of the book Continuous Delivery states that ”Intuitive objection to continuous deployment is that it is too risky” [4]. Also he continues that it is also known fact that more releases lead to lower risk in any release. So the continuous deployment might reduce the risks involved in one particular release.

There is mention also about a **dilemma in quality assurance**. Quality assurance (QA) is usually quite heavy process in mission critical software (army, spaceflight etc.) [9]. QA is not suitable or scalable for example start-ups because of the beforehand costs. The Continuous deployment reduces the overhead in QA with automation and shifting the testing also to users, hence reducing the costs. [9]

Timothy Fitz states that usual misconception with continuous deployment is that ”Continuous deployment is for small startups who dont care about quality.” [16]. In the same blog post the Author tries to convince that continuous deployment strategy will scale up from small startups to bigger organizations, you’ll just have to create multiple build-pipelines and make the software more modular for example with SOA-patterns.

Some counter arguments were also presented [9]: customers of mission critical software won’t accept new releases on a continuous basis and continuous deployment leads to lower quality software than software built in large batches. In the same article there were some hints of how to overcome these kind of assumptions by implementing the continuous deployment strategy gradually. For example at first release immediately only the changes that are assumably side effect free. Secondly try to separate the concept of marketing release from concept of engineering release.

B. Deployment strategies

There is always a risk involved when deploying changes to a production environment. A Production environment is always unique with its unique state. Even though the system is thoroughly tested by quality assurance team it will be highly likely that defects are found when the application is in the production environment. You can never say that tests have catch all defects. Next is presented few deployment strategies that are assumed to reduce the risk involved in deploying a software to a production environment.

1) Feature Flags: The idea behind Feature Flags is that you have toggles to turn some feature on or off. If some new feature is causing issues, you can just disable it with configuration file, from the back-end or through some API call [17]. This is relatively quite straight forward and can be implemented in the code level. It does not need any special tools or processes to be achieved.

2) Dark Launches: With Dark Launches [17] you deploy new features to the production environment, but they are not visible to anyone else than testers. Testers could be bunch of automated test processes or humans interacting with the system. After the testers have approved the performance of the new features, you could make the features visible to everyone. The reason why you would like to do this is, because in this way you could really test the performance in the real production environment. This strategy should only be considered for performance testing and all the other tests should have been already run in the automated build-pipeline.

There is also different kind of description for Dark Launches from the Author of the book Continuous Integration: Improving Software Quality and Reducing Risk [18]. Duvall describes Dark Launces as launching a new application or feature when it affects the least amount of users.

3) Canary releasing (or gradual roll-outs[9]): Canary releasing as presented in the book Continuous Delivery [4] is a deployment strategy where the release of a software is released first only to a small group of users. This could be done automatically by the help of release management system. After it has been determined that there are no problems with the new version, the release management system will automatically (or via manual step) release the latest build to all of the users.

4) Beta users: Beta users as a deployment strategy is similar to canary releasing, but with the except that with beta users you deploy new features only to the willing subset of users: the beta users [17]. If there is no defects, you may release the new features to other users as in canary releasing approach.

5) A/B test approach: Vincent also suggests that you could utilize A/B testing approach in your deploying strategy that could help you to decide which feature would perform better [17]. A/B testing is quite simple form of controlled testing where the selected user base is split into the two groups and for both of these groups you expose different variation of a feature [19].

6) Blue-green deployments: Blue-green deployments give you the ability to rollback quickly if something goes wrong with the latest production build [4]. With Blue-green deployments you have two different environments: one for the new production build and one with the current production build. At first you deploy the new build to the other environment and when everything is loaded up as it should be you switch all traffic to the environment with the new production build. Fast switching could be achieved with the help of a router. If something goes wrong you can just switch the traffic back to the latest working production build. This strategy also reduces the downtime of deployment. Notice that changes in database must be managed somehow at the event of rollback.

C. Other useful practices

Other practices that are useful and important with continuous deployment are collecting statistics and monitoring metrics [20]. With monitoring it is possible also to alert automatically of unusual behavior of the system performance or users. For example if there is no users at the system or some typical usage pattern is not successfully accomplished after the last deployment. Monitoring could span for example from various business metrics to performance of the running server.

D. Issues

In case if anything goes wrong, which should be assumed, there should be a plan to **rollback the changes** made with the last deployment. If we assume that the state of the production environment is *s1* before the deployment to the production environment and the state after the deployment is *s2*, then by *rollback* we mean the plan or process (that could and should be automated) how to get from state *s2* back to the state *s1*.

Rollback is quite simple to implement with a software where there is no data stored between the sessions. You just redeploy the last known working build. Things will be lot harder when there is some database migrations involved or some interactions to some other external systems [4]. For example if the defect that will cause the rollback to be made is detected after some new data from users is already stored to the database you will either lose that data when rollbacking or you could get the database in corrupted state. This is more general problem in software releasing and not exclusively for the continuous deployment.

When continuously deploying a new build to the production environment it is extremely important that deploys do not cause any reduction to the **availability**. One way to achieve this is always deploy to a new environment and switch all traffic to this new environment when the deployment process is finished. This is also a problem related to software releasing.

Time of building and testing might grow when the software builds up. The building and testing phases might become a bottleneck to achieve the continuous deployment. To overcome this the build and testing processes should leverage concurrency and parallelization.

VI. Continuous Deployement in Cloud

The continuous deployment might need quite amount of computing resources to run all automated tests. Usually when the software is getting new features the amount of tests will grow. To run tests as fast as possible to reduce the feedback loop to the developer, the tests could be run in parallel with multiple processors [5]. Also this is crucial for Lean principles point of view to reducing waste [21]. It would be good that the developer would get the feedback from automated tests as fast as possible and without the need to switch tasks between the commit and the possible feedback of some failed tests. This could happen if the developer moves to the next task because running the tests takes too long and if the tests fails, the developer needs to switch back to the previous task. Cloud services could provide this kind of scalable build, test and deploy infrastructure. These kind of platforms are usually called as *PaaS (Platform-as-a-Service)* solutions.

A. Heroku

Heroku is a PaaS-solution that supports applications written in Ruby, Node.js. Clojure, Java, Python and Scala-languages [22]. Heroku provides also database, caching and other supporting software via add-ons.

As the code is commited to the Heroku-platform via Git, the Heroku automatically builds and deploys the new code to the production environment. One could easily create an continuous integration environment outside the Heroku-platform that commit changes to Heroku git-repository when the test-suite in continuous integration server has passed successfully.

Heroku itself provides the Tddium-addon [23] which is a Cloud-based continuous integration and deployment service. Tddium also states that it will automatically split up the test suite so to run the tests in parallel. This would provide great benefit for continuous integration, delivery and deployment because of the speed improvement in testing. It currently supports only Ruby, Jruby and Python languages.

B. CloudBees

CloudBees is also a PaaS-solution but it states that it provides a middleware services on top of IaaS [24]. They divide their offering in three different category: build, run and manage. The core of the CloudBees is Jenkins which is a popular Continuous Integration server [25]. The platform mainly supports JVM-based frameworks and languages. It is also possible to build and deploy JavaScript-based apps. Interesting aspect of CloudBees is that via its AnyCloud-solution it promises to support multiple choices as underlying deployment infrastructure. Currently the CloudBees website states that via AnyCloud it is possible use AWS (Amazon Web Service) or HP Cloud Services as underlying infrastructure. It is also possible to use own private cloud as an underlying infrastructure but it will need to support the CloudBees AnyCloud solution.

With Jenkins and CloudBees it is possible to run multiple builds and tests concurrently. There is no mention about automation related to running tests in parallel. CloudBees is promoting Continuous Cloud Delivery and Deployment a lot and there is a white paper about Continuous Cloud Delivery on their website available to download.

C. Red Hat OpenShift

OpenShift is the Red Hat's Cloud Computing Platform as a Service [26]. It provides built-in support for Java, PHP, Perl, Ruby, Node.js and Python. Also with OpenShifts customizable cartridge functionality it is possible to add the platform to support any language. Multiple popular frameworks for the languages mentioned above is also provided.

OpenShift also provides Jenkins as a build server. There is no mention about concurrency or parallel tests. Jenkins build is triggered after a git push and it deploys the build to production automatically if the build is successful. Running the build does not make the application unavailable. There is no support for rollback after the deployment phase takes place. Problems in the deployment process might cause failures as stated in the OpenShift website.

VII. Future Works

This article is about the continuous deployment and it is based mainly on the articles and blog posts written by practicioners from the software engineering industry. Some of the writers are advocating continuous deployment and at the same time are providing services to build this kind of continuous deployment systems. The concept needs some more research and experimental studies to understand the benefits more clear and more objectively.

It should be explored more that what are the real benefits of continuous deployment. Will it improve the quality of a software with learn fast, fail fast approach versus for example weekly releases? What are the benefits of continuous and automated deployment versus releasing the build manually? Will it make the development process faster as Continuous Integration in [27] versus doing it all manually? How much are the defect rates for a team that follows the continuous deployment strategy? How do the software development processes, for example Scrum, fit to the continuous deployment strategy? Is there some suitable existing process models or do the continuous deployment strategy need a process model of its own to structure the software development process?

Platforms like the CloudBees, that provides infrastructure of cloud platforms on top of IaaS and extends that with a tools and services which support continuous deployment, could be integrated to the IDE that works via browser as is proposed in [28]. With a system like that, enhanced with social and collaborative features, it might enable even faster development process in distributed environment and provide a whole new way to develop a software.

VIII. Conclusion

Continuos Deployment is quite unknown term in the scientific publications of Software Engineering field. There are some articles that defines continuous integration and some experiences that shows the advantages of continuous integration which is the base for continuous delivery and continuous deployment.

As the Author in the book Continuous Delivery[5] states even if it is not possible to apply the continuous deployment as your software development strategy, you should build your build-pipeline as such as you could switch to continuously deploying every commit to production at any time. That, as the Author states, is because continuous deployment strategy forces your team to good software development practices and build automation where any process is clearly defined because of the automation.

The cloud-based continuous integration, delivery and deployment solutions provides good alternatives to a traditional

local continuous integration server option with the ability to scale up when the builds and tests are being processed. Without the ability to scale up the continuous deployment would be impossible when the amount of running tasks are increasing.

REFERENCES

[1] M. Poppendieck and M. Cusumano, "Lean software development: A tutorial," *Software, IEEE*, vol. 29, no. 5, pp. 26–32, 2012.

[2] H. Olsson, H. Alahyari, and J. Bosch, "Climbing the stairway to heaven – a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software," in *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, 2012, pp. 392–399.

[3] M. Fowler. Continuous integration. [Online]. Available: http://martinfowler.com/articles/continuousIntegration.html

[4] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, 1st ed. Addison-Wesley Professional, 2010.

[5] J. Humble. Continuous delivery vs. continuous deployment. [Online]. Available: http://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment/

[6] M. Birkner. Continuous delivery in the cloud - part 1: Overview. [Online]. Available: http://blog.codecentric.de/en/2012/04/continuous-delivery-in-the-cloud-part1-overview/

[7] J. Abrantes and G. Travassos, "Common agile practices in software processes," in *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, Sept., pp. 355–358.

[8] Y. Sundman. Continuous delivery vs. continuous deployment. [Online]. Available: http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment

[9] E. Ries. Continuous deployment for mission-critical applications. [Online]. Available: http://www.startuplessonslearned.com/2009/12/continuous-deployment-for-mission.html

[10] P. B. Andreas Rehn, Tobias Palmborg. The continuous delivery maturity motel. [Online]. Available: http://www.infoq.com/articles/Continuous-Delivery-Maturity-Model/

[11] F. Abdul and M. Fhang, "Implementing continuous integration towards rapid application development," in *Innovation Management and Technology Research (ICIMTR), 2012 International Conference on*, 2012, pp. 118–123.

[12] F. Cannizzo, R. Clutton, and R. Ramesh, "Pushing the boundaries of testing and continuous integration," in *Agile, 2008. AGILE '08. Conference*, 2008, pp. 501–505.

[13] B. Jiang, Z. Zhang, T. H. Tse, and T. Chen, "How well do test case prioritization techniques support statistical fault localization," in *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*, vol. 1, 2009, pp. 99–106.

[14] P. Agarwal, "Continuous scrum: agile management of saas products," in *Proceedings of the 4th India Software Engineering Conference*, ser. ISEC '11. New York, NY, USA: ACM, 2011, pp. 51–60. [Online]. Available: http://doi.acm.org/10.1145/1953355.1953362

[15] N. Middleton. Continuous deployment with heroku. [Online]. Available: http://neilmiddleton.com/continuous-deployment-with-heroku/

[16] T. Fitz. Scaling up continuous deployment. [Online]. Available: http://timothyfitz.com/2012/12/03/scaling-up-continuous-deployment/

[17] J. E. Vincent. Deploy all the things. [Online]. Available: http://blog.lusis.org/blog/2011/10/18/deploy-all-the-things/

[18] P. M. Duvall. Continuous delivery patterns. [Online]. Available: http://refcardz.dzone.com/refcardz/continuous-delivery-patterns

[19] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: survey and practical guide," *Data Min. Knowl. Discov.*, vol. 18, no. 1, pp. 140–181, Feb. 2009. [Online]. Available: http://dx.doi.org/10.1007/s10618-008-0114-1

[20] E. Ries. Case study: Continuous deployment makes releases non-events. [Online]. Available: http://www.startuplessonslearned.com/2010/01/case-study-continuous-deployment-makes.html

[21] M. Ikonen, P. Kettunen, N. Oza, and P. Abrahamsson, "Exploring the sources of waste in kanban software development projects," in *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*, 2010, pp. 376–381.

[22] Heroku Inc. Heroku - cloud application platform. [Online]. Available: https://www.heroku.com

[23] ——. Tddium - continuous integration and deployment in the cloud. [Online]. Available: https://addons.heroku.com/tddium

[24] CloudBees Inc. Cloudbees.com. [Online]. Available: http://www.cloudbees.com/

[25] Wikipedia the free encyclopedia. Cloudbees. [Online]. Available: http://en.wikipedia.org/wiki/CloudBees

[26] Red Hat Inc. Build with jenkins. [Online]. Available: https://www.openshift.com/jenkins

[27] A. Miller, "A hundred days of continuous integration," in *Agile, 2008. AGILE '08. Conference*, 2008, pp. 289–293.

[28] T. Mikkonen and A. Nieminen, "Elements for a cloud-based development environment: online collaboration, revision control, and continuous integration," in *Proceedings of the WICSA/ECSA 2012 Companion Volume*, ser. WICSA/ECSA '12. New York, NY, USA: ACM, 2012, pp. 14–20. [Online]. Available: http://doi.acm.org/10.1145/2361999.2362003

Open source cloud platforms

Jussi Hynninen (Author)
Department of Computer Science
University of Helsinki
Helsinki, Finland
Email: jussi.hynninen@helsinki.fi

***Abstract*—During the last few years, the cloud computing paradigm has increasingly gained popularity as a means of providing scalable, on-demand computational resources. The basis for a cloud service is a software stack that provides consumers with an interface to the service while hiding the underlying complexity of the system.**

While many commercial cloud services rely on proprietary software stacks, a number of open source software solutions have emerged and gained popularity among service providers. Although the end-user may not always care how the service is composed, a service provider offering a cloud service may benefit from building on open source.

In this paper, motivators and benefits for using open source cloud software stacks are analysed. Prominent software stacks are presented and compared. The purpose of this paper is to give the reader a good glance on the current proceedings on open source cloud software and to give insight into how it can benefit the service provider.

***Keywords*- open source; cloud computing; software; infrastucture as a service;**

I. INTRODUCTION

A. A definition of cloud computing

Cloud computing is a computing paradigm that can be described with the following characteristics [1]:

- **On-demand self-service** - Computing resources can be acquired from a service provider by consumers without human interaction
- **Broad network access** - Computing resources are accessed using a network and potentially using a wide variety of end-user devices
- **Resource pooling** - The service provider's computing resources are pooled to serve multiple consumers. The physical location of the resources need not be known by consumers.
- **Rapid elasticity** - The capabilities of the computing resources used by a consumer can be rapidly scaled up or down according to demand.
- **Measured service** - A cloud system automatically optimizes and controls resource use according to collected metrics. Generally, this means the consumer only pays for the resources used.

In a nutshell, a cloud service allows consumers to access scalable computing resources over a network in a pay-as-you-go manner, using a self-service interface.

Cloud service models can be divided in three basic categories [1]:

- ***Software as a service*** (SaaS)
- ***Platform as a service*** (PaaS)
- ***Infrastructure as a service*** (IaaS)

In the SaaS model, the end product offered for the consumer is an application running on a cloud infrastucture. The service can be accessed using e.g. a web browser or via an application programming interface (API). In the PaaS model, the consumer is provided with a development platform that can be utilized for creating and deploying users' own applications. In the IaaS model, the consumer is provided with server infrastructure the consumer can use as she would use any physical hardware. Very commonly these resources are provided by means of hardware virtualization [2].

This paper concentrates on software platforms intended for realizing IaaS services.

Furthermore, cloud deployment models can be divided in four categories [1]

- Private clouds - Cloud services intended solely for the use of a single organization
- Community clouds - Cloud services intended for the use of a community consisting of consumers from different organizations with shared intentions
- Public clouds - Cloud services that can be acquired by the general public
- Hybrid clouds - Clouds that make use of two or more of the aforementioned deployment models

B. A general architecture for IaaS services

Generally, a single IaaS platform consists of at least the following or similar functions provided by a cloud software stack or, in other words, a cloud operating system (OS) [3]:

- **Virtual machine manager** - Manages the life cycle of virtual machines (VM's) deployed on the hardware infrastructure, on top of the virtualization hypervisor [2]
- **Network manager** - Manages the deployment of private networks in the cloud infrastructure, provides connectivity to the Internet and provides isolation for private networks
- **Storage manager** - Provides storage services and virtual storage to the infrastructure in an elastic manner

- **Image manager** - Manages users' VM images and provides a way to create, delete and copy images
- **Information manager** - Monitors and collects measurement data from the cloud infrastructure that is essential for optimizing the usage of hardware resources
- **Authentication and authorization** - Provides users with a way to authenticate with the cloud infrastructure and assigns them appropriate user privileges
- **Accounting and auditing** - Monitors the users' usage of resources for billing purposes and keeps an activity log of the users' activities in the cloud
- **Scheduler** - Responsible for the optimal placement of VM's on the hardware resources and interacts with the VM manager to launch new VM's
- **Administrative tools** - Tools for cloud users and administrators to perform administrative tasks within the limits of their user privileges
- **Service manager** - A component for managing multitier services running on the infrastructure, potentially involving virtual networks and/or storage
- **Cloud interfaces** - Expose cloud OS functionalities to the consumer using an API

Some of these functions may be combined under a single software component as will be seen later. The software stack composed of these functions is illustrated in figure 1.

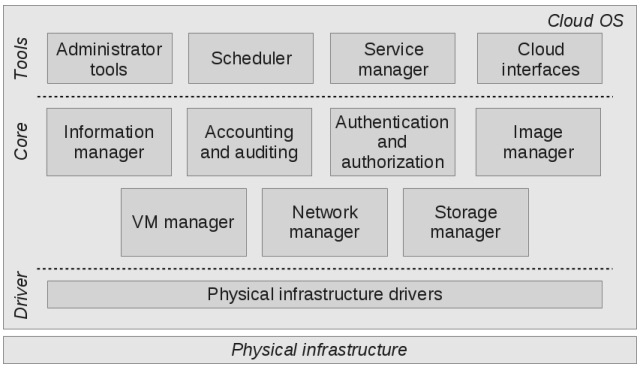


Figure 1. A generic IaaS architecture *(Adapted from [3])*

II. Open source software stacks in cloud computing - motivation

In this section, some key motivators for choosing to use open source software for building cloud infrastructures are discussed. This is done from the perspective of a service provider since the service provider gains most from open source [4]. There are consumer aspects as well, but those are out of the scope of this paper.

A. Economic benefits

When using open source software in building a service, the service provider does not need to pay for software licenses. In a potentially very large-scale installation like a cloud infrastructure, this is especially emphasized as cloud hardware may consist of thousands of hardware nodes each containing multiple processors.

An essential aspect of cloud computing are economies of scale [5] that allow a service provider to take the most out of its hardware. This requires large installations and - depending on licensing model - potentially increases costs very quickly as the number of licenses needed when using proprietary software can be very large. For instance, pricing for the proprietary VMware vCloud Suite [6] is tied to the number of processors which means that the price for a large-scale installation may be unbearable for the service provider.

It is obvious that software costs do not only include software licenses but also manpower and possibly installation and/or maintenance support are needed to realize a ready product. Without commercial support, more working hours are likely to be spent on installing and maintaining open source software. This, however, does not typically exceed the savings made from software licenses. Also, it is good to notice that commercial support may not be included in software license fees (as in [6]) - and on the other hand, commercial support for open source software is a growing business area as well.

Being able to cut down on software expenditure allows for lower customer prices and thus makes it possible reach a larger number of customers [4]. This results to more sales and - potentially - more profits.

B. Other benefits

By nature, open source software can be modified. This makes it possibly for a service provider to customize the software and provide bug fixes. Although this requires manpower, the costs can be offset if the result benefits the service provider [7]. Quite a few open source cloud OS's do have an active community built around them. This results to a few other things:

- It is likely that many organizations need a same new key feature or customization - thus feature requests are likely to be realized as well in an active community
- An active community is able to provide comprehensive support and answer questions quickly

Using a cloud OS to manage IT infrastructure is apt to protect the service provider from vendor lock-in on hardware level [3] and further, using an open source cloud OS helps both the consumer and the service provider to avoid it on software level.

III. An overview of prominent software stacks

In this section, four cloud software stacks are presented. A quick comparison of their features is given and a glance is taken on the current state of the community built around the software.

A. Eucalyptus

Eucalyptus is a cloud software platform allowing for building private or hybrid clouds. It has been developed by Eucalyptus Systems, Inc. [8] and available under the GPLv3 license.

Architecture: The architecture of Eucalyptus [9] is modular and consists of the following core components:

- **Cloud Controller (CLC)** - Entry point for the users and administrators and a high level scheduler. Collects information about resource usage from node controllers and implements high level scheduling decisions via cluster controllers. Provides an Amazon EC2 compatible REST/SOAP API as well as a web interface.
- **Walrus** - Implements an Amazon S3 compatible storage service via a REST/SOAP API that can be used to stage data in and out of the cloud. Also acts as a storage service for VM images.
- **Cluster Controller** - Schedules instance run requests from CLC to node controllers, controls virtual networks and gathers data from node controllers in the same cluster.
- **Storage Controller** - Provides block level storage to virtual machines
- **Node Controller** - Manages VM instances on a single cluster node

The Eucalyptus architecture is illustrated in Figure 2.

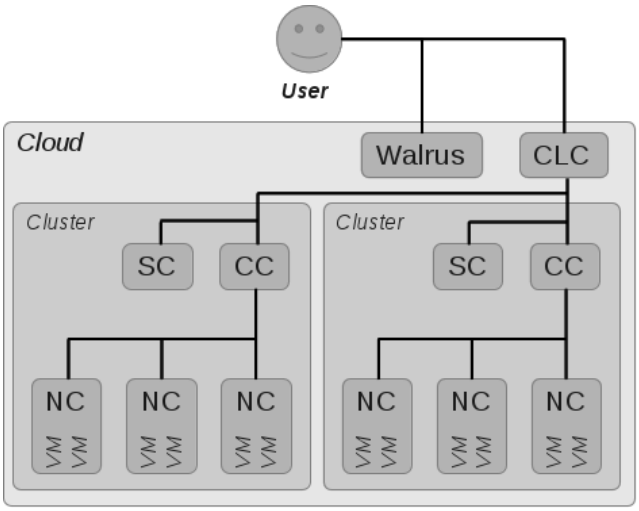


Figure 2. The Eucalyptus architecture

As can be seen, the basic idea of Eucalyptus is to combine multiple clusters to a hierarchical cloud infrastructure, having CLC as the central component and CC as the connection point for cluster nodes. On cluster level, Eucalyptus routes all network traffic through the CC, resulting to a limit of 750-800 running virtual machines per cluster. As many control operations are iterative, having over 200 node controllers in a cluster starts to deteriorate the performance on of Eucalyptus [10]. This makes Eucalyptus unsuitable for large scale installations.

Community: The Eucalyptus software stack is backed by a commercial vendor whose business model is based on consulting, support and training. Eucalyptus Systems also provide a hosted public cloud based on Eucalyptus. This is likely the reason for the role of the user community not being very actively emphasized on the Eucalyptus WWW site [8].

However, a user community exist and provides support for the product via a knowledge database, mailing lists and IRC. Meet-ups and events are organized and local user groups exist at least in the US.

The source code of Eucalyptus is available on GitHub [12] and there seems to be a steady flow of commits. However, the author was unable to find out the number of active contributors to the source code.

B. OpenNebula

OpenNebula is a open source cloud OS with wide adoption and support from both public and private sector [11].

Architecture: On a high level, OpenNebula consists of the following components [11] [15]:

- The OpenNebula core
- A scheduler
- Infrastructure drivers
- Cloud interfaces for consumers, administrators and services

Figure 3 provides a simplified view of the OpenNebula architecture. Cloud interfaces include EC2 [13] and OCCI [14] interfaces and OpenNebula's own command line interface (CLI). They communicate with the core software using OCA, a Java and Ruby abstraction layer for the XML-RPC [16] API.

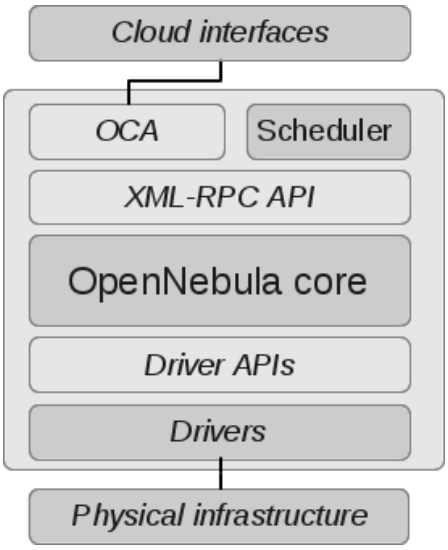


Figure 3. OpenNebula architecture simplified [11] [15]

The driver layer contains drivers for virtualization, storage, monitoring and authorization and connects to the underlying physical infrastructure.

Community: OpenNebula is a fully community-backed product. Support is available via mailing lists and IRC and events are regularly organized. In order to catalyze development and usage of the software stack, an ecosystem of tools, extensions, plugins and documents has been created. Via the ecosystem, users and developers can easily distribute their add-ons to OpenNebula. A marketplace for VM appliances exists as well.

The OpenNebula developer site [17] lists around 450 members. Source code of OpenNebula and project activity listings are available on the site as well.

C. CloudStack

CloudStack [18] is a cloud OS originally developed by Cloud.com and Citrix. It was made fully available under the Apache license in 2011 [19] and accepted as an Apache Incubator project in 2012. It provides Amazon EC2 and S3 APIs for compatibility. At least Datapipe [20] offers commercial cloud services based on CloudStack.

Architecture: Essential concepts in the CloudStack architecture are [21]:

- Computer Node (CN)
- Cluster
- Pod
- Availability zone

A CN is a single computer node running the CloudStack agent, along with a VMM, allowing scheduling of VM's. A Cluster consists of CN's sharing the same primary storage and having the same VMM installed. A pod is a collection of clusters and an avaliability zone consists of a collection of pods. In CloudStack terminology, an availability zone is the basic unit for an IaaS offering or, put in a simpler way, a single cloud. Figure 4 illustrates the CloudStack service layout.

A CloudStack management server can manage multiple zones and is the component that offers interfaces to consumers and administrators. It exposes its services through a web interface along with a REST API.

Community: CloudStack is backed by the Apache Foundation and anyone can participate to the project. Product support is available via mailing lists and IRC. Events and meet-ups are organized as well.

D. OpenStack

OpenStack [22] is a cloud software stack written in Python and intended for building IaaS services. It has recently gained a lot of momentum [23] and has support from ca. 150 organizations.

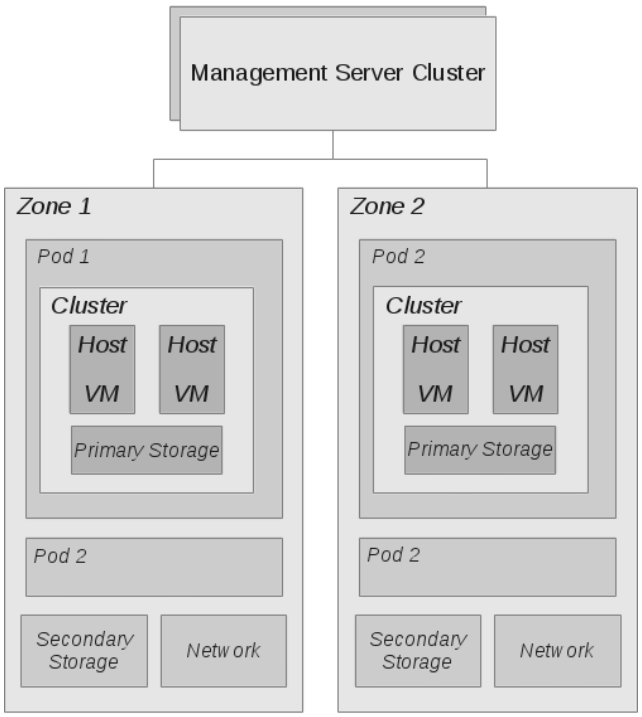


Figure 4. CloudStack architecture

Architecture: The OpenStack architecture consists of the following components:

- Keystone (Identity service)
- Glance (Image service)
- Nova (Compute)
- Cinder (Block storage)
- Swift (Object storage)
- Quantum (Networking services)
- Horizon (Dashboard)

Each component is written as a web service and offers a REST API through which the service can be accessed by clients. Figure 5 illustrates the OpenStack architecture in the Folsom release (27 September 2012).

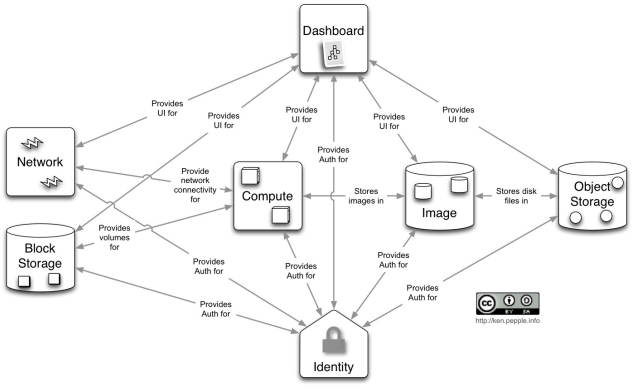


Figure 5. OpenStack architecture (Folsom release) [24]

Community: The development of OpenStack is steered by the OpenStack Foundation that has backing from 150 organizations. The community is active and there were over 500 contributors for latest release (4th April 2013).

Support is available via mailing lists, IRC and an online Q/A database. Events, such as meet-ups and conferences are frequently organized. Local user groups exist all over the globe.

Commercial offerings: A few commercial offerings based on OpenStack are available for consumers. For instance, RackSpace has commercial service offerings based on OpenStack [25]. IBM [26] currently has an OpenStack-based PaaS service in beta use and will be opening the service to a wider audience during the year 2013. In March 2013, the company Nebula introduced Nebula One [27], a complete IaaS solution that incorporates dedicated OpenStack controller nodes with racked worker nodes.

E. Comparison of features

Table I gives a quick glance on the essential features of the cloud OS's discussed. The set of features is very limited, but a reader should know the necessary sources to get more information having read this paper.

Table I
COMPARISON OF FEATURES

	Eucalyptus	OpenNebula
API compatibility	AWS (EC2, S3)	AWS
Community support	X	X
Commercial support	X	X
Scalability	Up to 800 VMs / 200 nodes per cluster	Tens of thousands of nodes
Hypervisor support	Xen, KVM, Vmware, (Bare metal)	QEMU/KVM, Xen, VMware
	CloudStack	OpenStack
API compatibility	AWS	AWS, OCCI
Community support	X	X
Commercial support	X	X
Scalability	Tens of thousands of nodes	Tens of thousands of nodes
Hypervisor support	QEMU/KVM, Xen, Vmware, Bare metal	Xen, KVM, QEMU, LXC, VMware, Hyper-V, Bare metal

API compatibility: Each of the given cloud OS's implement at least some API through which the cloud implemented by the software is accessible. Most popular are the Amazon Web Services (AWS) [13] compatible interfaces. Amazon Elastic Compute Cloud (EC2) is the computing service by Amazon Web Services, Inc. and Amazon Simple Storage Service (S3) an object storage service.

Open Cloud Computing Interface (OCCI) is an open set of specification of standards aimed to provide inter-cloud operability.

Available software support: Each of the presented software stacks have some kind of community support available. The business model for the company behind Eucalyptus is to offer commercial, high priority support for paying customers. Commercial support is also available for the other presented products.

Scalability: Aforementioned products (except for Eucalyptus) scale up to tens of thousands of nodes, enabling very large scale cloud installations. The issues of Eucalyptus have already been discussed above.

Hypervisor support: Xen, KVM, QEMU and LXC are open source virtualization products. Bare metal refers to running OS images on a computing node without utilizing virtualization at all - in this case a cloud OS is used to provision full hardware nodes. Rest of the products are proprietary, although ESXi (a stripped version of VMware's ESX hypervisor) is available free of charge.

A bare metal driver allows provisioning of bare compute nodes without utilizing virtualization at all. The approach has the advantage of having no computational overhead but also disadvantages, e.g. the lack of security due to direct customer access to hardware. An interesting feature of bare metal provisioning is that it allows system administrators to use a cloud software to manage non-customer hardware.

IV. CONCLUSION AND FUTURE WORK

In this paper, motivators for using open source cloud software stacks for IaaS cloud installations were presented. An overview of four cloud software stacks was given.

It can be concluded that open source is a compelling alternative even for commercial public clouds. Using an open source cloud OS, a commercial service provider may be able to lower infrastructure costs and to draw more paying customers due to the ability to do more compelling pricing.

The author was unable to find any real data that could have been used to calculate how much it is possible to save in software costs when building a cloud based on open source software. A case study on the issue would be of interest.

For research purposes and academia open source is an essential choice due to the fact that for proprietary software, source code typically is not available for modification or even for inspection.

For the presented cloud OS's, there are not many differences in the core features. Bigger differences can be found in the architectures and from this follow differences in scalability. Both commercial and community-based support is available for each of the given products so lack of support would be an invalid reason not to prefer open source cloud OS's.

REFERENCES

[1] P. Mell & T. Grance, *The NIST Definition of Cloud Computing*, September 2011 http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf *(Accessed on 15th April 2013)*

[2] M.F. Mergen, et al, *Virtualization for high-performance computing*, SIGOPS Oper. Syst. Rev., Volume 40 Number 2, April 2006, Pages 8-11, ACM New York, NY, USA

[3] R. Moreno-Vozmediano, R.S. Montero & I.M. Llorente, *IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures*, Computer, Volume 45 Number 12, 2012, Pages 65-72, IEEE

[4] D. Riehle, *The economic motivation of open source software: Stakeholder perspectives*, Computer, Volume 40 Number 4, 2007, Pages 25-32, IEEE

[5] M. Armbrust et al, *A view of cloud computing*, Communications of the ACM, Volume 53 Issue 4, April 2010 , Pages 50-58, ACM New York, NY, USA

[6] *VMware vCloud Suite Pricing*, http://www.vmware.com/products/datacenter-virtualization/vcloud-suite/pricing.html *(Accessed on 16th April 2013)*

[7] J. Lerner & J. Tirole, *Some simple economics of open source*, The journal of industrial economics, Volume 50 Number 2, 2002, Pages 197-234, Wiley Online Library

[8] *Eucalyptus*, http://www.eucalyptus.com/ *(Accessed on 15th April 2013)*

[9] D. Nurmi et al, *The eucalyptus open-source cloud-computing system*, Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009, Pages 124-131, IEEE

[10] R. Bradshaw & P.T. Zbiegiel,*Experiences with eucalyptus: deploying an open source cloud*, Proceedings of the 24th international conference on Large installation system administration, 2010, Pages 1-16, USENIX Association

[11] *OpenNebula*, http://www.opennebula.org/ *(Accessed on 17th April 2013)*

[12] *GitHub*, https://github.com/ *(Accessed on 21st May 2013*

[13] *Amazon Web Services*, https://aws.amazon.com/ *(Accessed on 21st May 2013*

[14] *Open Cloud Computing Interface*, http://occi-wg.org/ *(Accessed on 21st May 2013*

[15] B. Sotomayor et al, *Virtual infrastructure management in private and hybrid clouds*, Internet Computing, IEEE, Volume 13 Number 5, 2009, Pages 14-22, IEEE

[16] *XML-RPC Specification*, http://xmlrpc.scripting.com/spec.html *(Accessed on 22th May 2013*

[17] *OpenNebula Development website*, http://dev.opennebula.org/ *(Accessed on 21st May 2013*

[18] *CloudStack*, http://cloudstack.apache.org/ *(Accessed on 21st May 2013*

[19] *CloudStack Process Changes: Working the Apache Way*, http://buildacloud.org/blog/125-cloudstack-process-changes-working-the-apache-way.html *(Accessed on 15th April 2013)*

[20] *Datapipe, Inc.*, http://www.datapipe.com/ *(Accessed on 17th April 2013)*

[21] V.M. Muñoz et al, *The Integration of CloudStack and OCCI/OpenNebula with DIRAC*, Journal of Physics: Conference Series, Volume 396 Number 3, 2012, IOP Publishing

[22] *OpenStack*, http://www.openstack.org/ *(Accessed on 15th April 2013)*

[23] O. Sefraoui et al, *OpenStack: Toward an Open-source Solution for Cloud Computing* International Journal of Computer Applications, Volume 55 - No. 03, October 2012, Foundation of Computer Science, 244 5th Avenue, 1526, New York, NY 10001, USA India

[24] K. Pepple, *OpenStack Folsom Architecture*, http://ken.pepple.info/openstack/2012/09/25/openstack-folsom-architecture/ *(Accessed on 23th April 2013, licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.)*

[25] *Rackspace open cloud*, http://www.rackspace.co.uk/open *(Accessed on 15th April 2013)*

[26] *IBM To Make Its Cloud Services and Software Open Sourced-based*, http://www-03.ibm.com/press/us/en/pressrelease/40519.wss *(Accessed on 15th April 2013)*

[27] *Introducing Nebula One*, https://www.nebula.com/nebula-one *(Accessed on 15th April 2013)*

Secure data management for cloud-based storage solutions

Mikael Svenn
Department of Computer Science
University of Helsinki
Helsinki, Finland
mikael.svenn@cs.helsinki.fi

Abstract— **Organizations and individuals worldwide are evaluating and experimenting the possibilities of cloud-based computing. Organizations examine cloud computing as a simple and flexible model for outsourcing the management and maintenance of IT-infrastructure, whereas individuals experience cloud as a realm of services. As the cloud-based data storage services have evolved to meet the requirements of modern data management, they have gained wide interest from both organizations and individuals. However, the interest is restrained, because the basic concept of cloud-based storage services has not gained complete trust from either side. Several concerns have risen about storing data into a completely unknown grid. Organizations are impeding on migrating their data to storage services due to data security and availability related issues. According to studies, individual users similarly consider storage services to be suspicious when it comes to the privacy of their sensitive data. This paper addresses the common benefits and concerns related to cloud-based storage solutions and presents major enhancements to privacy, security, availability and trust of storage services.**

Keywords: cloud; secure storage; storage as a service; data security; encryption

I. INTRODUCTION

The development of computing resources resembles a wave-motion in many ways. It was found revolutionary when the first Personal Computers were introduced, triggering a transition from the era of large centralized mainframes to a realm of personally managed devices. The development of various mobile technologies has further blurred the limits of personal computation, introducing modern laptops, smart phones, tablets and embedded devices. As a result an end-user is often in possession of several different devices instead of a single personal computer. Simultaneously the connectivity of different devices has evolved in both bandwidth and connection mediums. Generally the model of connecting devices has moved from wired low-bandwidth peripheral- and network connections to wireless networks and high-speed Internet connections.

Due to the development of high-speed broadband and cellular data connections the modern service infrastructures are capable of integrating various devices over the Internet thus providing any kind of services or content to any kind of device base [3]. As a result the services are moving from individual and locally managed environments back to centralized service models, where both the service and the content is hosted by a dedicated service provider and often accessed over the Internet [11].

The development of virtualization technologies has allowed more efficient use of hardware resources, introducing concepts such as dynamic resource allocation, allowing the service infrastructure to scale on demand. Such new centralized service models have introduced a concept of disposable IT-resources, where any technical aspects may be provided as a service, whenever needed and regardless of whether the aspect is an entire IT-infrastructure, a platform or even a software performing a dedicated task [16]. The service model is generally considered an umbrella term known as cloud computing.

All cloud services share the idea of outsourcing a technical asset to a service provider, relieving the user from any concerns of maintaining the provided asset. The technical details such as implementation of the provided service are often not visible to the end-user and therefore usually masked behind a service client or interface through which the service is used. Cloud services are often not limited to any specific use-cases or clientele, being suitable for both individual consumers and organizations. However, the need for outsourced resources and therefore requirements and concerns related to the service may vary between organizations and individual consumers. Since the end-user has virtually no control over the technical details of the provided asset, several security related questions and issues have risen regarding the provided cloud services [1,16]. As a result, users often hesitate to utilize cloud computing for any sensitive operations such as storing or processing any sensitive data. Moreover several organizations are impeding the adoption of cloud services until the security and privacy related questions of the data processed in the cloud have been resolved [1].

This paper focuses on cloud-based storage solutions from the perspective of both individuals and organizations. The concept of cloud storage is introduced, accompanied with the most common the security and trust-related issues causing potential customers to avoid storing any sensitive data to the service, or to even impede the adoption of cloud-based storage services. The rest of the paper focuses on presenting requirements and some of the most promising alternatives for providing the required security and trust to cloud-based storage solutions. The end of the paper addresses difficulties and questions related to the suggested solutions.

II. CLOUD STORAGE

The requirements of data management have significantly changed during the past decades due to evolution of mobile devices, various different platforms and networked services [1]. Data management has become a complex task as the data is assumed to be accessible from multiple different services and devices, regardless of the physical location of the data or the services producing and consuming it [5]. Especially social networking and increasing interoperability of services have further highlighted the demand for sharing the data between various applications and services. High speed network connections enable services to produce and consume vast amounts of data, regardless of any physical aspects of the data storage, such as location or the device hosting the data. As a result, requirements such as high availability, accessibility and easy maintenance have grown to be the key factors of modern data management schemes.

One of the most common forms of cloud computing is storage as a service, providing a scalable and flexible storage resource for any kind of use [3]. Cloud storage services aim to meet the challenges of data management by centralizing the entire storage solution to the service provider, relieving the customer from any storage-related concerns such as data synchronization or backups. As the storage is provided as a service, cloud storage solutions are often considered to be highly available, reliable and scalable despite of the use-case of the service [3]. Individual users may use cloud storage for backing up or synchronizing the local content of any end-user devices, sharing data to other users or even for replacing solutions such as personal media servers requiring large storage capacity. Organizations may similarly utilize cloud storage for backup, distribution and even data versioning purposes [17,20].

The entire storage solution is managed by the service provider from physical level to application interfaces [3,16]. Access to the storage is often provided to the end-user either by a specific storage client, an Application Programming Interface (API) or both [22]. The storage clients usually integrate to the Operating System (OS), utilizing the built-in sharing and storing functionalities in mobile OSs or the shell functionalities in desktop operating systems. Many desktop clients mask all of the client functionalities to shell extensions or user-space File System (FS) drivers, providing a convenient access to the cloud storage [17]. The storage services usually provide both platform specific Software Development Kits (SDK) and Representational State Transfer (REST) based APIs [18,21,23].

The most common cloud storage services apply a pay-by-use payment model in which the customers subscribe for a certain features or certain amount of storage on a solid and recurring fee. The storage capacity and features can often be upgraded by simply subscribing for a larger storage or for more features. Cloud storage services are offered by several large providers such as Microsoft, Amazon, Google, Apple, IBM, Oracle and HP [3] as well as plethora of smaller providers such as Dropbox [17] and Just Cloud [19].

A. General Implementation

Even though the cloud storage services are generally proprietary solutions, each being a unique service with unique implementation details and API, some general design aspects are often shared by the actual storage service implementations. The physical storage implementation is highly distributed and often divided to multiple layers abstracting the low-level implementation. As large scale data storage is highly demanding for both storage devices and network resources, the storage services usually form an interconnected storage grid of vast amount of geographically distributed servers [3,15]. The individual servers are often referred to as nodes, located in clusters which are hosted in a datacenter. Storage services usually consist of several geographically distributed datacenters and thousands of individual storage nodes [23], as illustrated in figure 1.

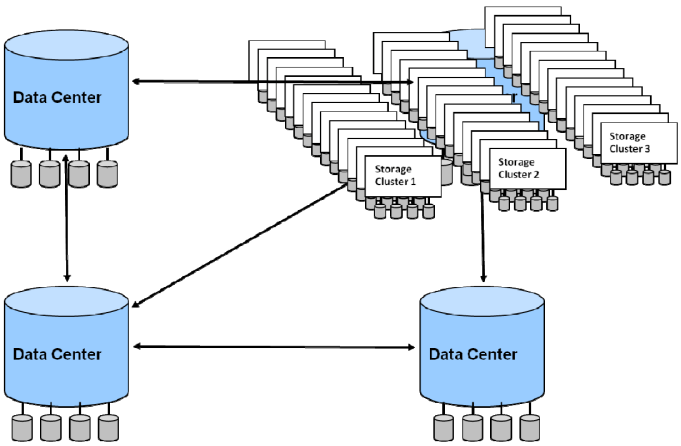


Figure 1. High-level overview of cloud storage structure [23]

The abstraction levels of a storage service can generally be divided to five categories, as shown in figure 2, each implementing a specific functionality of a large scale data management scheme. The highest level of abstraction is the service interface, providing an access gateway to the service. The service interface is primarily used for access control and interpreting client calls, thus utilized either by a client application or direct API calls unique to the service. All authorized calls are relayed to the storage overlay responsible of managing the underlying physical resources and abstracting the implementation for the service interface [3]. Storage resources are often virtualized and accessed through a distributed FS combining the geographically distributed nodes under one logical file system. For efficient storage management, each file stored in the service might be distributed to multiple different storage nodes and physical hard drives around different datacenters, depending on the utilization level of the data centers [22].

The low-level implementation is abstracted by a separate metadata-layer combining the service-wide information required for efficient storage management, such as locations of logical entities or current load and utilization of resources. The information is used to assist in service optimization such

as resource allocation, data dispersion and load balancing [3]. The logical entities are mapped to physical resources in the storage management layer, organizing the geographically distributed physical clusters to logical storage domains. Each cluster of physical storage nodes is managed in the network and storage infrastructure layer, exposing the physical storage resources to the storage management [3].

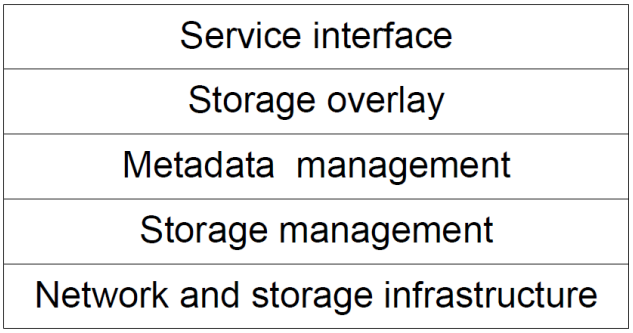


Figure 2. Abstraction levels of cloud storage service [3]

B. Resource Optimization

The overall storage management in the scale of cloud storage is a highly demanding task, requiring continuous optimization of the resources. Even though the details of storage allocation and load balancing implementations depend on the Storage Service Provider (SSP), some properties are common to most of the storage services. Generally the data can be considered to be in constant change as the storage optimization mechanisms re-allocate stored data between different nodes, clusters and even datacenters. Data reallocation might occur, depending on the load of the nodes, even in extremely dense cycles of only minutes. The data reallocation is masked so that it is not visible to the higher layers of abstraction and it should not cause any visible latencies to the end-user [22].

In order to provide redundancy and to increase the efficiency of networking and storage resources, Information Dispersal Algorithms (IDA), based on erasure coding algorithms such as Cauchy Reed-Solomon or RAID-6 Liberation Codes, are applied to the stored data [24]. IDAs shred the stored files to small chunks of data, and an erasure code is calculated and added to each of the shredded data chunks [24,25]. The small pieces of data are then stored to selected nodes in the service, specified by the utilized storage optimization and load balancing algorithms [24]. When the stored data is retrieved, only a subset of the stored data chunks needs to be read in order to reconstruct the original data, based on the previously calculated erasure codes [22,25]. In addition to providing efficiency in data operations, the IDAs also provide data redundancy in case of failing nodes [22]. The size of the data chunks is often determined so that only minimal amount of overhead would be caused by the information dispersal [25].

As users are expected to store sequences of data that might already be stored in the service, data deduplication, also known as single-instance storage, is applied to avoid duplicating any previously stored sequences of data [3]. Deduplication algorithms are very similar to compression algorithms, identifying any repeating occurrences of data sequences that might have already been stored to the service. Each repeating occurrence of already existing sequence of data is replaced with a reference to the existing data [3, 25]. In order to maximize the efficiency of the deduplication scheme, the operations do not consider any ownership of the referenced data, thus practically sharing sequences or even complete chunks of data between different users [25].

III. SECURITY AND PRIVACY CONCERNS

Regardless of the various benefits provided by cloud storage services, several privacy and security related questions have reduced and even impeded the adoptability of cloud storage. Generally both individual users and organizations are concerned about the safety of cloud storage, even though the service contracts and details might differ between the services offered for organizations and private consumers [1]. Consumer services and especially free storage service accounts often lack any promises regarding privacy, availability or quality of service, whereas corporate solutions may be negotiated with specific Service Level Agreements (SLA) aiming to improve the availability of the storage service [3]. Despite of any negotiated SLAs, reliability factors such as data integrity or privacy cannot be completely guaranteed. In order to share liability with the service provider, organizations and corporate customers often make special legal agreements against any security breaches or loss or corruption of data, whereas no promises or guarantees are made for private consumers [1].

Despite of having SLAs and legal agreements with the service providers, corporate security guidelines often advise against storing any sensitive data to cloud, as the integrity or safety of stored data cannot be completely guaranteed against security breaches or data corruption [1,2,4]. Standard privacy policies and Terms of Service (TOS) for consumer cloud storage services often take no liability of any corruption or loss of data and claim all rights to read, delete, modify and sell any stored data [1,9]. Consumer cloud storage providers might also claim rights to disable user accounts without a reason and to modify or even stop offering the service without any prior notice [1].

According to a study conducted by Ion et. al. attitudes and beliefs towards cloud storage services vary based on the background and country of the user [1]. The study reveals that private consumers consider Internet as a whole, as well as cloud storage to be less safe than local storage. Even though the assumptions and beliefs related to privacy of the stored data varies between private users, sensitive documents are most often preferred to be stored offline, as shown in figure 3. It also appears to be common that private consumers might not read the TOS, thus being unaware of the statements they have agreed to, believing to have more rights over the stored data than they actually do [1]. Private consumers also show willingness to invest in privacy similar to corporate agreements, should such options exist. In addition to of willing to buy security insurances for the stored data, the private users also show great interest in paying for the SSP not to sell the stored data [1].

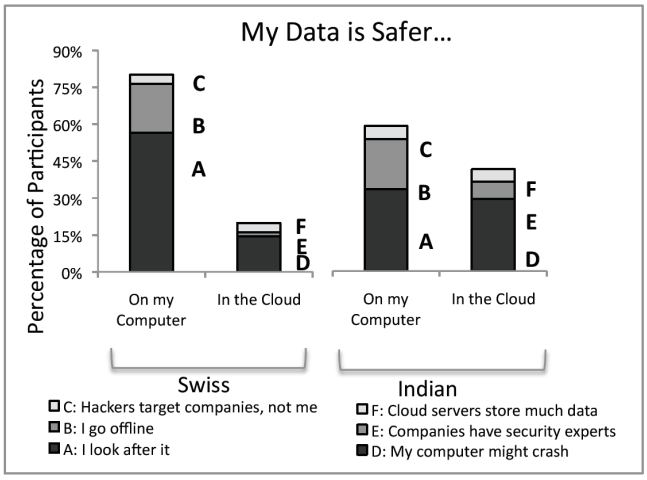


Figure 3. Private users' assumptions about the safety of the stored data [1]

A. Storage Security

In addition to concerns regarding agreements and legal aspects, several detailed issues and limitations have been identified in storage service implementations, reducing the adoptability of cloud storage for any sensitive or high-value data. The issues can be roughly categorized to four key topics addressing availability, integrity, privacy and portability of the stored data [4]. All of the identified issues and limitations concern both the private users and the organizations, regardless of any agreements made between the customer and the service provider.

The major concern is related to the privacy of the stored data [5]. In order to optimize the use of resources, the service providers often store the data unencrypted, rendering the data vulnerable to any attacks targeted at the service provider. As the stored data is constantly moved between different nodes, clusters and even geographically distributed datacenters, the risk of eavesdropping is further increased. Any weaknesses discovered in the access control implementation of the storage service would expose all of the stored data to any potential attackers.

To address the risks of eavesdropping or being hacked, some SSPs offer enhanced security by encrypting the content in prior to storing or moving it between different nodes. In order to avoid altering the service interfaces and to provide convenient user experience, the encryption in such solutions is completely managed by the SSP [22]. As the implementations of storage services are not transparent to the end-user, the implementation of the encryption cannot be verified or audited by the customer [2]. Since the end-user has no control over the encryption keys, the data is not protected against the SSP itself for any malicious insiders or situations where the SSP would arbitrarily access the stored data for example in selling purposes [1,2,4]. If the applied encryption would be implemented on low-level layers, such as storage management layer, the encryption would only protect the data against any attacks targeted to the physical nodes. If a weakness would be found from the access control mechanisms, an attacker utilizing the entire service layer stack could still access the data with the key management built in to the service layers.

Similar privacy-related issues have been identified regarding the deletion [1,14] and the locality of the stored data [1,22]. As the data is often stored in several chunks, deduplicated and constantly moved around the world, it is not clear where, how and in which form the data is being stored by the moment. The question might become extremely relevant when corporate confidential or especially government related data is moved to a datacenter belonging to the legislation of some other country or region [1]. As the SSPs might be obligated to allow a local governments to access any of the data stored within the boundaries of the specific datacenter, any classified data could be at the risk of exposure. Since the data cannot be guaranteed to reside within the borders of the country of origin, the stored data may be subject to varying privacy and data protection laws, depending on the location of the datacenter [22]. Similarly due to the common implementation of storage services, the deletion of data can neither be guaranteed. The actual data might still physically exist in the service for undefined amount of time after deletion [14]. Even if the data could technically be deleted instantly, some SSPs might be reluctant to enforce any deletion policies, preserving copies of the deleted data for mining purposes [14].

B. Storage Availability and Integrity

As the cloud storage represents a highly distributed storage system consisting of multiple layers of data management and mapping, the reliability of the service especially in maintaining data integrity has grown to be one of the concerns reducing the adoption of storage services [2]. Several occasions are known where the stored data has been outdated, corrupted or completely lost after being stored to the cloud [2,4,7]. The SSP might even employ storage optimization algorithms deleting files that are rarely accessed [9]. Similarly the data might be corrupt due to unauthorized modifications to the stored data [1,9,12]. Even though the storage systems are designed to be fault tolerant, the data might still become corrupt in the processes of dispersing, reallocating, reassembling or transferring the data [2,12]. Despite of the erasure coding, crucial amount of data might be stored to nodes failing simultaneously thus corrupting the entire data entity [2]. Similarly the data might be lost due to bad system configuration or an operator error [2].

Another open question is related to the availability of the stored data. As described earlier, failures in the storage system are likely and might occur on any level. In addition of being completely lost or corrupt, the stored data might also become outdated or temporarily unavailable as a result of data reallocation, failing nodes or even failures in general infrastructure, such as network and power distribution [7]. The data might be mostly stored within a cluster or a datacenter suffering from a temporary power outage. Similarly parts of the service or even the entire storage service provider might be subject to a Denial of Service (DOS) attack, either impairing parts of the service or completely disabling the service [4].

IV. INTEGRATED SOLUTIONS

Large organizations and corporate customers usually rely on legal agreements in order to enforce the security of cloud storage. Such legal agreements may be tailor made with the service provider, being highly expensive to the customer [1]. Depending on the internal security policies the organization may also choose to enforce the security by outsourcing the data storage only to certified service providers complying with industry security standards such as US Health Insurance Portability and Accountability Act (HIPAA) or Payment Card Industry Data Security Standard (PCI-DSS) [26,27,28]. However, complying with the pre-defined industry standards does not automatically address all of the previously stated security and availability issues, being inadequate for customers requiring zero-tolerance for information disclosure or data corruption. As being designed for specific clientele, the certified storage services might lack features desired by private consumers or they might be too expensive or too complex for the required use. Thus a general-purpose solution applicable for both private users and organizations, addressing the various security and availability issues is desired.

A. Third Party Auditors

One recognized approach is to introduce a trusted Third Party Auditor (TPA) to verify the correctness of the stored data [7,9,11]. Since the storage services are not transparent to the end-users, enforcing security policies in cloud storage is often impossible. For the same reason, verifying correctness of the stored data is traditionally considered to require downloading and verifying all of the data, repelling the overall benefits of the cloud storage [9]. Data auditing can be divided to private and public auditing schemes. In private auditing schemes the TPA is a trusted organization considered to have the required expertise and means to verify the correctness of the stored data. The private audit may be conducted upon a request, reporting directly to the end-user and possibly to the service provider of any inconsistencies or threats detected regarding the storage [9]. Private TPAs are given access to the stored data, thus requiring active cooperation and legal agreements with the end-user. If any security policies are enforced during the audit, the SSP is similarly required to allow the TPA to verify the implementation and details of the storage service. Private auditing could be very expensive, and in most cases extremely inefficient as the stored data needs to be downloaded in prior to verification. As the SSPs are not expected to allow any third parties to examine the implementation of the storage service, the audit may be expected to cover only the integrity of the stored data.

In order to address the ineffectiveness and shortcomings of private audits, an alternate role has been suggested for the TPAs, often referred to as public auditability. The basic concept in public auditability schemes is to enable verification of the data integrity without exposing the actual data to the TPA, allowing any third party to act as an auditor [9], as illustrated in figure 4. The main benefit in public auditability schemes is the ability to create automated and lightweight audits to the stored data, without revealing any details of the end-user or the stored data to any third parties [9]. Common aspect to all public auditing schemes is the requirement for the end-user to pre-calculate a metadata value, also referred to as challenge token, unique to the data before storing the data to cloud. The challenge token is in its simplest and most inefficient form a hash value of the data, but it may also be for example be a digital signature or a Message Authentication Code (MAC) [9,11]. The pre-calculated metadata is then published to the TPA for performing the audit either automatically or by following a chosen audit plan. In the auditing process the TPA issues an audit request, also known as challenge, to the SSP. Once receiving an audit challenge the SSP is required to calculate a response value to the requested token and respond to the challenge issuer with the calculated value [9]. If the calculated value matches with the original token provided by the end-user, the data has remained intact [9,11].

Homomorphic authenticators have been identified as an efficient and forge-resistant technique of calculating metadata to individual data blocks, allowing a TPA to securely verify the storage integrity without having access to the actual data [9]. However, it has also been identified that a malicious TPA could derive content of the stored data by using the given authenticators derived from the actual data and the challenge responses sent by the SSP [9]. To address such issue, Wang et. al. have proposed integrating a pseudo random function to the SSP in order to include random masking data in the challenge response thus making it impossible to derive the content of the data blocks, while still preserving the validity of the response and the overall audit scheme [9].

Even though public auditability schemes may be able to identify unauthorized changes or corruption in the data, they cannot be used for identifying and preventing unauthorized reads, data deletion and corruption or service outages. Furthermore not only do public auditability schemes require pre-calculation from the end-user, but they also integrate to the storage service, requiring SSPs to modify their service platforms to be compatible with the solution. In addition the solution poses significant amount of overhead to the storage management of the cloud storage services, thus significantly reducing its adoptability from the service providers' perspective.

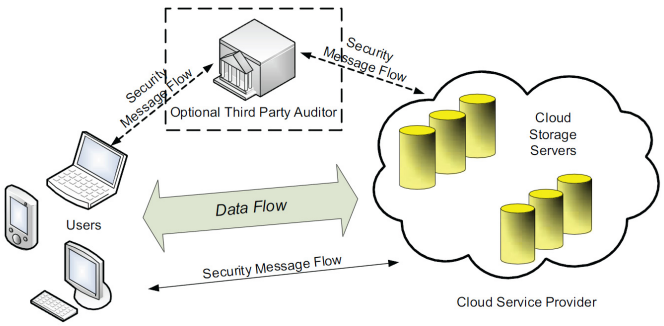


Figure 4. Third Party Auditor in public auditability scheme [9]

V. OVERLAY SOLUTIONS

In order to retain the benefits of cloud storage, a satisfying solution addressing the identified issues in storage security and availability should not introduce any changes to the existing storage services nor assume trust to any third parties. Furthermore a satisfying solution should not introduce any major cost to the end-user, should be adoptable by both the individual users and organizations and finally should support multiple simultaneous users accessing the same resource. One common concern with any cloud services is related to the cost of changing the service provider. As the end-user interfaces provided to access the cloud services are not standardized, each service provider have employed proprietary means to interact with the service. If an end-user solution is designed to interact only with a specific service interface, changing the service provider could require major changes to the overall solution, introducing an issue known as vendor lock-in [4]. Even though the issue might not be very relevant to private consumers, organizations and corporate customers could find vendor lock-in as an obstacle for adopting cloud services.

An efficient way to avoid requiring any changes to the service provider's infrastructure, and to remove any application-level dependencies to the varying service interfaces, is to introduce a solution that forms a completely new layer on top of the provided service. Such services could be considered as overlay solutions, operating on a level that is abstracted from the underlying service [2,4,22]. By relying on an overlay solution, no cooperation to address the specified issues is required from the SSP, and therefore the described issues with privacy, availability and integrity can be solved with solutions controlled completely by the end-user. As the service-related dependencies can be abstracted by the overlay, the solutions are often capable of combining storage resources from several SSPs to a single logical storage resource, addressing any issues regarding the data availability and integrity [2,4,22]. The overlay solutions may also choose to apply encryption to the stored data in order to enhance the privacy against unauthorized reads [4,22].

A. *Cross-SSP Information Dispersion*

The availability and integrity of the stored data can be significantly improved by combining the resources of several SSPs to a single overlay solution [4,22]. In such scenario multiple cloud storages are accessed by the overlay client, replicating the stored data over several cloud providers. However, simply increasing the amount of SSPs to provide redundancy would significantly increase the overall cost of the solution. To avoid purely replicating the entire data, the overlay may utilize an IDA to provide more efficient use of storage resources [4]. Despite or erasure coding, the implementation should place minimum trust to the underlying SSPs, assuming that any cloud provider could fail in a Byzantine way [2,4]. To provide maximal tolerance against simultaneous data corruption and failures in the underlying SSPs, some of the dispersed data could be replicated across varying service providers, introducing a trade-off between integrity and expenses. One promising example of an overlay solution dispersing data across multiple SSPs is DepSky-CA, introduced by Bessani et. al [4], illustrated in figure 5. DepSky-CA provides privacy and integrity by utilizing optimal erasure coding and symmetric encryption with a secret-sharing scheme, dispersing the shared secret between several SSPs in a similar way to erasure coding [4]. DepSky is generally based on quorum protocol design, capable of tolerating Byzantine failures from $N = 3F + 1$ clouds, where N is the total number of applied SSPs and F is the amount of simultaneously failing storages [4]. To satisfy the properties desired from an overlay implementation, DepSky operates completely on client-side, being capable of serving multiple simultaneous readers and a single simultaneous writer thus being adoptable by either a private user or an organization [4].

Every overlay solution needs to implement the means for metadata management in order to utilize the underlying storage services [2,4,22]. One convenient approach for any type of overlay is to persist the metadata within the storage services in order to provide convenient access to the index of content [4]. Similarly the metadata could be distributed between the end-users through some alternative channel such as a separate metadata server, peer-to-peer network or such [2,22]. However, storing the metadata solely on the client would be disputable even in single-user overlays as the storage would be tied to a single client only, and loss of the client would disable the entire storage.

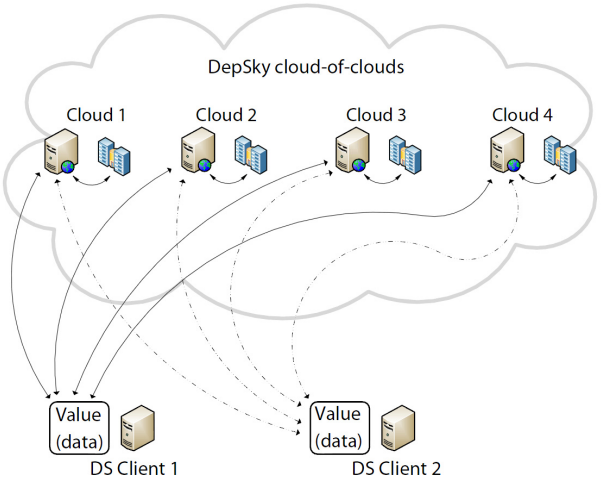


Figure 5. High-level overview of DepSky [4]

B. *Storage Encryption*

Dispersing the data to several SSPs cannot be considered sufficient for protecting the privacy, as all of the data chunks would presumably be stored in plain-text and would thus be accessible by the corresponding SSPs or any potential attackers. If the data would be dispersed to only a small amount of storage providers in an uncontrolled manner, it could be likely that a even a single SSP would contain enough data to be able to reconstruct the original data object. Similar threat rises if SSPs operating in the same region would sell, or be obligated to hand out any stored data to a common third party, such as a local government.

The common third party could combine data from several SSPs, eventually possessing enough data chunks to derive a full plain-text copy of the stored data object.

An intuitive way of increasing privacy is to apply encryption for all of the data stored to the cloud storage. However, a traditional symmetric encryption, such as AES, might not be adequate if a versatile access control scheme is required and the overlay is designed to solely operate on the client-side. For example private users may decide to share some of the selected data objects with a specified group of people or to some specific services. Similarly corporate users often employ different access groups and access levels for specified data objects [4].

Seiger et. al. have suggested a solution for organizations called SecCSIE, introducing a proxy server deployed to a corporate network, shown in figure 6 [22]. The proxy server forms a storage overlay by dispersing data to the connected SSPs similar to the DepSky, but instead of relying on a direct client-side access to SSPs, the storage resources are exposed to the end-users in a centralized manner. In order to provide the required level of privacy, the stored data is encrypted using AES [22]. As the proxy server is considered a restricted-access central hub to the cloud storage services, the same encryption keys are shared by all of the corporate users and managed by the proxy [22]. Even though the solution does not propose any public access interfaces to the encrypted overlay storage, nor any advanced access control mechanisms for managing access groups and levels, such features could simply be added to the internal storage management implementation of the proxy server.

Even though encryption may be considered mandatory in order to provide privacy, applying an encryption may become disputed by some SSPs for a couple of reasons. Since encryption is expected to increase the overall entropy of the data [29], the efficiency of data deduplication algorithms employed by the SSP could significantly decrease, thus increasing the amount of required storage and thereby increasing the expenses caused to the SSP. Also as some of the SSPs targeted to private consumers might gain part of their revenue from mining the stored data, applying a client-side encryption could have an impact to the revenue gained by the SSP, thus possibly even leading to service agreements forbidding the use of any kind encryption algorithms.

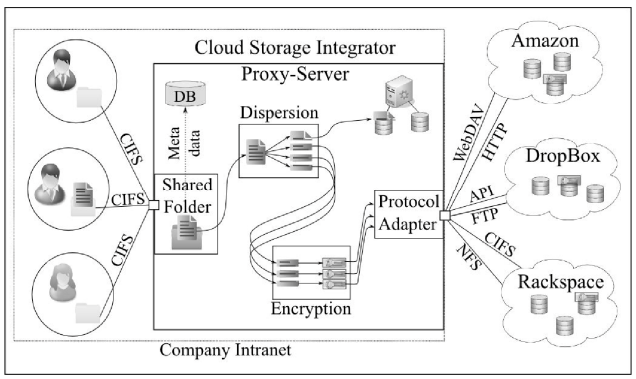


Figure 6. Design of SecCSIE Proxy server [22]

C. *Key Management*

One essential issue in encrypting the storage is related to key management. If the storage content is protected using symmetric encryption, sharing the data would generally require sharing the encryption key with the targeted group of people. If all of the stored data is encrypted with the same key, revealing the key would be impossible. Even if all of the files would be encrypted with different keys, a compromised key would compromise the specific data object. Sharing the encryption keys with dedicated group of people could similarly be subject to leakage of the shared key. Such leak would be crucial for the privacy of the stored data object, as detecting leaked keys is such scenarios could be considered impossible. Even though DepSky doesn't directly address the question of sharing the encrypted data, the secret sharing scheme applied in the solution maximizes the privacy against unauthorized reads causing very minimal key management overhead to the user [4]. Solutions such as SecCSIE may easily abstract the key management as part of the internal implementation of the solution to reduce the overhead caused by key management, but any solution intended to operate mainly on the client-side would have to employ more complex key-sharing and revocation mechanisms [14].

In order to make the key management more feasible, either asymmetric encryption, such as RSA, or hybrid solutions combining both symmetric and asymmetric encryption are suggested [14]. Relying solely on techniques such as Public Key Infrastructure (PKI) to encrypt and share the data might not always be sufficient either, as the assumed file-keys would need to be encrypted with a public key specific to the targeted user [30]. As the same public key could be used for several other occasions, possibly being distributed to other users and services as well, the privacy of the data would be compromised to any other parties in possession of the same public key. Also, sharing the data with a group of people would require encrypting the file-key with the corresponding group-key and a separate key-management scheme to control the group key [30]. To create an effective key management scheme, Tang et. al. have introduced an overlay solution called FADE, relying on separate key managers deployed and maintained by the end-user [14]. The access management of FADE employs three types of encryption keys, named data, control and access keys, consisting of both symmetric and asymmetric keys. The three layers of keys are utilized in combination to encrypt the actual data and to enforce any deployed access policies related to the stored data objects [14]. However, as FADE requires separate key managers to provide the basis for access control, the solution may not be adoptable by individual users, unless the key managers could be publicly accessed.

VI. CONCLUSIONS AND FUTURE WORK

This paper described the common architecture of cloud storage services accompanied with the identified issues reducing the trust and possibly even impeding the adoption of cloud services. The described shortcomings and issues of cloud storage services were described from the perspective

of individual users and organizations. Third party auditors were introduced as an option to verify the correctness of the stored data, and the public auditability scheme was described, providing feasible auditing through automated and lightweight verification of audit metadata. As auditing may only verify the correctness of the stored data, the concept of overlay solutions was introduced to address the described security and availability issues related to cloud storage. In order to satisfy the desired properties of a storage overlay, several solutions utilize cross-SSP data dispersion, relying on several cloud providers to provide enhanced availability. Similar to the storage providers, overlay solutions often not only disperse the data, but also employ erasure coding to further enhance the integrity of the stored data.

As dispersing the data to several SSPs is not adequate to provide the required privacy, encryption needs to be applied to the stored data. However, applying encrypting introduces new difficulties in sharing the data. As the stored data needs to be accessible from several different locations, and often by more than one user, symmetric encryption may not be applicable without a central actor enforcing any access policies and managing the encryption keys. Additional key management schemes are also required even if asymmetric encryption is used. To fully meet the desired aspects, the overlay solution should be capable of providing the required security while maintaining the data availability to any selected group of users. Further studies are required on specifying an implementation that would meet such requirements without introducing any additional components, such as key management servers. PKI could prove out to be a promising alternative if the end-user certificates could be effectively and securely utilized.

REFERENCES

[1] I. Ion, N. Sachdeva, P. Kumaraguru and S. Capkun, “Home is safer than the cloud! Privacy concerns for consumer cloud storage”, Symposium on Usable Privacy and Security (SOUPS), 2011, pages 14-16.

[2] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin and M. Walfish, "Depot: cloud storage with minimal trust", ACM Transactions on Computer Systems volume 29, issue 4, Article 12, 2011.

[3] W. Zeng, Y. Zhao, K. Ou and W. Song, “Research on cloud storage architecture and key technologies” ICIS 2009 Proceedings of the 2nd international conference on interaction sciences: Information Technology, culture and human, 2009, pages 1044-1048.

[4] A. Bessani, M. Correia, B. Quaresma, F. André and P. Sousa, “DepSky: dependable and secure storage in a cloud-of-clouds”, EuroSys'11, 2011, pages 31-46.

[5] M. Munier, V. Lalanne and M. Ricarde, “Self-Protecting documents for cloud storage security”, IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, 2012, pages 1231-1238, DOI 10.1109/TrustCom.2012.261.

[6] J. Feng, Y. Chen, D. Summerville, W. Ku and Z. Su, “Enhancing cloud storage security against roll-back attacks with a new fair multi-party non-repudiation protocol", Consumer Communications and Networking Conference (CCNC), 2011, pages 521-522, DOI 10.1109/CCNC.2011.5766528.

[7] S. Han and J. Xing, "Ensuring data storage security through a novel third party auditor scheme in cloud computing", Cloud Computing and intelligence Systems (CCIS), 2011, pages 264-268. DOI 10.1109/CCIS.2011.6045072.

[8] X. Zhang, H. Du, J. Chen, Y. Lin and L- Zeng, “Ensure data security in cloud storage”, Network Computing and Information Security (NCIS), 2011, pages 284-287, DOI 10.1109/NCIS.2011.64.

[9] C. Wang, Q. Wang, K. Ren and W. Lou, “Privacy-preserving public auditing for data storage security in cloud computing”, IEEE INFOCOM, 2010, pages 1-9, DOI 10.1109/INFCOM.2010.5462173.

[10] M. Tribhuwan, V. Bhuyar and S. Pirzade, “Ensuring data storage security in cloud computing through two-way handshake based on token management”, International Conference on Advances in Recent Technologies in Communication and Computing, 2010, pages 386-389, DOI 10.1109/ARTCom.2010.23.

[11] M. Venkatesh, M.R. Sumalatha and C. SelvaKumar “Improving public auditability, data possession in data storage security for cloud computing”, Recent Trends In Information Technology (ICRTIT), 2012, pages 463-467, DOI 10.1109/ICRTIT.2012.6206835.

[12] Y. Wei, Z. Jianpeng, Z. Junmao, Z. Wei and Y. Xinlei, “Design and implementation of security cloud storage framework”, Second International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), 2012, pages 323-326, DOI 10.1109/IMCCC.2012.79.

[13] R. Roman, M. Felipe, P. Gene and Jianying Zhou, "Analysis of security components in cloud storage systems", APMRC, 2012, pages 1-4.

[14] Y. Tang, P. Lee, J. Lui and R. Perlman, "Secure overlay cloud storage with access control and assured deletion", IEEE Transactions on dependable and secure computing, volume 9, issue 6, 2012, pages 903-916, DOI 10.1109/TDSC.2012.49.

[15] A. Kumar, B. Lee, H. Lee and A. Kumari, “Secure storage and access of data in cloud computing”, International Conference on ICT Convergence (ICTC), 2012, pages 336-339, DOI 10.1109/ICTC.2012.6386854.

[16] M. Hamdi, "Security of cloud computing, storage, and networking", Collaboration Technologies and Systems (CTS), International Conference, 2012, pages 1-5, DOI 10.1109/CTS.2012.6261019

[17] Dropbox, 2013, https://www.dropbox.com/

[18] Dropbox API, 2013, https://www.dropbox.com/developers/core

[19] Just Cloud, 2013, http://www.justcloud.com/

[20] Cubby, 2013, https://www.cubby.com/

[21] Microsoft SkyDrive API, 2013, http://msdn.microsoft.com/en-us/library/live/hh826521.aspx

[22] R. Seiger, S. Groß and A. Schill, "SecCSIE: a secure cloud storage integrator for enterprises", 13th Conference on Commerce and Enterprise Computing (CEC), 2011, pages 252-255, DOI 10.1109/CEC.2011.45

[23] S.V. Gogouvitis, G. Kousiouris, G. Vafiadis, E.K. Kolodner and D. Kyriazis, "OPTIMIS and VISION cloud: how to manage data in clouds", Proceedings of the international conference on Parallel Processing (Euro-Par'11), 2011, pages 35-44, DOI 10.1007/978-3-642-29737-3_5

[24] G. Bombach, S. Matthischke, J. Muller and R. Tzschichholz, "Information dispersion over redundant arrays of optimal cloud storage for desktop users", Fourth IEEE International Conference on Utility and Cloud Computing (UCC), 2011, pages 1-8, DOI 10.1109/UCC.2011.11

[25] J.S. Plank and L. Xu, "Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications", Fifth IEEE International Symposium on Network Computing and Applications (NCA), 2006, pages 173-180, DOI 10.1109/NCA.2006.43

[26] Payment Card Industry Security Standards Council, "Requirements and Security Assessment Procedures, Version 2.0", 2010, https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf

[27] CloudDIP, 2013, http://www.clouddip.com/healthcare/

[28] US Department of Health & Human Services, Federal Register Vol. 62, No. 34, February 2003 "45 CFR Parts 160, 162 and 164 Health Insurance Reform: Security Standards; Final Rule", February 2003, http://www.hhs.gov/ocr/privacy/hipaa/administrative/securityrule/

[29] M.W. Storer, K. Greenan, D.D.E. Long and E.L. Miller, "Secure data deduplication", Proceedings of the 4th ACM international workshop on storage security and survivability (StorageSS '08), 2008, pages 1-10, DOI 10.1145/1456469.1456471

[30] V. Kher and Y. Kim, "Securing distributed storage: challenges, techniques and systems", Proceedings of the 2005 ACM workshop on storage security and survivability (StorageSS '05), pages 9-25, DOI 10.1145/1103780.1103783

Secure Cloud Application

Javad Sadeqzadeh Boroujeni
Department of Computer Science
University of Helsinki
Email: sadeqzad@cs.helsinki.fi

***Abstract*—Cloud computing as a big step toward grid computing provides computing as a utility. However, security (if not addressed correctly) is one of the serious obstacles in the spread of cloud computing. In this article, we review some aspects of cloud computing security. In section I, we discuss the importance of security challenge in cloud computing, and the differences in various cloud service models from security perspective. In section II, we overview general security concerns which are the issues related to the nature of cloud model and the considerations that an entity should take into account if it intends to migrate to cloud. These concerns include data privacy and trust, data integrity, virtualization vulnerabilities, web application firewall, data availability, interoperability, and authentication and authorization. In section III, we overview application security concerns. In this section, first, we take a look at software assurance and seven principles that support it in a cloud environment. These principles include confidentiality, integrity, availability, authentication, authorization, auditing, and accountability. Then, we give some insights about secure development practices. These include some practices and guidelines about handling data, code practices, language options, and input validation and content injection. In addition, we describe the characteristics of security requirement engineering process for cloud application development. Furthermore, we provide a security checklist to enable the entity which is considering migrating to cloud to assess different cloud service providers. Eventually, we conclude with emphasizing the importance of balance between the three important elements which are security, functionality, and ease of use.**

I. Introduction

Regarding the recent increasing use of multi-tenant cloud computing and services, the security has become one the areas of cloud computing which has attracted much attention. Multi-tenancy essentially refers to the fact that the users of many customers of a single cloud service provider are served on single physical server (regardless of cloud service delivery model). This make the misuse of security vulnerabilities effective on a larger scale. The interference of a potentially malicious cloud customer on other customers on the same physical machine is one the challenges in this regard.

As Fujitsu Journal customer questionnaire depicted, security is the most important concern of cloud computing customers [2]. In order to achieve widespread use of cloud computing and services, there are several areas in which security issues should be addressed. Different cloud service models (i.e. Infrastructure as a Service, Platform as a Service, and Software as a Service) establish different trust boundaries and therefore, there are some differences in security concerns of each model (and the severity of the issues), though many of the concerns are common among all. For example, PaaS and SaaS environments introduce additional threats that come from software and platform administrators that work for an external entity (cloud service provider) [15]. The level of isolation is a major difference between the three cloud service models. At IaaS level, isolation is done at virtual machine image level, while at Paas and SaaS, isolation is done at process level, which is more difficult to achieve and maintain in a multi-tenant environment. Since using IaaS level brings the lowest level isolation, and moreover, the enterprise does not need to trust to third parties' (e.g. cloud service providers) APIs (which might have security ambiguities), we believe that this model should be preferred over PaaS and SaaS by large enterprises. However, as we will see in the next section, using this level has its own risk, which should be analyzed carefully. In this article, we give an overview of general concerns and some guidelines in order to incorporate security into the software development life cycle.

We can categories security concerns and issues into two categories: general security concerns, and application security concerns.

II. General Security Concerns

General security concerns can be further divided into universal cloud model vulnerabilities, and cloud-specific vulnerabilities. As GroBauer *et al.* pointed out, the prevalent mistake is that we usually do not distinguish between universal cloud model vulnerabilities and cloud-specific vulnerabilities [7]. We review both of them briefly under this section.

A. Data Privacy and Trust (aka Data Confidentiality)

As the name suggests this category of issues is associated with the privacy or confidentiality of customer's data on cloud service provider's (CSP) infrastructure. Since the storage of files on the cloud is location transparent (and relocation transparent), and moreover, data replication for availability and disaster recovery is a core operation at cloud, this lack of awareness of data location leads to increased customers' concerns about their data privacy. There are some arguments that most of the issues in this category come back to the storage transparency nature of cloud model and the "lack of control over infrastructure" [1].
Customers (especially enterprises with sensitive data) are concerned about the individuals and organizations that can see and access their data, including cloud service provider's personnel, governments, etc. Encryption is a solution to data

privacy concerns; before storing any data item into the cloud, cloud consumers should encrypt it using cryptographic key files, and after restoring (downloading) they should decrypt it. However, this method is not appropriate for many enterprises, since they need quick and in-place processing of data on the cloud. Of course, some novel methods have been proposed by researchers to process encrypted data, but they are not very practical and thus are not likely to be usable in near future. Nevertheless, some approaches have been suggested to search encrypted data on the cloud without decryption. These methods are more likely to be utilized in near future. Anyway, for some of enterprises, nobody except the data owner must be able to access their data, not even the CSP's (cloud service provider) technical administration personnel. In this case, data confidentiality comes to the top of the preferences list when deciding on the migration to the cloud, and in cloud service provider selection phase.

B. Data Integrity

This category of issues concerns to the malicious modification of customer's data. The altered data may be used by the data owner to make critical decisions [3]. There is a need for a mechanism that can check whether the consumer's data is intact while at rest or in transit [4]. According to Rong et. al, at the moment, there is no common standard to guarantee data integrity.

C. Virtualization Vulnerabilities

Although virtual machines in Infrastructure as a Service model are supposed to be isolated at a lower level in contrast to Platform as a Service and Software as a Service, there are some known vulnerabilities with virtualization software. Virtualization risk is not a exclusive risk of cloud computing environment; it exists also in the traditional web hosting environments. However, the multi-tenancy nature of the cloud makes the adverse effects of this risk more severe. An attack to a virtual machine can influence other virtual machines on the same physical server [3] or can even lead to destruction or loss of control of cloud service provider's infrastructure.
Virtualization security strength depends on the used hypervisor in the CSP's infrastructure. Hypervisor or virtual machine monitor (abbreviated as VMM) is a system software similar to an operating system, but theoretically more lightweight and less complex than an operating system. However, in practice, hyporvisors become as complex as operating systems. This complexity leads to vulnerabilities in their architecture. The main responsibility of a hyporvisor is to manage virtual machines, to distribute resource among them, and to isolate them. Oracle VM Server for SPARC, Oracle VM Server for x86, the Citrix XenServer, VMware ESX/ESXi, KVM, and Microsoft Hyper-V are the common hypervisors. However, some hypervisors fail to isolate virtual machines properly and as a result, malicious virtual machine owners can interfere in other virtual machines. When choosing a cloud service provider, an enterprise should try to figure out the hypervisor used by various cloud service providers and investigate their vulnerabilities. A good approach for both virtualization security and operating system security (which leads to general cloud security) is to install the latest patches periodically and automatically. This is called patch management.

D. Web Application Firewall

Web application firewalls (WAFs) are common means of security for traditional web applications used in traditional data centers or web hosting facilities. These firewall should be modified in order to be usable in a cloud environment. The main goal here is to make WAFs scalable, flexible, and easy to manage [15]. They should not be limited by hardware, they should "scale across CPU, computer, server rack and datacenter"; their resource consumption should be minimal and they should not add "undue" complexity for CSPs; they should be scalable enough to manage "highly-loaded web applications"; and because of various shapes and sizes of clouds, web application firewalls should be highly configurable to different scenarios [15]. This new type of WAFs are called distributed WAF or dWAF [15].

E. Data Availability (aka Business Continuity)

Can a particular cloud service provider safely and securely recover the sensitive data and applications of customer in case of confronting a disaster? Can an enterprise rely on a cloud service provider and outsource its critical-mission applications to it? Data availability or business continuity is one of the factors which an enterprise should use to build its external "due diligence" which refers to the examination of a cloud service provider's capabilities to meet the enterprise's requirements [5].

F. Interoperability (aka vendor lock-in)

Can an enterprise easily and smoothly migrate from the current cloud service provider (CSP) to another CSP in case of dissatisfaction with the current CSP? How much difficulty (in terms of data and application transformation, service interruption, etc) the enterprise will encounter in case of such shift? The lack of a widely accepted cloud standard has led to a poor interoperability situation among various CSPs. With the current situation, it is most probable that migration of a cloud customer to another CSP imposes a load of difficulties. This situation is called vendor lock-in. Although interoperability does not seem to have a direct impact on security since it is related to data availability, in most of the literature it has appeared together with other security concerns.

G. Authentication and Authorization (Access Control)

Authentication refers to "the mechanism whereby systems may securely identify their users" [6]. In contrast, authorization specifies "what level of access a particular authenticated user should have" [6]. In multi-tenant cloud environments, the traditional authentication mechanisms of data owners and users may not be as effective as they are in the traditional web hosting environments. How to keep and manage account

tokens (keys) securely? How to restrict unauthorized access? These questions are related to authentication and authorization.

III. Application Security Concerns

If a enterprise or developer uses Infrastructure as a Service (IaaS), as we suggested, there is not much difference in application-level security concerns comparing traditional web applications. Arnold *et. al* also mentioned this fact: "The architecture for IaaS-hosted applications resembles that of normal web applications, with a web-server based, n-Tier distributed architecture" [15]. Anyway, creating and using your own securely configured virtual machine image is usually preferred over using pre-ready virtual machine images provided by cloud service providers. However, even in this case, you should secure your inter-host communications since the infrastructure is shared with other entities [15].

However, if an entity intends to develop software for Platform as a Service (PaaS), it should manage application keys securely properly. When you use this service model, you need to include those keys in API calls. An application key should not be stored in clear text; it should be secured with other credentials the application utilizes [15]. Another important issue you should take into account when designing and implementing applications for PaaS is that PaaS platform's service bus is equivalent to an ESB [15], however, you should remember that because of multi-tenancy nature of cloud, you should not assume that this ESB is secure. PaaS platform?s ESB provides both asynchronous messaging and message routing, but before using them, you should secure your message communication [15].

Another key point about the security of PaaS is that "each PaaS platform has its own unique security challenges" [15], therefore, before using a PaaS, you should make yourself familiar with its security challenges. However, as Arnold *et. al* mentioned, "knowledge bases for PaaS environments are scarce" yet [15].

If you develop software for SaaS service model, you need to take care of your API calls carefully, although there are some existing security policies and standards defined by each SaaS provider [15]. Data communication should be secured in this service model as well. The source or the destination of data can be applications within the same cloud, within the enterprise, or within other clouds [15]. Since "secure software development life cycle is shared between the SaaS vendor and the enterprise" (or the developer) in this service model [15] , the enterprise or the developer should mind about how their secure SDLC integrates with SaaS vendor's secure SDLC.

When developing software for the cloud, security should be treated as an integral element of software development life cycle, not just an add-on. In order to produce secure cloud software, we first need to answer the question: "what makes software vulnerable?". Goertzel *et al.* enumerated three major factors [10]: lack of developer motivation, lack of developer knowledge, lack of technology. lack of developer motivation refers to the fact that because consumers evaluate the software vendors by being first in the market, and having more features, not by producing higher quality and more secure, software vendors have no financial motivation to produce better and more secure software [10]. Lack of adequate and appropriate tools to assist software developers in the software development life cycle is called "lack of technology". And the factor that we target in this article (lack of developer knowledge) is because software and generally information systems are very complex; "it exceeds the human ability to fully comprehend it" [10].

At the other hand, software security has a high degree of importance. As the US President's Information Technology Advisory Board (PITAC) emphasized in their report (on cyber security) to the US president in 2005 [11], software is an major component in cyber security; it worsen the security problem. In the aforementioned report, they signified: "Network connectivity provides "door-to-door" transportation for attackers, but vulnerabilities in the software residing in computers substantially compound the cyber security problem" [11]. They artistically depict how an insecure software can be used to damage the information systems: "Today, as with cancer, vulnerable software can be invaded and modified to cause damage to previously healthy software, and infected software can replicate itself and be carried across networks to cause damage in other systems" [11]. They continue with a preventive suggestion: "And as in cancer, both preventive actions and research are critical, the former to minimize damage today and the latter to establish a foundation of knowledge and capabilities that will assist the cyber security professionals of tomorrow reduce risk and minimize damage for the long term" [11]. In order to undertake such preventive actions, we need to know how we can systematically reduce the vulnerabilities in our software systems during the software development life cycle (SDLC). *Software assurance* provides some sound knowledge for such systematic prevention.

A. Software Assurance (aka Software Security Assurance, and Information Assurance)

As Goertzel *et al.* pointed out in a report called Software Security Assurance - the State-of-the-Art Report [10], until 10 years ago, the term software assurance was used to refer to two software properties: quality, and reliability; since ever, the term has been "adopted to express the idea of the assured security of software". The aforementioned report defines *software assurance* precisely: "to provide a reasonable level of justifiable confidence that the software will function correctly and predictably in a manner consistent with its documented requirements. Additionally, the function of software cannot be compromised either through direct attack or through sabotage by maliciously implanted code to be considered assured."
The objectives of software assurance are [9]:
Dependability: The software shows a "justifiable confidence" that it works predictably or "only as intended".
Trustworthiness: The software has no "exploitable" vulnerabilities or "malicious logic".
Resilience: The damage to the software resulting from a compromise is minimized. In such situation, the software recovers quickly and to an "acceptable level of operating

Fig. 1. The CIA Triad

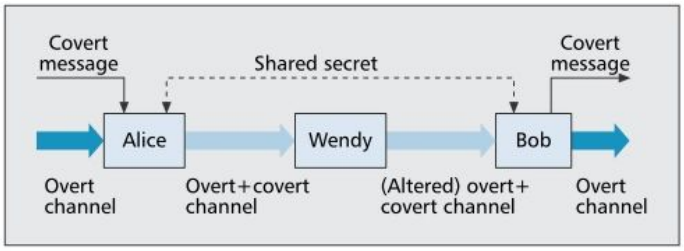


Fig. 2. The de-facto model of covert channel communication [13]

capacity".

Conformance: The software conforms to the (security) requirements and the relevant standards and procedures. In order to ensure conformance, a "planned and systematic set of multidisciplinary activities" should be performed.

When developing software for cloud, the importance of considering software assurance and its principles becomes more crucial since because of the multitenancy nature of cloud and the scale, security, as a matter of quality, is more vital and harder to achieve in cloud environment comparing to traditional desktop environment or even traditional web hosting environment where establishing trust is not such challenging as it is in cloud computing environment.

Krutz and Dean Vines aggregated 7 security principles that support information assurance [8] . As figure 1 depicts, the first three principles are known as the *CIA triad* of information system security [16] [8]. These three principles are also mentioned in other literature (though in a slightly broader view) [12] [16] and are as follows:

Confidentiality: Confidentiality is associated with the prevention of unauthorized information revealing, either intentional or unintentional [8]. It covers the following areas:

Intellectual property rights: Rights to intellectual properties should be protected by the national and international copyright laws, which covers patents, artistic works, designs, literary works, and musical works. [8]

Covert channels (aka subliminal channels): This terms refers to "unauthorized and unintended communication paths" that can be utilized for secret transfer of information [8]. There are two types of covert channels: storage covert channels and timing covert channels. Storage covert channels (that can be the result of inappropriate usage of storage mechanisms [8]) can be utilized for writing (or overwriting) of object values, or reading object values [13]. Zander *et al.* have reviewed common methods for detecting, removing and preventing covert channels in network protocols [13]. The de-facto model of covert channels is shown in figure 2.

Traffic analysis: This is a security breach that can be done through message traffic analysis, even if the traffic is encrypted. By analysis of the volume, rate, source, and destination of the traffic, attackers can obtain information to undertake malicious actions [8]. An approach to address this breach is to preserve a near-constant rate of traffic [8]. Fu *et al.* applied a statistical pattern recognition solution to evaluate the effectiveness and to counteract active traffic analysis attacks [14].

Encryption: Encryption refers to the encoding of messages' contents using cryptographic keys so that unauthorized entities could not be aware of the contents of messages. The strength of the encryption key, together with the quality of the encryption algorithm specify the robustness of the encryption [8].

Inference: Inference refers to the "use and correlation of the information protected at one level of security to disclose information protected at a higher level of security" [8]. The lower level here usually refers to database level.

Integrity: Integrity is e general term that is being used for different levels, including data, software, and hardware [12]. In order to guarantee data integrity, three requirements should be met [8]

- No modification or alteration to data is done by unauthorized people or processes.
- No unauthorized modification or alteration to data is done by the authorized people or processes.
- Data is internally and externally consistent.

Authentication, authorization, and auditing (that will come in the rest of the article) are three means that should be used for guaranteeing data integrity.

Software integrity refers to prevention of unauthorized fabrication, removal, alteration, and theft [12]. Since the cloud software which are designed to be deployed to PaaS or SaaS environments utilize and rely on the programming interfaces from the cloud service provider, for those software, it is vital to ensure about the security of those interfaces, since the security and integrity of the software heavily depends on the security of the interfaces and components used [12].

Availability: Availability refers to the accessibility and usability of a system when it is needed [12]. A denial of service (DoS) attack is an example of threats against availability [8]. The system must be able to continue to function even in the possibility of a security breach [12].

The remaining four principles directly affect cloud software assurance [8].

Authentication: As in general security concerns section mentioned, authentication refers to "the mechanism whereby systems may securely identify their users" [6]. Winkler defines authentication as the establishment of a user's identity [16].

Traditionally a system authenticate a user if the credentials provided by the user correspond to those in the database.

However, this conventional method called single-factor authentication is not appropriate and secure for sensitive systems since the user's credentials may be easily leaked due to user's mistakes or lack of care, or social engineering methods, or even physical attacks. Therefore, we need stronger authentication methods for the multi-tenant environment of the cloud. Some novel approaches have been introduced during recent years. Multi-factor authentication is a method of authentication in which in order to establish their identities, users should perform several actions rather than simply providing their credentials. Using STP generator devices in combination with the traditional credential-based authentication is a multi-factor authentication method which has been used by many financial institutions and banks around the world to authenticate their users when committing financial transactions or initiating major modifications to the user's account. Using communication channels secured with digital certificates is another alternative that can substitute the traditional insecure credential-based authentication.

Some modern multi-factor scheme are introduced in very recent years. One of them, for example, suggested to use a combination of digital signatures and third level features of fingerprints (a bio-metric factor) as a 2-factor (2FA) anonymous password authentication scheme [17]. There are three levels of details (features) in human fingerprint. The first level called pattern level usually represents the pattern type of the fingerprint. The second level called points level represents "minutiae points such as spur, bifurcation, and terminations" [17]. And finally the third level called shape level contains all "dimensional characteristics of a ridge". According to Yassin et al., the first level features are not unique and should not be used for identification purposes, but the second and third levels provide individuality and each one of them can be used for identification purposes. This scheme looks very strong since it utilize mutual authentication and it can be used to bypass some threats like DNS cache poisoning (aka DNS spoofing). Figure 3 depicts the details of this scheme. As the name suggests, mutual authentication refers to the two-way authentication in which in addition to the authentication of user by the system, the user also authenticates the system in order to avoid providing credentials to a fake system (for example by malicious modification of DNS entries to forward traffic towards a malicious server).

In addition to the aforementioned authentication methods, three are some other methods of authentication, which can be utilized to provide stronger authentication. Some examples of other authentication methods are graphical and 3D passwords. Recently, in some efforts to apply the SSO (single sign-on) theory, there has been a trend in use of third party authentication programming interfaces, like Facebook authentication API. External APIs are means for delegation of authentication or authentication outsourcing. OAuth (Open Authentication) protocol, which was the result of efforts for standardizing external authentication, was invented in 2007. However, external (third party) authentication APIs might not be secure for sensitive systems, like enterprises' mission-critical cloud software, since the system security will depend on security of the used third party API. And for designing a secure system, external security dependencies should be limited.

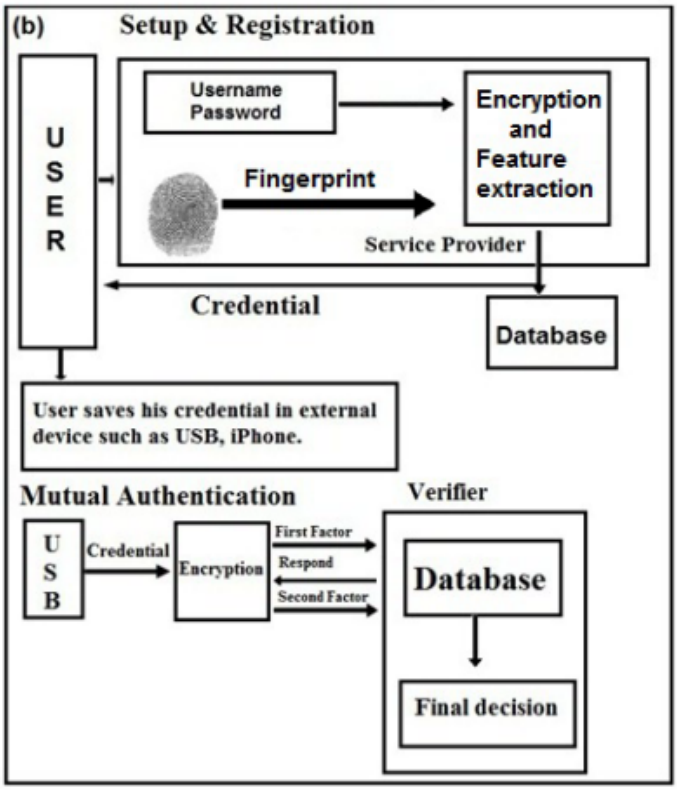


Fig. 3. Mutual Two-Factor (2FA) Authentication Using Anonymous Password Authentication Scheme [17]

However, the strength of an authentication method should not be the mere measure. When thinking about security, we should have ease-of-use in mind as well. There is a trade-off triad between security, ease-of-use, and functionality. Sometimes complexity of security mechanisms and procedures makes it difficult for user to use the system and as a side-effect, might finally lead to adverse impacts on users decision.

Authorization: Authorization refers to the process of specifying user's access rights and privilege extent. There are several means of authorization. ACLs (Access Control Lists) as a mean for authorization are simple lists of users and their access rights against either specific resources or classes of resources [16]. After users get authenticated, they should be appropriately authorized to be able to access to only those sections of system and those date that is absolutely necessary for them. There is a security principle called least privilege principle which states that users should be limited to a "minimal set of privileges". This principle is directly associated to authorization. Within a multi-tenant environment such as the cloud, especially in SaaS and PaaS models, in addition to software developer, the cloud provider should also undertake a strict authorization to cloud consumers.

Auditing: System audits and monitoring are two mechanisms that enterprises can use to achieve and maintain operational assurance [8]:
System Audit: A system audit is an event, either one-time or periodic for the purpose of security evaluation [8]. There are two types of IT auditors: internal and external. External auditors are "certified public accountants (CPAs) or other audit professionals which are hired to perform an independent audit" [8]. Internal auditors work for a certain enterprise, and usually have a much broader mandate [8].
Some techniques for system auditing are source code review, binary code review, threat modeling, penetration test, etc.
Monitoring: This term refers to an continuous activity that evaluate either the system or the users. Intrusion detection systems are examples of monitoring tools.

Accountability: This term refers to the ability of tracing behaviors of users and accurately identifying the misbehaving users. Audit trails and logs are means of accountability [8].

B. Secure Development Practices

Handling Data: Sensitive and private data need special care. Krutz *et al.* provided some important guidelines for handling sensitive data [8]. These guidelines are as follows:
- Passwords must not be saved or transmitted in clear text.
- The passwords must not be viewable on the screen, when users are entering their passwords.
- Passwords must be encrypted with one-way hashes.
- Credit card information must not be transmitted in clear text.
- Cloud servers must minimize the transmission and printing of credit card information. Since this information might be logged and printed in internal reports that are used for troubleshooting for example.
- Sensitive data (such as credit card numbers) should not be transmitted to server as part of the query string.

Code Practices: The information included in the source code of a cloud application must be minimized [8]. Account tokens and/or keys must not be included in the source code. Names and other personal information must not be included in source code comments. HTML and client-side scripts comment fields are very sensitive since they are viewable by clients; only non-sensitive information should be in that comments. There should not be any clue about the architecture or structure of the cloud application in HTML or client-side script comments. This comments should not disclose any exploitable information about the organization or the developers [8]. Extra care should be applied when working with the extra software packages such as web servers, since they usually display the version of the software. The attackers may use this version information to concentrate their attacks toward the vulnerabilities of this version [8]. As mentioned earlier, some programming languages require more care, because of their security mechanisms limitations; if there is a obligation regarding the use of those languages, the developers should train themselves and practice to improve their skills to manage security manually and carefully.

Language Options: Each programming language has its own strength and weakness points. Some languages are more appropriate or have rigorous security management for cloud software development. As Krutz *et al.* pointed out, one of the most frequent vulnerabilities in cloud applications is the result of using C and C++ [8]. Since C cannot manage buffer overflow, use of it for developing cloud application is risky and requires high level of developer's skill in managing memory and safe programming techniques. The developer should check for the boundary limits and make sure that all functions are correctly called [8]. A reason of popularity of Java is JVM's capabilities for security and memory management.

Input Validation and Content Injection: User input must be immediately verified and validated before passing it around for any further action. Content injection attack which is a result of lack of sufficient user input validation refers to those kind of attacks that the attacker inserts malicious contents (such as SQL commands) into the system through user input. *SQL injection* refers to the attacks that the attacker tries to alter the database using injecting SQL commands through parameters or text boxes. A SQL injection can be used to drop all tables or a specific table in the database. *Cross-site scripting (XSS)* is another type of content inject attacks which abuse the lack of (or weak) user input validation to inject client-side script to bypass access controls. These kind of attacked are happened when the user input is used (without validation) to display a page of results.

C. Security Requirement Engineering

As we mentioned earlier, security mechanisms should not be treated as some add-ons, but rather they should be in very heart of the system. Therefore, as Goertzel *et al.* [9] pointed out, we need to enhance software development life cycle to produce secure software. Cloud software security requirement specification is an integral part of cloud software engineering. Security requirements should be defined as early as possible in the software development life cycle. Like with legacy software, the process of cloud software requirement engineering produces both functional and non-functional software characteristics. Figure 4 depicts the traditional requirements engineering process for survivable systems. Figure 5 illustrates secure software additions to the requirements engineering process.

Software Security Requirements Properties (SMART+):
In 1995, in their paper, Mike Mannion and Barry Keepence (from Edinburgh Napier University) proposed five basic properties of software requirements which is abbreviated as SMART. These properties are: Specific, Measurable, Attainable, Realizable, and Traceable. The Open Web Application Security Project (OWASP) has modied the acronym to SMART+ [18], which stands for the terms that follows:
Specific: The requirements should be detailed, so that there are "no ambiguities" in any single requirement.
Measurable: "It should be possible to determine whether the requirement has been met, through analysis, testing, or both."

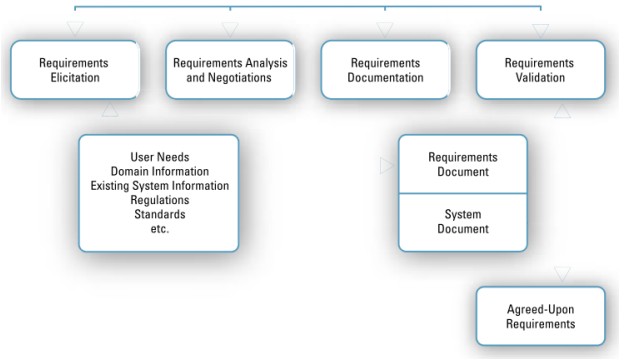


Fig. 4. Requirements Engineering for Survivable Systems [9]

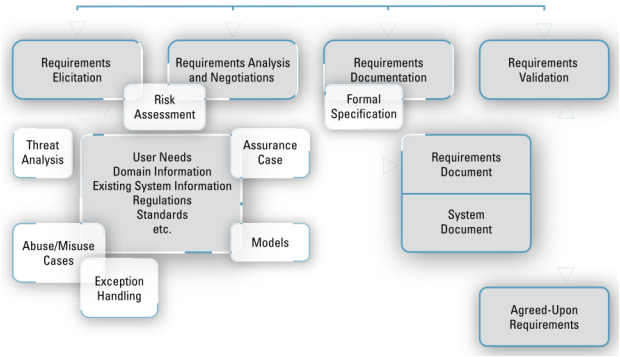


Fig. 5. Secure Software Additions to Requirements Engineering Process [9]

Appropriate: "Requirements should be validated, thereby ensuring that they not only derive from a real need or demand but also that different requirements would not be more appropriate."
Reasonable: validations should be done to to specify the physical feasibility or feasibility with regard to other project constraints.
Traceable: "Requirements should be "isolated" to make them easy to track/validate throughout the development life cycle."

As Goertzel *et al.* stated, security requirement engineering is hard to do: "The number of requirements needed to adequately define even a small software project can be very large. That number grows exponentially with the effects of large size, complexity, and integration into evolving operational environments" [10].

Like other requirements, there are two types of security requirements: Internal, and external [8]. In security context, internal requirements refer to non-functional and functional security requirements which correspond to the enterprise's security policy. External security requirements refer to compliance of software details to national regulations and legal obligations. For example, in most countries sensitive and categorized governmental information cannot reside on abroad locations, or even cannot be placed on networks belonging to third parties. Regulatory compliance is a determining security policy factor in cloud service provider selection phase.

D. Cloud Service Provider Assessment Checklist

Choosing an appropriate cloud service provider that can meet the security requirements (and other requirements) is one of the critical decisions that an enterprise (or even a SMB (small or medium size business)) make. This decision should be based on the enterprise's security policy. As Karadsheh pointed out, due diligence serves as an input to security policy formulation [5]. In their Guidance for Application Security V2.1, the Cloud Security Alliance (CSA) listed a set of concerns to be incorporated into due diligence [15]. The following questions which should be asked from the candidate CSPs (cloud service providers) cover the "application-layer-specific" concerns [15].

The security checklist for all service delivery models:
- What "Secure Development Life Cycle" activities does the cloud service provider use in developing the service's software?
At design level: threat modeling, and secure design reviews
At implementation level: manual secure code reviews, static code analysis, manual security testing, and tool-based security testing
- Which software development standards or procedures does the CSP use for design and coding during Secure Development Life Cycle?
And an important question associated to isolation (both process isolation and virtual machine isolation):
- How an application (and its components) belonging to a tenant is protected from an attack from other tenants?

The security checklist for Infrastructure as a Service:
- How does the platform resist against Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks "at the infrastructure and network layers"? What mechanisms are used for this purpose?
- What threat models are addressed "at the infrastructure and network layers"?
- How does the infrastructure validates the integrity of a virtual machine image?
- How does the infrastructure resist against root kit and BIOS types of attacks (which are among the toughest attacks to conquer)? Does any mechanism exist to detect and remove such kind of attacks?

The security checklist for Platform as a Service:
- "Where is the "line of responsibility" drawn between the security of platform and application components?"
- Does any real-time IDS (Intrusion Detection System) exist on the platform for detecting application level intrusions?
- What tools does the platform provide for logging at application level?
- How does the platform secure communication channels between application components? How does it protect (isolate) data while at rest and in use?
- How does the platform protect message data on the client's service bus?

The security checklist for Software as a Service:
- Which security standards for web application are used by

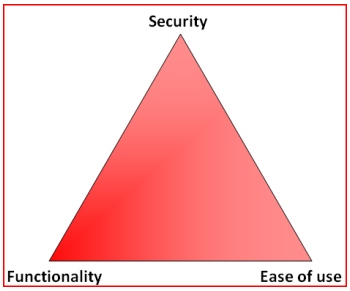


Fig. 6. Balance Triad

the CSP (Coud Service Provider) during the development of the software?
- Which "application and infrastructure controls" exist to protect one tenant's data from other tenant's data, while the data is in transit, in use, or at rest?

E. Balance Triad

As figure 6 illustrates and we mentioned earlier, there is a balance triad between security, ease-of-use, and functionality. When we are dealing with security in any stage, from security requirements specification to security assessment, we should have this balance in mind, especially if there is a competition in the market. The excessive complexity of system should not make the user to leave the service/product.

IV. Conclusion

Cloud security is neither a minor nor a simple matter. It is important and difficult to achieve. The security of a cloud application depends on two major category of factors: general cloud security factors and cloud application security factors. In order to develop and deploy a secure cloud application, we should incorporate security into the software development life cycle in early stages. In software requirement engineering phase, we should carefully define and establish cloud application security requirements. The requirements should have certain properties. Establishing a due diligence is an step toward the definition of the security policy which is a mean for defining security requirements and a measure for selecting the appropriate cloud service provider. We should follow secure design and coding principles (and tools) during the development process. The selection of cloud service provider is also important. With regard to the organization's security policy and requirements, we should assess different cloud service providers on the market against an evaluation checklist to see which one is more appropriate and more secure. In order to achieve general cloud security which is a prerequisite or complement to cloud application security, we should have general cloud security risks in mind and perform countermeasure. Patch management is a sample of general countermeasures both in cloud and non-cloud environments.

References

[1] Lin A. and Chen N., "Cloud computing as an innovation: Perception, attitude, and adoption," International Journal of Information Management, no.32, pp. 533-540, 2012

[2] M. Okuhara *et al.*, "Security Architectures for Cloud Computing," FUJITSU Sci. Tech. J., vol.46, no.4, pp. 397-402, Oct. 2010

[3] C. Rong *et al.*, "Beyond lightning: A survey on security challenges in cloud computing," Computers and Electrical Engineering, no.39, pp. 47-54, 2013

[4] *Handbook on Securing Cyber-Physical Critical Infrastructure*, Morgan Kaufmann, Waltham, MA, Feb. 2012, pp. 389-410

[5] L. Karadsheh, "Applying security policies and service level agreement to IaaS service model to enhance security and transition," Computers & Security, no.31, pp. 315-326, 2012

[6] Robert G. Carter. (Revisited: 1 April 2013), *Authentication vs. Authorization* [Online], Available: http://people.duke.edu/~rob/kerberos/authvauth.html (URL)

[7] B. Grobauer *et al.*, "Understanding Cloud Computing Vulnerabilities," IEEE Security & Privacy, pp. 50-57, Mar./Apr. 2011

[8] R. L. Krutz and R. Dean Vines, "Cloud Security: A Comprehensive Guide to Secure Cloud Computing," Wiley Publishing, Inc., pp. 61-121, 2010

[9] K. M. Goertzel and T. Winograd, "Enhancing the Development Life Cycle to Produce Secure Software," US. Department of Defense, pp. i-iv, Oct. 2008

[10] K. M. Goertzel *et al.*, "Software Security Assurance: A State-of-the-Art Report (SOAR)," US. Department of Defense, pp. 19-29, Jul. 2007

[11] Presidents Information Technology Advisory Committee, "Cyber Security: A Crisis of Prioritization," US National Coordination Office for Information Technology Research and Development, pp. 5-18, Feb. 2005

[12] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," Future Generation Computer Systems, No. 28, pp. 583-592, 2012

[13] S. Zander *et al.*, "a Survey of Covert Channels and Countermeasures in Computer Network Protocols," IEEE Communications Surveys & Tutorials, Vol. 9, No. 3, pp. 44-57, 2007

[14] X. Fu *et al.*, "Active Traffic Analysis Attacks and Countermeasures," in *Proc. Int. Conf. on Computer Networks and Mobile Computing*, pp. 1-9, 2003

[15] J. Arnold *et al.*, "Guidance for Application Security V2.1," Cloud Security Alliance, pp. 5-29, 2010, Available: http://www.cloudsecurityalliance.org/guidance/csaguide-dom10-v2.10.pdf [URL]

[16] V. Winkler, "Introduction to Cloud Computing and Security," in *Securing the Cloud*, Syngress, 2011

[17] A. A. Yassin *et al.*, "Anonymous Password Authentication Scheme by Using Digital Signature and Fingerprint in Cloud Computing," in 2nd Int. Conf. on Cloud and Green Computing, 2012, pp. 282-289

[18] The Open Web Application Security Project.(Revisited: 8 April 2013), *Document security-relevant requirements* [Online], Available: //www.owasp.org/index.php/Document_security-relevant_requirements (URL)

Appendix – Alphabetical List of All Seminar Papers

Paper	Author
Analysis of the Availability of Amazon Web Services' Cloud Infrastructure Services	Santeri Paavolainen
Cloud Computing and Social Networking Services Engineering Challenges and Solutions	Mohsen Koolaji
Cloud Provider Interoperability and Customer Lock-in	Mirva Toivonen
Cloud Security - Challenges and Requirements	Michael Przybilski
Cloud-based Testing: Opportunities and Challenges	Yanhe Liu
Comparing Preconditions for Cloud and On-Premises Development	Teemu Mattila
Comparision of Different Approaches to Evaluate Cloud Computing Services	Rakesh Pandit
Continuous Deployment of Software	Ville Pulkkinen
Decision Making About Migrating To The Cloud Model	Emad Nikkhouy
Feedback-Driven Development: A Mobile Case Study	Alexander-Derek Rein
Impact of Cloud Computing on Global Software Development Challenges	Inna Smirnova
Open Source Cloud Platforms	Jussi Hynninen
Quality of Service: Metrics and Evaluation in Cloud-based Services	Zinat Rasooli Mavini
Secure Cloud Application	Javad Sadeqzadeh Boroujeni
Secure Data Management for Cloud-Based Storage Solutions	Mikael Svenn
Success with a Cloud Software Ecosystem	Aikeremu Tiemuer

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI