Proceedings of The 6th IEEE International Symposium on Service Oriented System Engineering (SOSE 2011)

Byzantine Fault-Tolerance in Federated Cloud Computing

Peter Garraghan, Paul Townend, Jie Xu

{scpmg, p.m.townend, j.xu} @ leeds.ac.uk

School of Computing, University of Leeds, LS2 9JT, UK

***Abstract* — Cloud computing has emerged as popular paradigm that enables the establishment of large scale, flexible computing infrastructures that can offer significant cost savings for both businesses and consumers by allowing compute resources to be scaled dynamically to deal with current or anticipated usage [1]. This concept has been further strengthened with the emergence of federated computing Clouds that allow users to scale applications across multiple domains to meet Quality of Service targets [2]. However, the challenge of building dependable and robust Clouds remains a critical research problem that has not yet been clearly understood [3], and yet is vital for establishing user confidence in Clouds. This is particularly true when considering Byzantine faults that are arbitrary in nature. This paper analyses the application of Byzantine fault-tolerance to federated Clouds in detail, and presents experimentation performed to analyse the effectiveness of Byzantine fault-tolerance in federated Clouds. We have developed a Cloud framework called FT-FC that allows us to very quickly create diversity-based Byzantine fault-tolerant systems and apply them to federated Clouds, and have produced initial results to demonstrate the feasibility and potential of this approach. We have furthermore identified a number of research problems and challenges that need to be addressed in order to progress this area further. Our current experimental results, although very initial, are highly encouraging figures, and demonstrate the effectiveness of the FT-FC framework.**

***Index Terms*— Cloud federation, Cloud computing, dependability, Byzantine fault-tolerance**

I. INTRODUCTION

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 an 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. However, as usage of compute and storage Clouds grow, there are increasingly limits on how much resource scaling a single Cloud can provide [4]; a proposed solution to this problem is to federate multiple Clouds together, in order to extend the scalability of Cloud systems [2]. As Celesti et al. [5] argue, it is unclear what exactly is meant by a federation in a Cloud context. 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 [2][6][7][8]. These papers describe very similar concepts using different terminology. We have outlined two main principles that they share.

- *Orchestration of multiple Clouds that are under different organisations 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 over all the Clouds within the federation. This idea of Cloud federation is different to that to that of multiple Clouds controlled by a centralised administrative domain such as Amazon EC2, which is separated into different Availability zones that all conform to the developmental practices and jurisdiction of a centralised entity [9].
- *The enablement of portability and exchange of information.* Cloud federation enables the exchange of computational or storage resources that result in increased capacity of a Cloud application. A practical example of this concept would be Cloud bursting [27], which is when a single Cloud is unable to provision additional resources at peak time, and acquires additional resources through the leverage of an additional independent Cloud to meet consumers quality of Service (QoS) requirements.

For this paper, we choose to lean towards the idea of Cloud federation that [7] proposes, which advocates client-centric protocols to orchestrate multiple Clouds. This model has been predicted to reflect most enterprises over the next few years looking to utilise hybrid Cloud infrastructure [10]. 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, which introduces new challenges such as the provisioning of complex services reliably "across a federated network of possible disparate data centers" being a difficult and unsolved problem [8].

A previously unexplored aspect of federating multiple Clouds is the increased potential for using fault-tolerant techniques to increase the dependability of applications spread across them, particularly with regard to Byzantine fault-tolerance. This paper analyses this topic in detail, and actual experimentation has been performed to test the effectiveness of Byzantine fault-tolerance in federated Clouds.

II. BYZANTINE FAULTS IN CLOUD COMPUTING

Many of the new challenges faced by the Cloud computing community can be related to the concepts of dependability and security. Writing in [11], Randell defines dependability as

978-1-4673-0412-2/11/$26.00 ©2011 IEEE

"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 behaviour 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.

Dependability in Clouds is a key concern, as there are potentially great economic consequences for any failures [12]; additionally, these failures are increasingly common due to the large scale of many Clouds [13]. For example, in 2009, Amazon's EC2 Cloud launched over 50,000 instances per day [14]. As of 2011, the Amazon Cloud contains more than 449 billion objects and processes up to 290,000 requests per second at peak time [15] with these figures predicted to continue increasing.

Byzantine faults (such as sending inconsistent values to requests [16]) are malicious arbitrary faults that do not fail gracefully. These faults can be caused from malicious attacks, operator errors, or software errors [17]. In Cloud, there is increased concern over virtual machine security, from potential compromise of credentials to access a virtual machine (VM), and issues relating to Cloud multi-tenancy which can result in malicious attacks [18]. An example scenario would be for an attacker to instill arbitrary (Byzantine) behavior into a compromised VM, and then commence a DOS attack affecting the other replicas and services on the same Cloud. Cloud services must be designed under the assumption that Clouds experience frequent and unpredictable failures [13], some of which will not fail gracefully, as seen in recent outages [19].

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 within a system, wherein a system can still achieve consensus and tolerate at most a third of its components behaving in an arbitrary manner [20]. BFT is typically achieved through the use of diversity; multiple applications (either copies or different designs) are executed, and their results sent to an adjudication system which can use a variety of algorithms, typically application specific, to decide upon a correct result. An example of such a scheme is shown in figure 1, which shows an N-Version Design system capable of handling Byzantine faults.

Current thinking, such as Birman et al.[13], has been that Byzantine fault-tolerance may not be a critical research agenda in single Clouds systems, citing the following reasons:

- *An unsuitable threat model.* Applying Byzantine fault-tolerance on a single Cloud to achieve consensus could potentially couple the behaviour of multiple nodes, threatening the dependability of the entire Cloud system, termed 'fear of synchronisation'. Providers instead choose fault prevention and detection, preferring to protect and isolate the critical Cloud component from the outside world instead of using BFT [21].

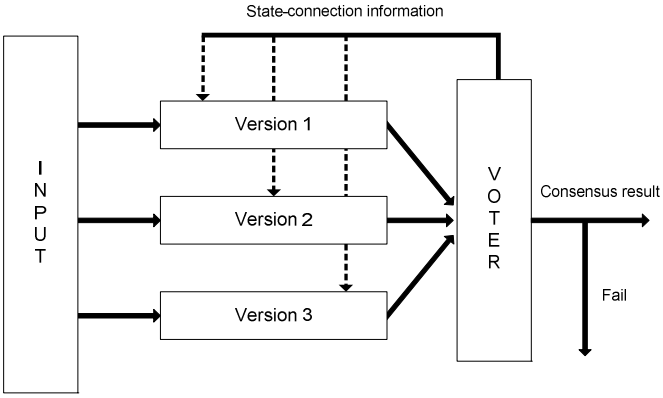

Figure 1. An n-version design system

- *Failure independence*. Byzantine fault-tolerance works under the assumption that there is strong failure independence between nodes within the system [16]. For stronger failure independence, the Cloud would have to be designed and developed by different teams, deploy different architectures as well as technologies and geographical locations. Building and maintaining Cloud scale infrastructure diversity BFT would be expensive. Single Clouds still experience a single point of failure on a network, especially if the Cloud provider is not geographically diverse [7].
- *Byzantine fault-tolerance requires larger replica cost*, compared to that of other fault-tolerant schemes, which will affect the availability of a single Cloud.

However, this situation changes when considering a federated Cloud model. 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 at no additional cost to a single Cloud provider or Cloud consumer*. This is due to different Clouds within a federation utilising different hardware, virtualisation technology [22], geographical locations, development teams, development architectures, power supplies and so on.
- *The threats to federated Cloud are more suitable than that of a single Cloud.* 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.
- *Byzantine fault-tolerance can improve the integrity of data and computation* as well as offer a way to mask potential inconsistencies in Clouds [7].

For these reasons, there is great potential in applying BFT techniques to federated Clouds; until now, there has been limited literature on this subject, and little experimental work has been performed to assess the feasibility of this conjecture.

III. Current BFT work in Cloud

At present time, there is limited literature regarding Byzantine fault-tolerance in Cloud computing systems. Birman et al.[13] discuss the applicability of BFT and consensus in Cloud computing, as well as outlining ideas and research opportunities for researchers. [21] 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. Zhang et al. [16] 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. [23], and instead are akin to Clouds composed of user contributed computing resources [33]; Our work aims to also encompass well-provisioned and well-managed Cloud infrastructures. Guearroiu et al. [22] provides a high level reevaluation of BFT protocols in various Cloud deployments by measuring performance, however the paper does not approach the work from a formal dependability perspective, nor does it contain an experiment framework.

IV. Implementation and Experimentation

In order to explore the feasibility of Byzantine fault-tolerance in federated Cloud, we have implemented a framework called "Fault-Tolerant - Federated Cloud" (FT-FC). It includes several features to facilitate and encourage the use of Byzantine fault-tolerance in federated Cloud applications. These features are as follows:

- An automatic job scheduling tool that allows Cloud application jobs to be automatically submitted to 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.
- A fault-tolerant adjudication system that can send and receive communications from multiple Clouds in order to achieve consensus on returned results from Clouds which could potentially fail.

FT-FC allows the creation of many different forms of redundant fault-tolerant algorithms; through the use of the framework, we have performed initial work to assess the feasibility and effectiveness of incorporating BFT into federated Cloud applications.

We have used FT-FC to perform a series of experiments that involve submitting real Cloud jobs into a federated Cloud. The Cloud application chosen is based on the MoSeS e-Social Science project [24], 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 MoSeS 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), and FT-FC framework code was added to allow the application to communicate results back to the adjudication system. This application was installed on four separate virtual machines evenly spread across the two Clouds.

The two Cloud systems we federated together are also real world systems; iVIC and the University of Leeds Test Cloud. iVIC is a virtual Cloud computing environment which allows an organisation to manage, configure and deploy large heterogeneous virtualised computing resources [32]. iVIC uses the KVM hypervisor and Fedora OS virtual machine images. The University of Leeds Test Cloud is a computing environment that uses the Debian OS and the Xen hypervisor.

The effectiveness of BFT is based on the premise of failure independence [16]; one of the ways to achieve this is through diversity [25]. The diversity mechanism selected for this system is an N-Copy system; N-Copy is similar to N-version design systems (as shown in figure 1) but instead of design diversity, implements data diversity [26] whereby multiple copies of the same application are given slightly different input data. The decision to use N-Copy design in this experiment is a result of the inadequate time frame required for designing separate heterogeneous services programmed in different languages and structures, as well as employing different developers to design and implement such systems was felt to be out of scope for these initial experiments. Although we are aware of the increase in dependency of failure in an N-Copy system, the purpose of our experiments in this paper is primarily to explore the feasibility of offering BFT in a federated cloud context; we find N-Copy Programming thus acceptable for this paper's scope.

The adjudicator performs acceptance testing by averaging the results of the returned values from the services. If the adjudicator detects that one or more of the channels violates the user specification by being outside of acceptable boundaries as a result of failure of a service such as sending an incorrect value or a result arriving too early or too late, it disregards the result and flags the service as failed.

Adjudication is successful when there are at least $2n + 1$ services that do not violate either time or value domains, else it is flagged as a failed adjudication. The reason being that $2n + 1$ requires fewer replicas it messages from channels are signed (we know who sent the message), therefore increasing the availability of the Cloud. If a consensus is not reached then the adjudication marks the adjudication stage as failed. The reason for this is because it is more damaging to allow violations of a system specification from both a security and dependability perspective as opposed to accepting no results at all; this case is true especially in mission-critical applications and infrastructure.

Experiments were performed with two different deployments. Entire Cloud infrastructures with a varying chance of failing and Clouds with a probability of failure propagation to other services on the same Cloud. The

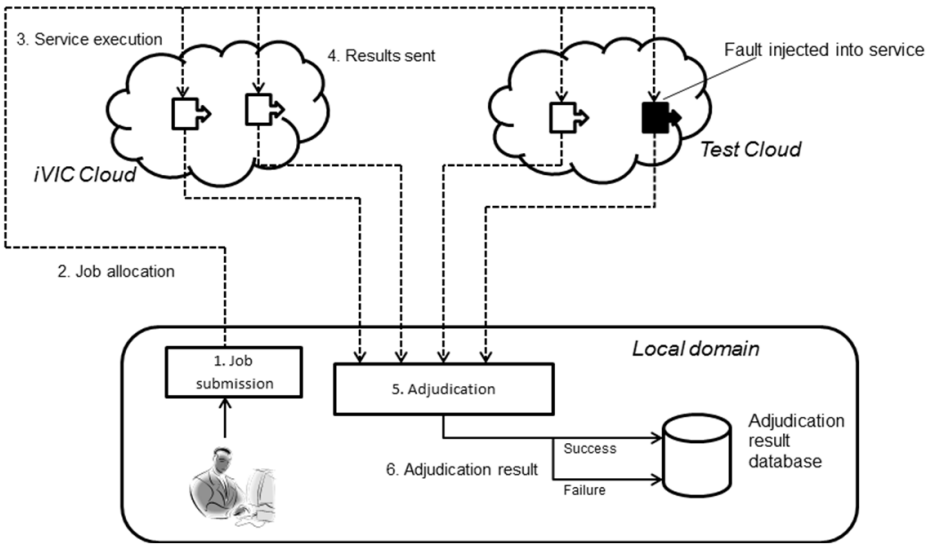

Figure 2. The experiment system created with the FT-FC Framework

percentage of failure for each service in the experiments were set at 10%, 5% and 1%.

For the first deployment, the above percentages were set and applied to all services on an individual Cloud at varying levels(for example, services in iVIC have a 10% chance of failing, while Test Cloud services have a 5% chance of failing). The second deployment contained failure percentages the same as the second experiment, but also included a 10% chance of failure propagation to a service residing on the same Cloud infrastructure. In this experiment we assume that the job submission systems and the adjudicator are fault free. We also assume that the failure independence between services residing across heterogeneous Cloud environments is stronger than that of services within the same Cloud.

Three different types of failure were modeled and injected into each of the services for the experiments: value faults, omission faults, and late timing faults. Omission faults cause the data submitted from the 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 the service actually arrives in the adjudication phase. Value faults returned to the adjudicator lie outside the valid value boundaries.

The experiment was run for all three fault types 2500 times, at all three failure rates respectively (10%, 5%, 1%). These failure rates values represent the annual failure rate of Amazon EBS snapshots amplified by a factor of 10.(Annual failure rate of EBSs lie between 0.1% and 0.5%) [27]; the reason for this is the necessary time required to run the experiments at these values and return accurate results in the given time frame. The experiments are visualised in figure 2; the experiment functions in the order as follows:

1) *Job Submission:* The system automatically submits jobs to individual Clouds from the local domain.

2) *Job allocation:* Jobs are submitted to N-Copy services in both Clouds.

3) *Service execution:* Each virtual machine containing the replica service executes and processes the e-science application. At this stage faults are injected into the service.

4) *Results sent:* Each service submits its results as structured data to the adjudicator, which resides back in the local domain.

5) *Adjudication:* The adjudicator decides whether the returned results are acceptable in the given time and/or value domains to remain dependable.

6) *Adjudication result:* Result is either cast as a success or failure; either result is recorded in a database in the local domain.

The framework records the result of the adjudication process as well as information such as the total job submission time, success state of the adjudicator, the reason for failing (if any) and the individual flagging of detected failures.

V. RESULTS AND EVALUATION

Table 1 shows the comparison of the rate of adjudication success in experiment 1 and 2. The second column 'Failure % iVIC, Test Cloud' contains two numerical values. For example, 10-5 represents the services in the iVIC Cloud having a failure rate of 10%, and 5% for Test Cloud. The reason for this table is to contrast the effectiveness of the N-Copy scheme selected in the presence of failure propagation and with no failure propagation. A diagram of table 1 is shown in Figure 3; which shows that our selected N-Copy scheme in a federated Cloud experiences a lower adjudication success in the presence of failure propagation than that of a federated Cloud without failure propagation. The scheme shows that it can achieve successful adjudication rate above 99.7% in the

presence of fault injections with Clouds failure rates of iVIC equal to 5% and Test Cloud failure 1%, as well as in the presence of failure propagation. It is expected that the adjudication success rates in experiment 2 would be lower of that in experiment 1, as failure propagation decreases the independence of components that reside within the same infrastructure.

Fault Type	Failure % iVIC, Test Cloud	Experiment 1 adjudicate success %	Experiment 2 adjudicate success %	Abj. difference %
Value	10-5	95.68	94.2	1.57
	5-1	99.76	98.8	0.97
Time	10-5	96.4	94.1	2.44
	5-1	99.44	98.6	0.85
Omission	10-5	96.6	92.6	4.31
	5-1	99.48	98	1.51

Table 1. Experiment 2: Differing failure rate Clouds with failure propagation

The success of adjudication would have been lower if all services with the potential to propagate faults resided on a single Cloud as opposed to a federated Cloud environment. In terms of our experiment set up, the reason for this phenomenon is because the failure independence between services on iVIC and Test Cloud is stronger than that of services residing on an individual Cloud. In the event of a service failing, instead of a propagation occurring from one service to another and only once, services residing on the same Cloud infrastructure could for example have the capability to propagate to one of three services. This problem is also amplified when considering secondary or tertiary failure propagation after the initial failure.

The experimental results, while preliminary, help us validate an issue concerning Cloud consensus described earlier in section II about the dangers of coupling system behavior and the increased probability of failure propagation within a single Cloud infrastructure, and how this problem might be better tolerated in a federated Cloud environment. This is something of great interest to us, which we hope to study in detail in later work.

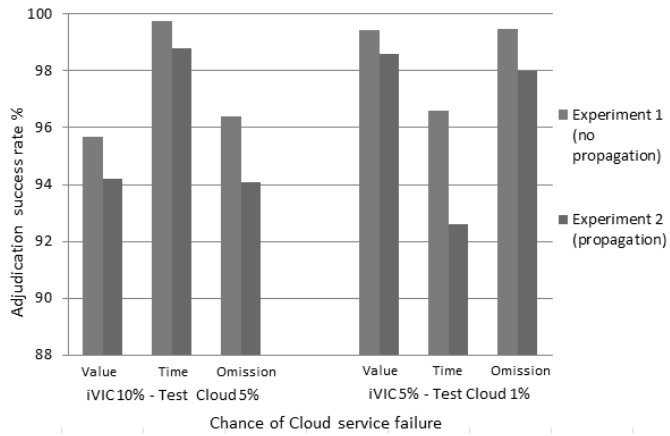

Figure 3. Experiment efficiency comparison

VI. FUTURE WORK

This initial work opens up a wide variety of research areas and challenges for future work; we have identified several promising areas, and list these as follows:

- *A more accurate real world Cloud faults and system model.* Our set up of faults and failure propagation in the experiments are relatively simplistic, with faults injected into the application layer of the Cloud, as well as a fixed rate of propagation. Research into the realistic classification of faults found on Cloud and the modeling of Cloud failure propagation is urgently needed. This would also give future work a more precise answer into what a Byzantine fault on a Cloud might occur and its effects, allowing us to inject more complex and real world faults. There has been some initial work in injecting faults into the virtualisation layer of Clouds [28]; however the paper does not encompass Byzantine faults.
- *Cloud characteristics that will affect BFT.* There are a number of characteristics that have not been considered in the scope of this paper. Cloud dynamicity in the form of scalability (this may affect the performance of a BFT scheme) and virtual machine migration (this may increase the probability of failure propagation between Clouds, which will affect the failure of independence and therefore the strength of BFT), the use of eventual consistency [29] in Clouds (not all applications can operate using eventual consistency) and critical Cloud components (for example, an investigation into how Cloud components used in a N-Design scheme could be given different weightings in an adjudication algorithm).
- *Further experimentation using current BFT protocols in federated Cloud*, as well as looking closer at the nature of coupling and decoupling Clouds. We believe that the experiments ran are decent preliminary work into how federated Cloud might be able to address the problems of coupling nodes in Cloud systems.

As mentioned in previous sections, one of the problems of Clouds is that of the massively scalable and contain large size of redundant components. As a Cloud system scales up, so does the frequency of failures on a Cloud platform. Therefore, Clouds should be designed under the assumption that they will experience frequent and potentially unpredictable failures; this requires services to tolerate and recover from failures autonomously as well as have fast recovery procedures [30]. Recovery Oriented Computing [31] works under the premise of reducing mean time to recovery, and a large and scalable form of this might be of benefit to this line of research.

Our experiment involves four virtual machines on two small Cloud infrastructures; future work and subsequent findings would be greatly improved with the inclusion of test data and experimentation from larger Clouds.

VII. CONCLUSION

Building dependable Clouds is a challenging research area, but the prospect of increasing numbers of federated Cloud environments offers significant potential to apply Byzantine fault-tolerance – a topic that has been hitherto considered unfeasible in traditional single Cloud systems. This paper presents that the application of BFT to federated Clouds has been analysed in detail, and that actual experimentation has been performed to test the effectiveness of Byzantine fault-tolerance in federated Clouds. We have developed a framework called FT-FC that allows us to very quickly create diversity-based Byzantine fault-tolerant systems and apply them to federated Clouds, and have produced initial results to demonstrate the feasibility and potential of this approach. We have furthermore identified a number of research problems and challenges that need to be addressed in order to progress this area further. Our current experimental results are very initial and need to be developed further, but show highly encouraging figures, and demonstrate the effectiveness of the FT-FC framework at its current stage of maturity.

ACKNOWLEDGEMENTS

We would like to thank James Hardy from the University of Derby for technical assistance in installing iVIC.

REFERENCES

[1] Above the Clouds: A Berkeley View of Cloud Computing" by Michael Armbrust et al. Technical Report EECS-2009-28, EECS Department, University of California, Berkeley

[2] Buyya, R.Ranjan, R.N.Calheiros 'InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services.' Proceedings 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

[3] Z.Zheng, T. Zhou, M.Lyu, I.King 'FTCloud: A Component Ranking Framework for Fault-tolerant Cloud Applications' 2010. IEEE 21st International Symposium on Software Reliability Engineering, Nov. 2010, pp 398-407

[4] B. Rochwerger 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 53 (4) (2009) 4:1_4:11 (special issue).

[5] A. Celesti, et al. 'Three-phase Cross-cloud Federation Model: The Cloud SSO Authentication' Published on Proceedings of The 2nd IEEE International Conference on Advances in Future Internet (AFIN 2010), Venice, Italy July 2010

[6] Jemal Abawajy, "Determining Service Trustworthiness in Intercloud Computing Environments," ispan, pp.784-788, 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, 2009

[7] C. Cachin, R. Haas, and M. Vukoli´c. Dependable storage in the Intercloud. Research Report RZ 3783, IBM Research, Aug. 2010

[8] B .Rochwerger, et al., 'Reservoir - When One Cloud Is Not Enough,' Computer, vol.44, no.3, pp.44-51, 2011.

[9] Jeff Barr, Attila Narin, and Jinesh Varia, 'Building Fault-tolerant Applications on AWS', October 2011

[10] Pankaj Goyal 'Enterprise Usability of Cloud Computing Environments: Issues and Challenges 2010' 19th IEEE International Workshops on Enabling Technologies Infrastructures for Collaborative Enterprises (2010) Pages: 54-59

[11] B. Randell et al., "Dependability – Its Attributes – Impairments and Means", in Predictably Dependable Computing Systems, Springer-Verlag, 1995

[12] S.Shankland 'Amazon suffers U.S. outage on Friday' http://news.cnet.com/8301-10784_3-9962010-7.html

[13] K. Birman, G.Chockler, R. van Renesse 'Toward a cloud computing research agenda.' SIGACT News, 40, 2, 68-80, 2009

[14] Amazon Usage Estimates - http://blog.rightscale.com/2009/10/05/amazon-usage-estimates/

[15] http://aws.typepad.com/aws/2011/07/amazon-s3-more-than-449-billion-objects.html

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

[17] M. Castro and B. Liskov. Practical Byzantine fault-tolerance and proactive recovery. ACM Trans. Comput. Syst., 20(4):398–461, 2002.

[18] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. 'Hey, you, get off of my cloud: exploring imation leakage in third-party compute clouds. CCS '09: Proceedings of the 16th ACM Conferon Computer and Communications Security, pp 199–212, New York, NY, USA, 2009. ACM.

[19] Amazon outage - http://aws.amazon.com/message/65648/

[20] Leslie Lamport , Robert Shostak , Marshall Pease, The Byzantine Generals Problem, ACM Transactions on Programming Languages and Systems (TOPLAS), v.4 n.3, p.382-401, July 1982

[21] Vukolić, M.: The byzantine empire in the intercloud. SIGACT News 41, 105–111 (2010)

[22] R. Guerraoui and M. Yabandeh. Independent faults in the cloud. In LADIS '10: Proceedings of the 4th ACM SIGOPS/SIGACT Workshop on Large-Scale Distributed Systems and Middleware, pages 12--16, 2010

[23] L. Tang, J. Dong, Y. Zhao, and L. Zhang, "Enterprise Cloud Service Architecture," in Proc. of CLOUD'10, 2010, pp. 27–34

[24] M. Birkin, P. Townend, A. Turner, B. Wu, J. Arshad, J. Xu, "MoSeS: A Grid-enabled spatial decision support system", in Social Science Computing Review, in press. DOI: 10.1177/089443930933229, 2009

[25] A. Avizienis, "The N-version approach to fault-tolerance software", IEEE Trans. Software Eng., vol. SE-11, pp.1491 - 1501 , 1985.

[26] Paul E. Ammann , John C. Knight, Data Diversity: An Approach to Software Fault-tolerance, IEEE Transactions on Computers, v.37 n.4, p.418-425, April 1988

[27] Srijith K. Nair, Sakshi Porwal, et al. "Towards Secure Cloud Bursting, Brokerage and Aggregation," ecows, pp.189-196, 2010 Eighth IEEE European Conference on Web Services, 2010

[28] C. Pham, D. Chen, Z. Kalbarczyk, R. K. Iyer CloudVal: A framework for validation of virtualization environment in cloud infrastructure 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN) (June 2011), pg. 189-196

[29] W. Vogels 'Eventually consistent', Communications of the ACM, v.52 n.1, January 2009

[30] [J. Hamilton. On designing and deploying Internet-scale services. In LISA'07: Proceedings of the21st conference on Large Installation System Administration, pages 1–12, Dallas, TX, 2007. USENIX Association.

[31] D Patterson. Recovery Oriented Computing. 2009. http://roc.cs.berkeley.edu.

[32] Liang Zhong, Tianyu Wo, Jianxin Li, and Bo Li,"vSaaS:A Virtual Software as a Service Architecture for Cloud Computing Environment", Poster of the 5th IEEE International Conference on e-Science, 2009.

[33] A. Chandra and J. Weissman, "Nebulas: using distributed voluntary resources to build clouds," in Proc. of HOTCLOUD'09, 2009