2016 IEEE International Conference on Computer and Information Technology

Byzantine Fault Tolerant Optimization in Federated Cloud Computing

Hojjat Baghban[1], Mahdis Moradi[2], Ching-Hsien Hsu[3*], Jerry Chou[1], Yeh-Ching Chung[1]
[1]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[2]Department of Computer Science, Islamic Azad University, Marvdasht , Iran
[3]Department of Computer Science and Information engineering, Chung Hua University, Hsinchu, Taiwan
* The corresponding author

***Abstract*— Cloud computing is a forthcoming revolution in information technology industry because of its performance, accessibility and, low cost. It is appropriate to maximize the capacity or step up capabilities vigorously without investing in new infrastructure, nurturing new personnel or licensing new software. The federated cloud, which is the combination of more than one cloud, is the next logical step after hybrid cloud and there are many indicators that are showing more requirements for such a model. Security is the challenging issue in all cloud infrastructures such as single cloud and federated cloud. And, it is significant in distributed systems which have highly fault tolerant. One of the algorithms for this issue is Byzantine Fault Tolerant. This paper introduces a new method, that optimizes the Byzantine Fault Tolerant and decrease the latency, and detecting the number of faults.**

Keywords*— Multi cloud, security, fault tolerant, byzantine fault tolerant

I. INTRODUCTION

Cloud primarily refers to the saving of user's data to storage system that is maintained by a third party. In other words, instead of storing information on user computer's hard disk or other storage devices, client saves it to a remote database where the internet provides the connection between the user's computer and the remote database.

Many companies migrate from single cloud to federated cloud, because federated cloud is a combination of more than one cloud (private, community or public) or it is a group of clouds that are not necessarily sharing the same infrastructure, architecture standards, geographical location or security setting. In cloud computing the security issues are very important and many researchers focus on this challenge.

In cloud computing or in federated cloud, any faults in software or hardware are known as Byzantine faults that usually related to inappropriate behavior and intrusion tolerance [5]. In addition, it includes arbitrary and crash faults. Much research is dedicated to byzantine fault tolerance (BFT).

BFT requires a high level of failure independence. If byzantine failure occurs to the particular node in the cloud, it is reasonable to have a different operating system, different implementation and different hardware ensure such failure does not spread to other nodes in the same cloud.

BFT algorithm typically requires 3f+1 servers to tolerate 'f ' byzantine servers [1], which involve considerable costs in hardware, software and administration. Byzantine fault tolerant protocol is complicated and hard to implement. Today's software industry is reluctant to adopt these protocols because of the high overhead of message exchange in an agreement phase and high resource consumption necessary to tolerate faults (as 3f+1 replicas are required to tolerate f faults)

Byzantine fault tolerant system use state machine replication to tolerate a wide range of faults. It means we need two phases: agreement and execution that will explain in the following sections [7].

II. RELATED WORK

Recent activities in federated cloud can be summarized as follow.

In 2011, Bessani, et al. introduced "Depsky", which is a virtual storage cloud system comprising of a combination of different clouds to build a cloud of cloud. The service provider can work with the mixture of Byzantine protocol, encryption and erasure code [3].

Also, the RACS (Redundant Array of Cloud Storage) is a protocol for inter-cloud storage in the year of 2010. This technique is similar to RAID and normally used by disks and file systems and replication offers better fault tolerant. But it has an availability problem that cannot address the storage areas and cannot update them [4]. Chachin presented a design for inter-cloud storage which is called ICStored in 2010. ICStored is the client centric distributed protocol which can handle the data integrity. But, it has poor performance in case of data intrusion and service availability [4].

HAIL technique (High Availability and Integrity Layer), which was introduced in 2009, is a distributed encryption system that lets the servers to store data like a retrieval system. It is worth saying that this model cannot guarantee the integrity [3].

The byzantine in distributed system has a history. This model was introduced by Pease, Shostock and Lamport in 1980 [5], for the first time. In 1983 the distributed system was divided into synchronous and asynchronous, in Ben-or introduced the t-resilient system for asynchronous system that can tolerate the $t<n/2$ faulty process that is fail-stop fault and tolerate $t<n/5$ faulty process that is Byzantine faults [9].

In 1985 in Bracha and Toueg introduced the weak model that many of scheduler is probabilistic [9]. Also, the Byzantine based algorithm such as PBFT, BFT2F, ZYZZYVA, cheapBFT were introduced in [5]. BFT2F look for 'f ' faulty areas and the main goal is limited the faulty system but the

978-1-5090-4314-9/16 $25.00 © 2016 IEEE
DOI 10.1109/CIT.2016.114

problem is still situated where the client should send two messages and if the system fell down between send the first and second message, this algorithm can't tolerate it. The ZYZZYVA is the optimistic model of BFT, which the main goal is reducing the cost and simplicity in designing and decrease the cost, computation overhead and latency. Then, the CheapBFT, which uses the resources very efficiently, was introduced [5]. Table 1 shows the summary of the mentioned algorithms [1].

TABLE I. COMPARISONSOF ALGORITHMS

	Throughput	Latency	Cost
PBFT	$2+(8f+1)$	$\frac{5}{4}$	3f+1 2f+1
ZYZZYVA	$2+\frac{3f}{b}$	3	3f+1 2f+1
TTCB	3	5	2f+1 2f+1
A2M-PBFT-EA	$2+(2f+4)$	5	2f+1 2f+1
MINBFT	$2+\frac{(f+3)}{b}$	4	2f+1 2f+1

Another replication based model was introduced in [6]. This model can have more availability in terms of cloud databases. Most of the replication protocols are designed for Crash-Stop model and the number of them was designed for Byzantine faults. But it has deficiency problem. Because, each server can execute the part of the transaction and, the server should run sequentially [6].

III. PROPOSED METHHOD

The most important goal of this research is to maintain the three vital security impact such as, Availability, Integrity and Confidentiality. Replication is widely used to improve the availability, reliability of services by mirroring the data from a server on multiple machines. If the primary servers crashed, the data is not lost. Because, it is available on the other machine. The question is how many copies should be existed as a backup The answer depends on how many faults the service can be tolerated. In general, two types of faults are addressed in replicated service: Omission faults and Commission faults [10]. Omission faults occur when a node does not send a message which would have been sent by a correctly operation node. Omission faults are common. But commission faults occurred when a node sends a message which would not have been sent by a correctly operating node. These kind of faults are difficult to resolve [10].

Byzantine faults are arbitrary faults that occurred in a system and make the system either unreliable or unresponsive to any client requests. In same system, there are two phases: agreement and execution. [7] High overhead existed on agreement phase and a lot of work has been done to improve the performance of execution phase and little work has been done to improve the agreement phase. Agreement phase sometimes known as consensus. In terms of fault tolerance system, we need State Machine Replication (SMR) [7, 8]. Every modern service uses SMR to tolerate faults. But, the SMR is non-deterministic. It is worth saying that in this paper all the states and sequence of requests are assumed to be deterministic. In agreement phase, we need two conditions as follow,

- All non-faulty replicas agree on the same value.

- If the sender is non-faulty then all the non-faulty replicas use its proposed value as agreed value.

A. *Conditions which affect the proposed method*

- The number of replicas or servers: In a normal form of byzantine algorithm, it demands 3f+1, in terms of tolerating the 'f ' faults. Decreasing the copies of replicas can be affected by price and intrusion power.

- Trusted service simplicity: In additional of decreasing the replicas it should be trusted that the services work truly.

- Number of communication steps: this part of the algorithm is very important because, it can affect latency in terms of sending and receiving messages. In this algorithm, two kinds of replicas are introduced, primary and back up, that the primary nodes send the messages which received from the sender to the backup nodes. The amount of latency can measured from a number of communications.

- Benefits and Drawback: the main problem of byzantine is duplicity. So, the faulty system, maybe send the messages to two systems on two different servers.

- Contribution: use the combination of last algorithms that work with each other truly and with high trust.

B. *The proposed algorithm operations*

- A sender sends the request (job) to the primary nodes for recall the service functions.

- The Primary replicas send the job to the 'f' backup nodes.

- The backup replicas execute the job and send the replies to sender.

- The sender waits until receiving the 'f' replies from replicas, compares the replies and shows the results.

- If the sender doesn't receive enough replies (less than f replies) resend the request to all replicas. If the request processed, the replicas resend the replies easily.

C. *Faults Detection and resolving levels*

First level: If the primary doesn't send the job, the probability that the primary is faulty will be increased.

Second level: If the primary sends the job to the backup provided that the backup doesn't execute the job, it is probable that one or many of backup nodes receive the faulty message or doesn't receive any messages.

Third level: If all the backups execute the job and send the replies to sender and also, if the number of received replies is correct, the sender compares the replies for finding the faulty nodes.

Figure 1 illustrates the proposed approach.

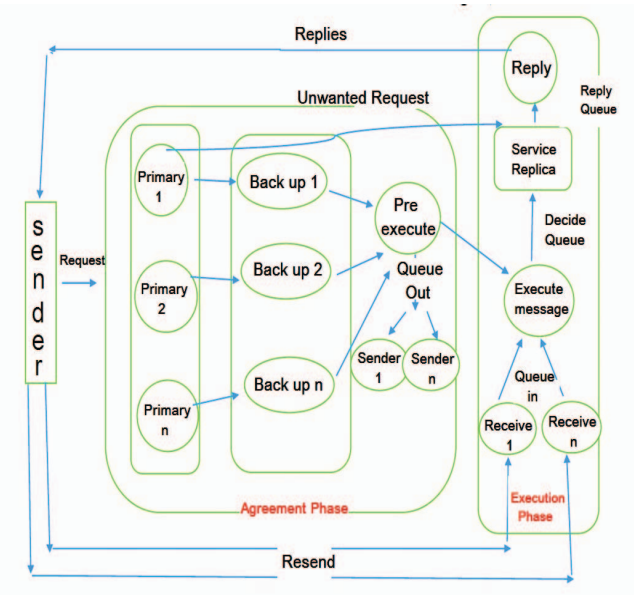

Fig. 1. The proposed models and relationships

IV. EVALUATION

The proposed approach is simulated through CloudSim [13] which is a widespread and extensible simulation toolkit and application that enables unified modeling, simulation and experimentation of developing cloud computing systems.

The node behavior in proposed method concern with the parameters, Availability (A), Reliability (R), Integrity (I) and Throughput (T), which are formulated as follow,

$$B = A \times R \times I \times T \tag{1}$$

where

$$A = \frac{a}{N} \tag{2}$$

$$R = \frac{b}{a} \tag{3}$$

$$I = \frac{c}{b} \tag{4}$$

$$T = \frac{c}{t} \tag{5}$$

Availability: The grade to which a system or component is accessible and working when required for use.

Reliability: The reliability of a cloud resource is an amount of accepted jobs that are completed successfully by the cloud resources.

Integrity: Security is a key factor that requires special care in cloud. Data integrity is a general term that comprises accuracy, privacy and security of data.

Throughput : The measurement of the amount of requests which can be processed in a given period of time.

- ✓ *a*: The number of jobs that accepted via Virtual Machines (VMs).
- ✓ *N*: The number of jobs that allocated to VMs.
- ✓ *b*: The number of requests that accepted via service providers.
- ✓ *c*: The number of jobs that accepted via data centers.
- ✓ *t*: Total executing time of accepted job.

The table 2 demonstrates an example which is concerned with mention equations and their results.

TABLE II. RESULTS OF THE MENTION SCENARIO (SECONDS)

		CSP1	CSP2	CSP3	CSP4
Availability Based	***N***	20	20	20	20
	A	14	12	13	14
Reliability Based	***A***	14	12	13	14
	B	10	10	9	12
Integrity Based	***B***	10	10	9	12
	C	6	7	8	9
Throughput Based	***C***	6	7	8	9
	t *(second)*	9	9	9	9

V. LATENCY AND COMMUNICATION OVERHEAD

Another important parameter which are concern with this research are latency and communication overhead in the multi cloud. The Overhead is measured through two fundamental metrics, the Average Number of Clouds (ANC) involved in service composition and, second, the Average Number of Service files (ANS) examined [11].

The communication overhead depends on the physical locations of managers [12]. For example, one data center across different data centers. We have '*n*' data centers D_1, D_2, …, D_n. For the data center D_1 there are m_i VMs running: $V_{i,1}$, ... , $V_{i,mi}$. As each VM is accompanied by a cloud provider. We denote the cloud provider as $C_i = C_{i,1}, \ldots, C_{i,mi}$. The size of one message from $C_{i,j}$ is $M_{i,j}$. For each cloud provider that located in data centers: D_{cp}, $cp \in [1, n]$.

Total communication overhead is measured as below,

$$\sum_{1}^{Mi} j((\sum_{1}^{n} iM_{i,j}) - M_{p,j} \tag{6}$$

If the message size is a fixed value *M* then communication overhead is:

$$M((\sum_{1}^{n} iM_i) - M_p \quad (7)$$

The number of primary nodes computes via this rule is n=3f+1 that 'n' is the number of nodes and 'f' is the number of faulty nodes. If the number of nodes is 4 then this algorithm can detect 1 fault (if it is exists in the system). If the number of nodes increases to 5 or 6, the algorithm also can detect just 1 fault, but if the nodes increased to 7, it can detect 2 faults. Figure 2. illustrates this situation. Fault detection in primary nodes follows the byzantine rules (n=3f+1). But, as it can be seen in Figure 3. there is an optimized situation in the backup nodes and can work with this rule which n= 2f+1 and detect the fault earlier than the previous level. For instance, at first level (primary nodes) if the number of nodes is 7, this algorithm can detect 2 faults, but in backup level, if the number of nodes is 7, it can detect 3 faults. This earlier detection can protect the system from distributing the faults and avoiding continuing to another level.

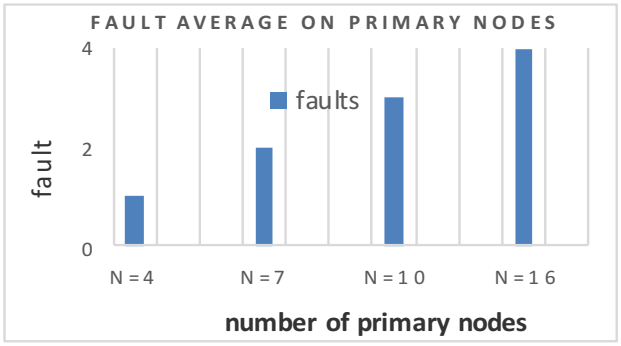

Fig. 2. Average fault for primary nodes

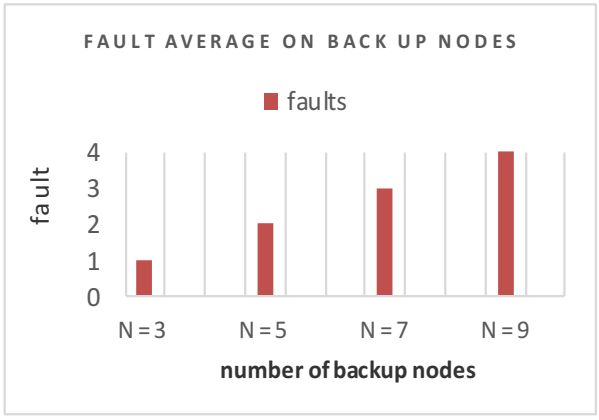

Fig. 3. Average fault for backup nodes

VI. CONCLUSION

It is clear that although the use of cloud computing has increased rapidly, most of the organizations eager to use federated cloud instead of single cloud and cloud security is a major concept in a cloud computing environment.

This paper focused on the issue related to security and fault tolerance. The evaluation shows the simulated model, which maintains the availability, reliability and integrity, detects the faults on different nodes in an earlier level to avoid expanding the faulty nodes on all of the system. It is worth saying that reliability, integrity and faults detection on different nodes are considered in an earlier level to avoid expanding the faulty nodes on the system.

ACKNOWLEDGEMENT

This paper is supported in part by Ministry of Science and Technology, R.O.C., under grant no. MOST-105-2218-E-007-011

REFERENCES

[1] Giuliana Santos Veronese , Miguel Correia , Alysson Neves Bessani, Lau Cheuk Lung and Paulo Verissimo , "Efficient Byzantine Fault Tolerance " , IEEE Transaction on Computer , 2013.

[2] Rodrigo Nogueira, Filipe Araujo and Raul Barbosa, " CloudBFT: Elastic Byzantine Fault Tolerance ", IEEE 20th pacific Rim International Symposium on Dependable Computing, 2014.

[3] Swapnila S Mirajkar and Santoshkumar Biradir, "Secret Sharing Based Approach to Enhance Security In Cloud Computing", International Journal of Advanced Research In Computer Science and Software Engineering, 2014.

[4] Swapnila S Mirajkar and Santoshkumar Biradir," Using Secret Sharing Algorithm for Improving Security in Cloud Computing", 2014.

[5]Vasileios Papadopoulos, "A Study of Byzantine Fault-Tolerant Algorithm".

[6] Fernando Pedone, Nicolas Schiper, Jose Enrique and Armendariz- Inigo, "Byzantine Fault-Tolerant Deferred Update Replication", Latine –American Symposium on Dependable Computing, 2011.

[7] Mohammed A.Alzain and Ben Soh and Eric Pardede, "A Byzantine Fault Tolerance Model for a Multi-Cloud Computing ", IEEE 16th International Conference on Computational Science and Engineering, 2013.

[8] Aldelir Fernando Luiz, Lau Cheuk Lung and Luciana de Oliveira Rech, "On the Practicality to Implement Byzantine Fault Tolerant Services Based on Tuple Space", IEEE 28th International Conference on Advanced Information Networking and Applications, 2014.

[9] Gabriel Bracha,"Asynchronous Byzantine Agreement Protocol", Information and Computational, 1987.

[10] Kim Potter Kihlstrom, Louise E.Moster and P.M.Melliar-Smith, "Byzantine Fault Detectors for Solving Consensus", British Computer Society, 2008.

[11] Jun Wen Luand Yongsheng Hao,"Toward Efficient Service Composition in Multi-Cloud Environment ", International Conference on Computational Science and Computational Intelligent.

[12] Khalid Alhamazani , Rajiv Ranjan, Prem Prakash Jayaraman, Karan Mitra , Chang Liu, Fethi Rabhi, Dimitrios Georgakopoulos and Lizhe Wang, "Cross–Layer Multi–Cloud Real-Time Application QOS Monitoring and Benchmarking As-a-Service Framework "

[13] Rodrigo N.Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A.F.De Rose and Rajkumar Buyya , "Cloudsim: A Toolkit for Modeling and Simulation of Cloud Resource Management and Application Provisioning Teckniques", Software :Practice and Experience, 2011.

[14] Li Lin, Tingting Liu, Jian Hu and Jian Ni, "PQsel: combining privacy with quality of service in cloud service selection", International Journal of Big Data Intelligence, Vol. 3, No. 3, pp. 202-214, 2016.