Int J Software Informatics, Volume 7, Issue 2 (2013), pp. 221–237 Tel: +86-10-62661040
International Journal of Software and Informatics, ISSN 1673-7288 http://www.ijsi.org

Email: ijsi@iscas.ac.cn, ijsi2007@gmail.com

Using Byzantine Fault-Tolerance to Improve Dependability in Federated Cloud Computing

Peter Garraghan[1], Paul Townend[1], Jie Xu[1], Xiaoyu Yang[2] and Peipei Sui[2]

[1](School of Computing, University of Leeds, Leeds, UK)
[2](National Key Lab of Software Development Environment, Beihang University, Beijing, China)

Abstract Computing Clouds are typically characterized as large scale systems that exhibit dynamic behavior due to variance in workload. However, how exactly these characteristics affect the dependability of Cloud systems remains unclear. Furthermore provisioning reliable service within a Cloud federation, which involves the orchestration of multiple Clouds to provision service, remains an unsolved problem. This is especially true when considering the threat of Byzantine faults. Recently, the feasibility of Byzantine Fault-Tolerance within a single Cloud and federated Cloud environments has been debated. This paper investigates Cloud reliability and the applicability of Byzantine Fault-Tolerance in Cloud computing and introduces a Byzantine fault-tolerance framework that enables the deployment of applications across multiple Cloud administrations. An implementation of this framework has facilitated in-depth experiments producing results comparing the reliability of Cloud applications hosted in a federated Cloud to that of a single Cloud.
Key words: cloud computing; cloud federation; Byzantine fault-tolerance; dependability

Garraghan P, Townend P, Xu J, Yang XY, Sui PP. Using Byzantine fault-tolerance to improve dependability in federated cloud computing. *Int J Software Informatics*, **Vol.7, No.2 (2013): 221–237. http://www.ijsi.org/1673-7288/7/i160.htm**

1 Introduction

Modern computing systems are characterized by their requirement for substantial computing power. This has been augmented by an exponential increase in the volume of data available for processing. One potential approach to fulfill this requirement is to establish large scale computing systems, which typically require significant time and financial effort. Such systems are usually designed with respect to a maximum, least or average usage requirement, making the resultant system either under-used or unable to perform the desired function due to shortage of resources. Furthermore, such systems do not exhibit the ability to grow dynamically and require extensive design and development procedures to cope with evolving user requirements. This problem is aggravated when the user requirements are susceptible to unexpected changes. Therefore, contemporary computing systems are inflexible by nature and it is extremely difficult to guarantee on-demand availability for them in a cost effective manner.

The work is partly supported by the National Basic Research Program of China (973) (No. 2011CB302602), the UK EPSRC WRG platform project (No. EP/F057644/1), and the Major Program of the National Natural Science Foundation of China (No. 90818028).
Corresponding author: Peter Garraghan, Email: scpmg@leeds.ac.uk
Received 2012-11-14; Accepted 2013-05-07.

Cloud computing has emerged as a computing paradigm to facilitate the establishment of large scale, flexible computing infrastructures that are available on demand. It provides the opportunity to dynamically scale-up and scale-down the infrastructure of an organization in accordance with the requirements of the users of that infrastructure mitigating the problems described above. According to the National Institute of Standards and Technology (NIST)[6], Cloud computing is "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider iterations". Such systems are typically characterized as large, complex and heterogeneous systems[1] which results in two distinct problems. First, an increase in the occurrence of hardware and software faults to a point where faults becomes a common occurrence rather than an exception, resulting in the reduction of system reliability and subsequently consumer Quality of Service (QoS). Secondly, as the usage and storage capacity of a Cloud grows, there are increasing limits to the resource capacity and scaling a single Cloud can provide to consumers[4]. A proposed solution to these problems is to federate multiple Clouds together, in order to mask potential inconsistencies in a single failed Cloud as well as increasing the failure independence between nodes used by Cloud applications and extend the scalability of Cloud applications. Such research challenges have been highlighted in Ref. [32], who introduce the concept of the Internet-based Virtual Computing Environment.

However, presently there has been limited work to identify precise Cloud characteristics and how they affect the dependability and more specifically the reliability of Clouds environments. This has been highlighted in Ref. [29] which states that building highly reliable and available Clouds is a "critical, challenging and urgently-required research problem". This is particularly true when considering Byzantine faults that are arbitrary in nature and exhibit malicious behavior. The authors in Ref. [6] argue that the use of Byzantine fault-tolerance (BFT) is not relevant within the Cloud environment. Other authors such as Ref. [3] disagree with this notion, stating that BFT when applied to a federated environment removes many of the issues involved with applying BFT to a single Cloud.

This paper investigates and analyses the implications of Byzantine fault-tolerance in Cloud computing and introduces a Byzantine fault-tolerance framework that leverages federated Cloud infrastructures to provide highly dependable fault-tolerant applications. An implementation of this framework is discussed and detailed experimental results are provided and evaluated showing a significant dependability gain in federated Cloud environments compared to that of a single Cloud. The primary contributions of this paper is an implementation of a Byzantine fault-tolerant framework deployed within real world Cloud infrastructures to evaluate the effectiveness and performance of BFT in a single and federated Cloud environment in the presence of injecting various faults into these systems.

The remaining sections of this paper are structured as follows: Section 2 provides a background on Cloud federation and Cloud characteristics that significantly impact dependability research. Section 3 describes the threats to Cloud, Byzantine faults and the applicability of Byzantine fault-tolerance in Cloud environments. Section 4 describes related work. Section 5 describes the experimental set up and infrastructure

used. Section 6 presents the fault model. Section 7 presents the experimental analysis results. Finally, Section 8 presents the conclusions and discusses future work.

2 Background

2.1 Cloud federation

It has been proposed that Cloud environments are in the process of evolving from single, monolithic Clouds to that of a federation of Clouds[8]. The evolution of existing Cloud systems and markets to a federation of interoperable Clouds offers a number of advantages compared to that of single Clouds including extended scalability, enhanced interoperability, improved QoS, improved economies of scale and removal of Cloud isolation. The continued increase of enterprises wishing to take advantage of Cloud computing can be seen as a natural progression of the integration of private and public Clouds. It has also been forecasted that as Cloud systems mature, more sophisticated services shall emerge that will require multiple Clouds to function[12]. As Celesti[8] argues, it is unclear what exactly is meant by a federation within the context of Cloud computing. There is emerging literature that proposes the idea of a federation, under the guise of different names that are frequently used interchangeably by different authors as shown in Table 1. Although the terminology differs in precise semantic details, they are essentially describing the same concept. In this paper we outline the key characteristics of Cloud federation as the following:

- *Orchestration of multiple Clouds that are under different organizations and administration domains.* The key concept of Cloud federation is that all the participating Clouds are independent entities that are not controlled by a central entity that has administrative control of the Clouds within the federation. This idea of Cloud federation is different to that to that of multiple Clouds controlled by a centralized administrative domain such as Amazon EC2, which is separated into different availability zones that all conform to the developmental practices and jurisdiction of a centralized entity[13].
- *The enablement of portability and exchange of information.* Cloud federation enables the exchange of computational and storage resources that result in increased capacity available to a Cloud application. A practical example of this concept would periodic or unexpected resource demand driven by consumer behavior requiring resources from additional Clouds.

This paper focuses on Cloud federation environments which advocate client-centric protocols to orchestrate multiple Clouds as proposed by Cachin in Ref. [16], which represents feasible usage of Cloud federation by enterprises, leveraging a hybrid Cloud infrastructure. Moreover, Cachin et al. also propose that client-centric Cloud federation will continue to evolve to contain more sophisticated services involving communication across different Cloud services. This introduces the challenge of provisioning complex services reliably "across a federated network of possible disparate data centers" which has been identified by Ref. [18] as a difficult and unsolved problem.

Table 1 Cloud federation literature review

Terminology	Description
Intercloud	Leverage of a multitude of Clouds providers for the purpose of exchanging resources[15]. Can be client centric, with client-side proxies managing multiple Clouds[16].
Hybrid Cloud environments	Composition of two or more Cloud Computing environments that are of varying deployment models (public, private, hybrid etc.) as well as unique entities that allows data and application portability[7].
Interoperable Clouds	Utilization of Cloud computing services across multiple Cloud service providers[17].
Cloud federation	Two or more independent Cloud providers that are capable of sharing resources and are "able to scale applications across multiple domains to meet QoS targets of Cloud customers"[5,18].

2.2 Cloud dependability

Many of the new challenges faced by the Cloud computing community can be related to the concepts of dependability and security. Writing in Ref. [20], Randell defines dependability as "that property of a computer system such that reliance can justifiably be placed on the service it delivers. The service delivered by a system is its behavior as it is perceived by its users". It is important to state that this definition of dependability is not simply a synonym for reliability; rather, reliability is just one attribute of the overall concept.

Traditionally, dependability is a global concept, and subsumes the attributes of reliability, availability, safety, integrity, maintainability, and confidentiality. However, these attributes are becoming increasingly differentiated; for example, Ref. [21] distinguishes between dependability and security attributes - as shown in Fig. 1 - in order to highlight the main balance of interest given to these attributes by their respective communities.

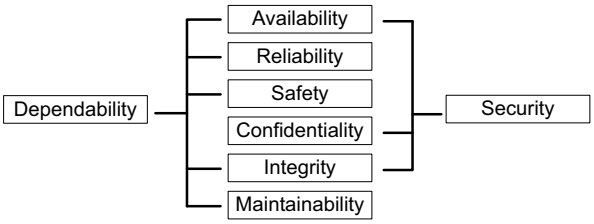

Figure 1. Dependability and security attributes

Dependability in Clouds is a key concern, as there are potentially great economic consequences for any failures; additionally, these failures are increasingly common due to the large scale of many Clouds[6]. For example in 2009, Amazon's EC2 Cloud launched over 50,000 instances per day[23]. As of 2011, the Amazon Cloud contains more than 449 billion objects and processes up to 290,000 requests per second at peak time[24] and these figures are predicted to continue to increase.

Cloud computing contains a large number of different deployment (Public, Private, Hybrid, etc.) and services models (SaaS, PaaS, IaaS, etc.). Furthermore, it is possible for two Cloud systems that share the same deployment and service model

to exhibit vastly different system environmental behavior due to the physical deployment, user behavior and business requirements of consumers within each Cloud. Due to this broad specification of Cloud environments, it is difficult and confusing to precisely describe unique challenges Cloud dependability research. As a result, it is important to relate challenges in dependability research to the characteristics which Cloud computing environments exhibit, which are summarized in Table 2.

- *Massively scalable and complex architectures.* Many Cloud computing systems heavily leverage virtualization technologies to enable massive scaling of Virtual Machines (VMs). Combined with the rapid growth and increased demand of Cloud computing services as well as federated environments becoming increasingly popular, such systems result in the creation of massive and complicated systems. It is well established that increasing the size and of a system results in the increased complexity in system management and architecture, and consequently increased occurrence of faults and failures.

- *Multi-tenant and high consumer QoS diversity and expectation.* Clouds typically are of a multi-tenant nature, with each consumer pursuing heterogeneous business objectives and QoS expectations that a provider must fulfill[7]. This heterogeneity results in a variation of dependability requirements between consumers. Furthermore, consumers expect high QoS expectations[31] from Cloud infrastructure to provision service equivalent and sometimes greater than their own private infrastructure.

- *Wide diversity of dynamic workloads.* Diversity of workload in terms of resource consumption and VM size is a result of unique consumer business objectives. Furthermore, Cloud workload size and resource consumption can reactively change due to leveraging virtualization technology. This behavior is a concern for dependability research as it has been shown that different workload properties such as computationally and I/O intensive workloads influence the type of faults that are activated[1] resulting in a larger number of potential faults and failures that providers have to address within their systems.

Table 2 Cloud characteristics and dependability challenges

Cloud Characteristic	Dependability Concern
Massive scale and complex architectures	Increase in size and complexity of system as well as rapid growth and interoperability of Cloud systems results in more frequent faults and failures.
Multi-tenant and high consumer QoS expectation	Requirement for Cloud infrastructure to provision attributes of dependability at a high level; potential conflicting QoS concerns.
Diverse workload dynamicity	Cloud scalability and generality of workloads require the Cloud to handle different types of faults and failures as a result of different workloads cause different types of faults to activate.

3 Cloud Threats

There are a number of faults that can reduce the reliability of Cloud environments. such as crash faults that manifest fail-stop failures in VMs and servers as well as network faults that cause disconnection or delayed service[29]. These types of faults can manifest due to the dynamicity of workloads within a Cloud infrastructure, as different workloads will result in different failures occurring. These failures result in reduced QoS for consumers, and in the result of catastrophic failure results in Cloud outages. Fault-tolerance can be deployed to increase the reliability of Cloud services and the system as a whole.

3.1 Byzantine faults

An example scenario would be for an attacker to instill arbitrary (Byzantine) behavior into a compromised VM and then commence a Denial-of-Service (DOS) attack affecting the other services on the same server and Cloud system[33]. Cloud services must be designed under the assumption that Clouds experience frequent and unpredictable failures[6], some of which will not fail gracefully as seen in recent outages.

It is therefore highly desirable to design a system that can reduce the likelihood of Byzantine faults affecting the overall dependability of applications running in Clouds; one method of achieving this is through Byzantine Fault-Tolerance (BFT). BFT is a well-established topic in the field of fault-tolerant research and is the application of the Byzantine general's problem[29] within a system, wherein a system can still achieve consensus and tolerate at most a third of its components behaving in an arbitrary manner. BFT is typically achieved through the use of diversity; multiple applications (either copies or different designs) are executed and their results are sent to an adjudication system which can use a variety of algorithms, typically application specific, to decide upon a correct result. A key strength of BFT is that it makes no assumption about the behavior of a faulty process, enabling the system to tolerate both consistent and inconsistent failures[21]. This is a particular challenge, as a large number of fault tolerant techniques focus on a protocol's ability to tolerate a set of faults that fail gracefully and assume components in a system fail by either stopping or omitting; this does not allow tolerance of faulty nodes that exhibit arbitrary behavior caused by malicious attacks, operator mistakes, or certain software errors which can result in arbitrary failures.

3.2 Byzantine faults in the Cloud

Current thinking such as Birman et al.[6], has been that Byzantine fault-tolerance may not be a critical research agenda in traditional, single Cloud computing environments, citing the following reasons:

- *An unsuitable threat model.* Applying Byzantine fault tolerance on a single Cloud to achieve consensus could potentially couple the behavior of multiple nodes, threatening the dependability of the entire Cloud system, termed 'fear of synchronization'. Providers instead choose fault prevention and detection, preferring to protect and isolate the critical Cloud component from the outside world instead of using fault-tolerance.

- *Failure independence.* Byzantine fault tolerance works under the assumption that there is strong failure independence. For stronger failure independence, the Cloud would have to be designed by different teams, deploy different architectures as well as technologies and at geographical locations. As a result, building and maintaining diverse Cloud scale infrastructure sufficient for the application of BFT would be expensive in terms of development and maintenance costs.

- *Expensive replica cost.* Byzantine fault-tolerance requires larger replica of software (in the context of Cloud VMs) which results in an increased cost. This cost in terms of resources and performance is greater than that of other FT schemes which can affect the availability of a single Cloud.

- *Single point of network failure.* Single Clouds still experience single point of failure on a network, especially if the Cloud provider is not geographically diverse[16].

However, this situation changes when considering a federated Cloud environment. Indeed, there are a number of benefits in applying BFT in federated Clouds which nullify many of the problems of applying BFT to single Clouds:

- *Cloud federation offers unprecedented levels of failure independence to a Cloud consumer.* Different Clouds within a federation are developed using different hardware, virtualization technology, geographical locations, development teams, development architectures, power supplies etc. This reduces the chance of a single point of failure for an application which is more likely to occur from using a single Cloud.

- *The threats to federated Cloud are more suitable than that of a single Cloud.* Potentially critical components of a Cloud application are now potentially accessible from anywhere as opposed to a node running deep within a protected environment within an inner Cloud infrastructure.

4 Related Work

At present time, there is limited literature regarding Byzantine fault-tolerance in Cloud computing systems. Birman et al.[6] discuss the applicability of BFT and consensus in Cloud computing, as well as outlining ideas and research opportunities for researchers. Reference [3] is a similar paper to that of Birman et al., arguing a stronger case for the applicability of BFT in federated Cloud, as well as a paradigm shift of reliability work in federated Clouds and outlining future research ideas. Our work provides experimental work and results in comparing the dependability gains in a federated Cloud against that of a single Cloud. Zhang et al.[30] developed a BFT framework for voluntary Cloud computing infrastructure. Voluntary Clouds are unlike Clouds that are well-provisioned and managed by a large (typically enterprise) groups such as Amazon, Microsoft, Taskforce etc., and instead are akin to Clouds composed of user contributed computing resources. Our work aims to also encompass well-provisioned and well-managed Cloud infrastructures. Guearroiu et al.[33] provides a

high level reevaluation of BFT protocols in various Cloud deployments by measuring performance metrics, however the paper does not approach the work from a formal dependability perspective, nor does it contain an experiment framework.

There are a number of federation models proposed, focusing on a number of dependability attributes such as availability[4] and other attributes such as interoperability, performance and security. Our federation model aims to improve the reliability of client-centric Cloud services.

5 Fault-Tolerant - Federated Cloud

5.1 Research objectives

In order to explore and assess the improvement in reliability of Byzantine fault-tolerance in a single and federated Cloud, we have implemented a framework called "Fault-Tolerant - Federated Cloud" (FT-FC). This framework consists of several components, all of which are programmed in Java. It includes several features to facilitate and encourage the use of Byzantine fault-tolerance in federated Cloud applications:

- An automatic job scheduling and Cloud application invocation tool that allows Cloud applications to be automatically submitted and invoked in multiple heterogeneous Clouds, running either Xen or KVM hypervisors.
- A messaging system based on the SSH protocol (i.e. ssh and scp) that allows communications between a Cloud and FT-FC to be performed securely; this tool can be embedded into Cloud applications to facilitate cross-cloud communication.
- An adjudication system to decide whether a Cloud application is exhibiting intolerable faulty-behavior that can send and receive data from multiple Clouds.

FT-FC allows the creation of many different forms of redundant fault-tolerant algorithms. Through the use of the framework we have performed work to assess the feasibility and effectiveness of incorporating BFT into federated Cloud applications and compared these results against that of a single Cloud environment.

We have used FT-FC to perform a series of experiments that involve invoking a real Cloud application within a federated Cloud. The Cloud application is based on the GENESIS e-Social Science project[9], and consists of a program that generates a virtual representation of a population and then performs various analyses on that population. It returns a series of aggregate values based on those analyses back to the user. The GENESIS application was instrumented in order to allow us to inject a variety of Byzantine faults into its processing (specifically value, omission and late timing faults) Experiments were performed to evaluate the reliability and performance of provisioning a Cloud application within a single and federated Cloud environment in the presence of injected faults.

5.2 Experiment environment

The experiment was set up into two parts. First, the GENESIS application was deployed on six different VMs in a single Cloud datacenter and were invoked using

the FT-FC. Secondly, the experiment setup was repeated but instead deploying six VMs evenly across two Cloud systems that form a federation as shown in Fig. 2. For each experiment a number of faults were injected into each VM at run time. Within the experiment, it was possible that a failure of a VM could potentially propagate to other VMs residing within the same infrastructure.

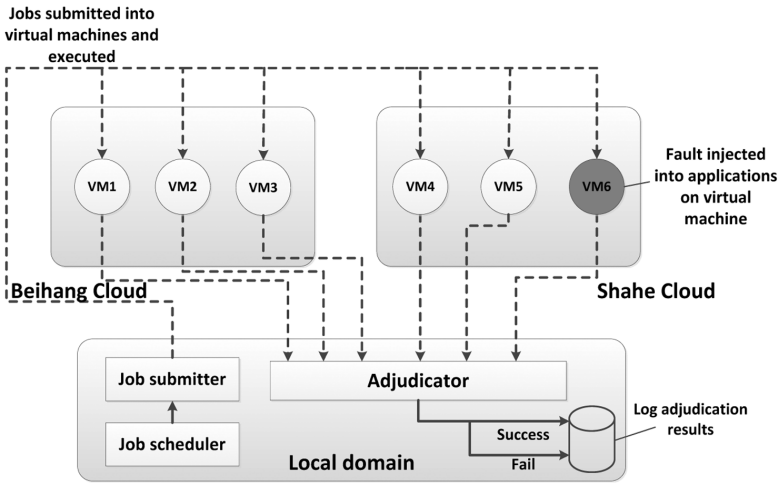

Figure 2. The experiment setup for federated Cloud application invocation with fault injection

5.3 Experiment setup

Both experiments were performed on two real word Clouds; Beihang Cloud and Shahe Cloud. Beihang Cloud is formed using 32 x HP BL460C blade servers, with each server consisting of 2 x Intel E5405 2.0GHz and 16GB memory. Shahe Cloud is composed of 32 x DELL 6100 machines, with each machine containing 2 x Intel Xeon E5620 and 48 GB memory. These Clouds are situated at geographically diverse locations. Furthermore, VMs were deployed on three different operating systems; Debian Squeeze, Fedora and CentOS. The effectiveness of BFT is based on the premise of failure independence[23]. One of the ways to achieve this is by design diversity, in which replicas within the systems are independently designed[30]. The diversity mechanism selected for this system is an N-Replica system. N-Replica is similar to that of N-version design systems but instead of design diversity implements data diversity[24]. Data diversity is based on the idea that a system can be divided into fault and non-fault regions, and the system only failing when the fault region is activated and applied[25]. Data diversity means a diversity of data inputs, which means portion of data inputs from a fault region can be tolerated. Although we are aware of the increase in dependency of failures in an N-Replica system compared to that of an N-Version system for the purposes of this experiment, which is an analysis of the reliability of a single federated Cloud there is a sufficient diversity in the respective Cloud datacenters.

In both experiments the Cloud application composed of six N-Replica VMs are invoked on a single Cloud and federated Cloud. The result calculated from each

application are then sent from the VMs to the adjudicator which waits to receive a sufficient amount of results to achieve consensus, with the maximum waiting time of 14 seconds. Within this time, one of three possible faults can be injected into VMs with a probability to propagate across the system. If adjudication is successful, the collected values sent from the VMs are averaged to give a correct result to the consumer. If the adjudicator fails, then all returned values are discarded for that invocation. The reason for this is because in many scenarios, especially mission critical systems, it is better for an adjudication to fail than take the risk of sending incorrect results. The Cloud application was invoked Cloud 1000 times for each experiment.

6 Fault Model

Three different types of failure were modeled for the experiments: value faults, omission faults, and late timing faults. Omission faults cause the data submitted from a service to never arrive at the adjudicator, violating the maximum adjudicator time boundary. Timing faults cause the data submitted from the service to arrive at the adjudicator outside of the specified time domain. This varies from the omission fault in that the adjudicator acknowledges that the data from a VM arrives in the adjudication phase. Finally, value faults returned to the adjudicator lie outside the valid value boundaries.

6.1 Failure propagation model

The weaker failure independence is between nodes, the more likely there will be a propagation of failures. We model a suitable failure propagation model for VMs residing within a single and federated Cloud environment. As a result of a lack of experimental data concerning the nature of fault propagation within Cloud environments, the formalisation derived from Haibin[26] is used to build a failure propagation model.

Failure propagation is characterised by failed components within a system causing other components to subsequently fail within the same system environment[26]. The probability of propagation P occuring between VMs i and j within a single Cloud C_a is shown in (1)

$$P = (Ca_i Ca_j; i \neq j; Ca_i Ca_j \neq 0;) \tag{1}$$

$$PCa_i Ca_i \rightarrow \infty \tag{2}$$

Furthermore, it is not possible for a failed VM to propogate a failure back to itself as it has already failed as shown in (1) and (2). P can be represented as a probability matrix for all possible connections of VMs within Cloud C_a as shown in (3).

$$P = \begin{vmatrix} PCaCa_1 & PCa_2Ca_1 & \dots & PCa_jCa_1 \\ PCa_1Ca_2 & PCa_2Ca_2 & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ PCa_1Ca_i & \dots & \dots & PCa_iCa_j \end{vmatrix} \tag{3}$$

For example, $P(C_{a2}C_{a4})$ represent the propability of failure propagation of VM 2 to VM 4 within Cloud a. Figure 3 presents a failure propagation scenario within Cloud C_a. In this scenario, node i within the system has the potential to propagate failures to four other VMs within the same infrastructure. We can also use (3) to describe the chance of failure propagation in a federated Cloud environment composed of Clouds a and b as shown in (4).

$$P = (Ca_i Cb_{j)} \tag{4}$$

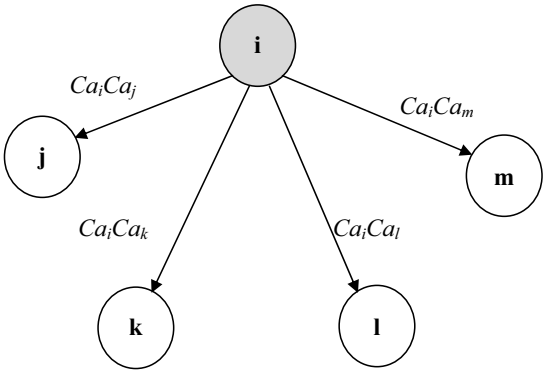

Figure 3. Example scenario of failure propagation within Cloud C_a

The failure propagation model described in (1) is implemented in both experiments when a failure occurs within a VM as a result of one of the three faults injected, described in the previously.

6.2 Failure rate model

VM Mean Time to Failure (MTTF) was modelled using a Weibull distribution. Weibull distributions are one of the most widely used methods of modelling failure time distributions within a system to calculate the MTTF as shown in (5). As a result of being able to collect the completion time t of applications for the respective different types of faults injected, a two parameter Weibull was selected for both experiments as a result of being able to calculate the average Cloud application execution time.

$$R(t) = e^{-(t/\theta)^{\beta}} \tag{5}$$

An initial run of the experiment was performed to collect the average time of execution of the Cloud application and adjudication as shown in Table 3 to derive the appropriate value of θ. Each of the individual VMs follow their own failure distribution derived from (5). A scale parameter of $\beta = 1.5$ was assigned to all failure distributions. A scale value $\beta > 1$ indicates that the failure rate increases with time. This is a reasonable assumption as it is known that software run over longer periods of time are more prone to software failures later in their life cycles due to software aging. For the experiment a $2n + 1$ fault tolerance scheme was implemented, as this emulates that the protocol employs signed messages, enabling consumers to be able to discern the origin of the results sent.

Table 3 Preliminary results for adjudication completion time

Number of application invocations	Average time of application completion (milliseconds)
100	12600

6.3 Byzantine faults in the Cloud

The experiment is defined by a number of assumptions. The local domain, shown in Fig. 2, including the job scheduler and invoker the adjudication process as well as the communication network between the consumer domain and the Cloud are all non-faulty. In addition, the recovery time of a VM has been defined as instant, with the assumption that a VM is repaired before the next invocation values selected for the VM failure distributions are a combination of numbers derived from initial experiment results as well as making reasonable assumptions. As Ref. [11] claims, some figures given for their MTBF and failure rates are arbitrary but reasonable chosen parameters, given the difficulty and restriction on discovering accurate failure rates of these parameters.

Due to the diversity of geographical location, administration and hardware specification between the two Cloud systems, we have assumed that the failure dependence between Cloud services residing on the same Cloud infrastructure (which include the same network, physical machine, design team etc.) is lower than that of the failure dependency of services residing in different Clouds as shown in (6). This assumption reflects work stating that the use of Cloud federation results in an increased failure independence of Cloud services[3].

Due to the diversity of geographical location, administration and hardware specification between the two Cloud systems, we have assumed that the failure dependence between Cloud services residing on the same Cloud infrastructure (which include the same network, physical machine, design team etc.) is lower than that of the failure dependency of services residing in different Clouds as shown in (6). This assumption reflects work stating that the use of Cloud federation results in an increased failure independence of Cloud services[3].

$$P = (Ca_i Cb_j) \approx 0 \tag{6}$$

A Cloud VM failure can propagate to other VMs within the same infrastructure a maximum of one iteration with a probability of 10% per invocation.

7 Results Analysis

Table 4 presents the success of adjudication within single and federated Cloud environment when faults are injected into the Cloud application. We observe that the adjudication success for single Cloud lies between 91.7% - 93.4% which is lower than that of the federated Cloud which ranges between 94.8% - 95.2%. Figure 4 shows the number of adjudications failures for single and federated Cloud environments over 1000 Cloud application invocations when omission, value and time faults were injected. These results demonstrate that the probability of successful adjudication of a Cloud application is greater within a federated Cloud environment compared to that of a single Cloud.

Table 4 Adjudication success rate

Type of fault injection	Single Cloud adjudication success %	Federated Cloud adjudication success %
Value	93.4	95.2
Time	92.4	95.4
Omission	91.7	94.8

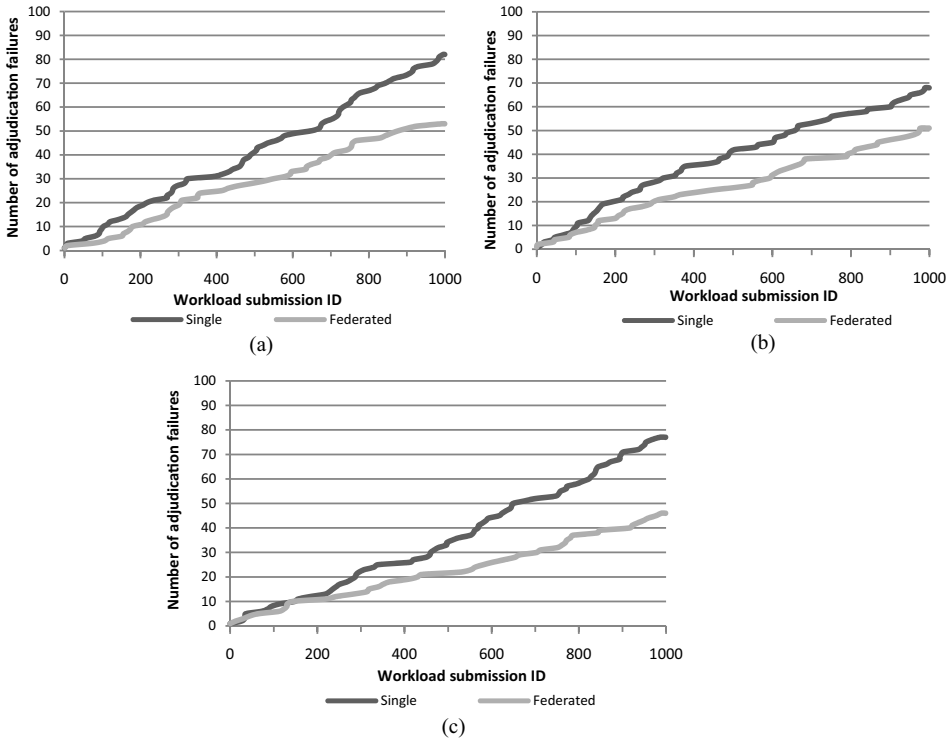

Figure 4. Experiment adjudication success comparison: (a) Omission, (b) Value, (c) Time

Another important value to measure is the performance, or more specifically the time taken to complete adjudicaton. The time taken to complete adjudication is higher within the single Cloud environment compared to the federated Cloud environment under the presence of omission as shown in Figs. 5(a) and 5(b) respectively. This is due to omission faults influencing adjudication completion time. We observe that when injecting omission faults the average adjudication completion time is equal to 12227 milliseconds for single Cloud and 12124 milliseconds for federated Cloud. These results support the results presented in Table 4 demonstrating that a federated Cloud environment enables the provisioning of more reliable service. Additionally, these results indicate that a system experiencing more frequent omission and timing failures result in a detrimental effect for dependability and performance within Cloud systems. More frequent omission failures result in an increased adjudication completion time, hence the use of a federated Cloud environment could potentially improve performance as well as reliability for Cloud

applications. From a consumer perspective, an omission or timing failure results in re-invoking the Cloud application, requiring additional time and resources allocated.

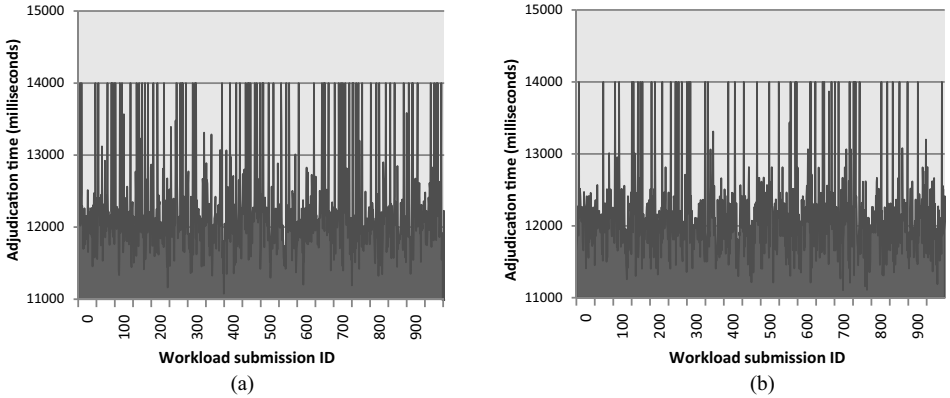

Figure 5. Adjudication time for Cloud application with injected Omission faults: (a) Single (b) Federated

Next, we analyzed the average application completion time for individual VMs within the Cloud environments. We observed that VMs which share the same OS but were deployed in different Clouds resulted in a noticeable difference in application completion time as shown in Fig. 6(a), which demonstrates that application execution time of the VM deployed on the Beihang Cloud is significantly slower than that of the VM on the Shahe Cloud. This is a result of interest as both VMs are running the same application as well as the same operating system (Debian Squeeze).

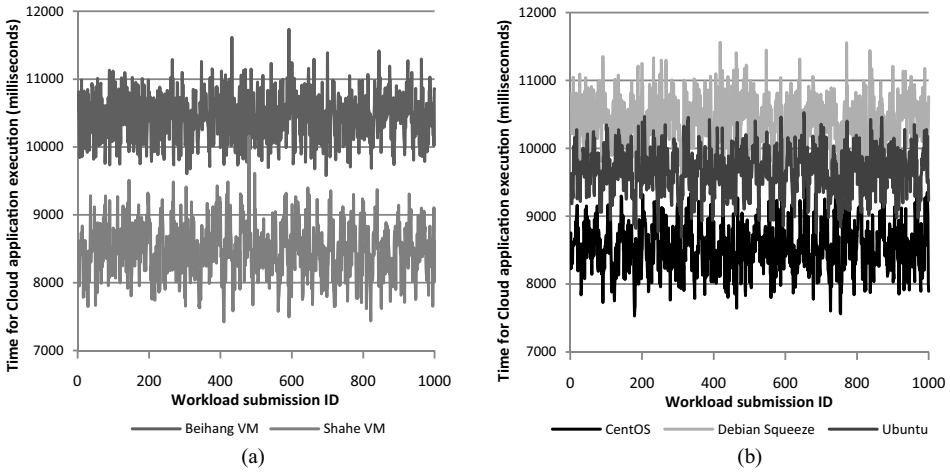

Figure 6. Adjudication completion time comparison: (a) Federated Cloud, (b) Single Cloud

Figure 6(b) presents the application execution time of three VMs using different operating systems within the same Cloud environment. It is observable that the application completion time for each VM varies depending on the type of OS, with CentOS and Debian Squeeze being the quickest and slowest respectively. This behavior is due to the underlying infrastructure the VM is deployed upon. Although

these results are specific to the experimental environment on which the VMs were deployed, it demonstrates a potential challenge of autonomous deployment of VMs in federated Cloud environments. On one hand, a consumer is unaware of the specific architectures within the Cloud environment when deploying Cloud applications while on the other, providers have a black-box view of the VM unable to ascertain the type of application or VM configuration the consumer is running. This divide can potentially result in uneeded timing failures occuring for consumers, which it further aggravated if the Cloud provider autonomously redeploys Cloud applications to other Cloud infrastructures within the federation without consumer interaction or acknowledgement.

8 Conclusion and Future Work

Building dependable and federated Cloud computing environments remains a critical research problem, and precisely what dependability challenges exist due to Cloud characteristics remains unclear. Byzantine fault-tolerance is a research area in which there has been disagreement on its applicability in Cloud environments. This paper presents experimental results comparing the reliability and performance of a single and federated Cloud environment. We have produced detailed results using real world Cloud systems that demonstrate using Byzantine fault tolerance in a federated Cloud environment is not only feasible and practical but also offers increased dependability for Cloud applications by taking advantage of the inherit failure independence of federated Cloud. Furthermore, we highlight how the heterogeneity of Cloud architecture and VM configurations introduce challenges in autonomous scheduling and migration of Cloud applications within a federated Cloud environment.

Future work includes applying more rigorous failure propagation modeling of VM failures within Cloud environments derived from analysis of production systems instead of assuming a theoretical probability. Furthermore, an interesting direction for this research includes further work concerning the challenges of autonomous scheduling and migration of VMs within federated Cloud environments composed of heterogeneous VM configurations, Cloud hardware and network architectures.

Acknowledgement

We would like to thank Beihang University for allowing access to their Cloud infrastructure for experimentation. The work in this paper has been supported part by the National Basic Research Program of China (973) (No. 2011CB302602), the UK EPSRC WRG platform project (No. EP/F057644/1), and the Major Program of the National Natural Science Foundation of China (No. 90818028).

References

[1] Fadishei H, Saadatfar H, Deldari H. Job failure prediction in grid environment based on workload characteristics, Computer Conference, 2009. CSICC 2009. 14th International CSI. 20–21 Oct. 2009. 329–334.

[2] Vukolić M. The byzantine empire in the intercloud. SIGACT News 41. 2010. 105–111.

[3] Rochwerger B, Breitgand D, Levy E, Galis A, et al. The reservoir model and architecture for open federated cloud computing, in: Internet and Enterprise scale Data Centers. IBM Journal

of Research and Development, 2009, 53(4): 1–11.
[4] Buyya R, Ranjan R, Calheiros RN. InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services. Proc. of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010). Busan, South Korea. Springer: Germany. 21–23 May 2010. 328–336.
[5] Birman K, Chockler G, van Renesse R. Toward a cloud computing research agenda. SIGACT News, 2009, 40(2): 68–80.
[6] Mell, Grance T. The NIST Definition of Cloud Computing. National Institute of Standards and Technology. Information Technology Laboratory. 2009
[7] Celesti A. How to Enhance Cloud Architectures to Enable Cross-federation. Proc. of The 3rd IEEE International Conference on Cloud Computing (IEEE Cloud 2010). Miami, Florida, USA. 2010. 337–345.
[8] Birkin M, Townend P, Turner A, Wu B, Arshad J, Xu J. MoSeS: A Grid-enabled spatial decision support system. Social Science Computing Review. DOI: 10.1177/089443930 933229. 2009.
[9] Kim DS, Machida F, Trivedi KS. Availability modeling and analysis of a virtualized system, dependable computing. PRDC '09. 15th IEEE Pacific Rim International Symposium on. 16–18 Nov. 2009. 365–371.
[10] Barr J, Narin A, Varia J. Building Fault-tolerant Applications on AWS? White Paper, Amazon. October 2011.
[11] Abawajy J. Determining service trustworthiness in Intercloud computing environments, pervasive systems, algorithms, and networks (ISPAN). 2009 10th International Symposium on. 14–16 Dec. 2009. 784–788.
[12] Cachin C, Haas R, Vukolic M. Dependable storage in the Intercloud. Research Report RZ 3783, IBM Research. Aug. 2010.
[13] Interoperable Clouds, A White Paper from the Open Cloud Standards Incubator. Distributed Management Task Force, Version 1.0. DMTF Informational. Nov. 11, 2009. DSP-IS0101.
[14] Rochwerger B. Reservoir - When One Cloud Is Not Enough. Computer, 2011, 44(3).
[15] Randell B. Dependability – its attributes – impairments and means. Predictably Dependable Computing Systems. Springer-Verlag. 1995.
[16] Avizienis A, Laprie J, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. on Dependable and Secure Computing, January 2004, 1(1): 11–33.
[17] Thorsten. Amazon Usage Estimates Internet: http://blog.rightscale.com/2009/10/05/ amazon-usage-estimates/Oct 5. 2009.
[18] Amazon S3 - More Than 449 Billion Objects Internet: http://aws.typepad.com/aws/2011/07/ amazon-s3-more-than-449-billion-objects.html July 19. 2011.
[19] Zheng Z, Zhou T, Lyu M, King I. FTCloud: A Component Ranking Framework for Fault-Tolerant Cloud Applications. IEEE 21st International Symposium on Software Reliability Engineering. Nov. 2010. 398–407.
[20] Castro M, Liskov B.Practical Byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst., 2002, 20(4): 398–461.
[21] Ristenpart T, Tromer E, Shacham H, Savage S. Hey, you, get off of my cloud: exploring imation leakage in third-party compute clouds. CCS '09: Proc. of the 16th ACM Conferon Computer and Communications Security. ACM. New York, NY, USA. 2009. 199–212.
[22] Guerraoui R, Yabandeh M. Independent faults in the cloud. LADIS '10: Proc. of the 4th ACM SIGOPS/SIGACT Workshop on Large-Scale Distributed Systems and Middleware. 2010. 12–16.
[23] Avizienis A. The N-version approach to fault-tolerance software. IEEE Trans. Software Eng., 1985, SE-11: 1491–1501.
[24] Ammann PE, Knight JC. Data diversity: an approach to software fault tolerance. IEEE Transactions on Computers, April 1988, 37(4): 418–425.
[25] Hawthorne M, Perry D. Applying design diversity to aspects of system architectures and deployment configurations to enhance system dependability. WADS'04. June 30, 2004.
[26] Haibin Y. Development of simulation model based on directed fault propagation graph.

International Conference on Computer Application and System Modeling (ICCASM), 22–24 Oct, 2010, 3: 686–690.

[27] Sahoo RK, Squillante MS, Sivasubramaniam A, Zhang Y. Failure data analysis of a large-scale heterogeneous server environment. Dependable Systems and Networks, 2004 International Conference on. 28 June-1 July 2004. 772–781.

[28] Zhang Y, Zheng Z, Lyu MR. BFTCloud: A Byzantine Fault-tolerance Framework for Voluntary-Resource Cloud. 2011 IEEE 4th International Conference on Cloud Computing.

[29] Lamport L, Shostak R, Pease M. The Byzantine Generals Problem. ACM Trans. on Programming Languages and Systems (TOPLAS), July 1982, 4(3): 382–401.

[30] Avizienis A, Kelly JP. Fault tolerance by design diversity - Concepts and experiments. Computer, Aug 1984, 17: 67–80.

[31] Buyya R. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst., 2009, 25: 599–616.

[32] Lu X, Wang H, Wang J, Xu J, Li D. Internet-based Virtual Computing Environment: Beyond the datacenter as a computer. Future Generation Comp. Syst. 2013, 29(1): 309–322.