International Journal of Technology 10(7): 1395-1406
ISSN 2086-9614

PREVENTING AND RECTIFYING CLOUD QUALITY OF SERVICE VIOLATION THROUGH ADAPTIVE RESOURCE SCALING AND REPLICATION

Tong-Sheng Wong[1], Gaik-Yee Chan [1*], Fang-Fang Chua[1]

[1]*Faculty of Computing and Informatics, Multimedia University, Persiaran Multimedia, 63100 Cyberjaya, Selangor, Malaysia*

(Received: November 2018 / Revised: January 2019 / Accepted: September 2019)

ABSTRACT

One major challenge in delivering and accessing cloud applications is the management of Quality of Services (QoS). It is mandatory for cloud service providers to ensure their performance and fulfil QoS, as defined in the Service Level Agreement (SLA). In this paper, we propose a Scaling and Fault Tolerance (SFT) algorithm to deploy preventive or remedial measures based on 16 decision rules for QoS violation detection and prediction. We simulate the SFT algorithm in a cloud simulator with four scenarios to measure its effectiveness in handling events such as faulty virtual machines (VMs), or over and under-provisioning of resources. Our experimental results show that the proposed SFT algorithm performs effectively (close to a 90%−100% effective rate) in providing preventive or remedial measures and reducing the number of VMs when they are not needed.

Keywords: Cloud computing; Fault tolerance; Quality of service violation; Replication; Scalability

1. INTRODUCTION

According to the National Institute of Standards and Technology (NIST) (Mell & Grance, 2011), cloud computing has emerged as one compelling paradigm for providing convenient and 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 interaction. This has made possible the hosting of cloud services provided by cloud service providers, such as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). An increasing number of organizations are adapting to cloud computing platforms for their daily business functions. These organizations are showing determination in embracing Industry 4.0, as cloud computing helps to pool and centralize information for making better business decision. The focus of this paper is therefore on the Quality of Service (QoS) pertaining to SaaS.

Cloud service providers are mandated to enforce service performance and the quality of their services, as defined in the Service Level Agreement (SLA). From their perspective, maintaining the conditions defined in the SLA and maximizing the QoS metrics are important tasks. We define QoS metrics as CPU load, response time and throughput, as elements of the performance aspect, for evaluating cloud services (Bardsiri & Hashemi, 2014).

Cloud monitoring tools measure and collect cloud QoS data, and this information is used for making decisions on scaling cloud resources horizontally, and also for providing fault tolerance

[*]Corresponding author's email: gychan58@yahoo.com, Tel. +60122097865
Permalink/DOI: https://doi.org/10.14716/ijtech.v10i7.3252

if necessary. Therefore, using QoS metrics for modelling the performance of cloud services could enable better decision making with regard to preventing or rectifying any cloud QoS violation by correlating the decision rules with the QoS metrics, such as response time and throughput. A detailed description of the formulation of these decision rules can be found in the study by Wong et al. (2018).

The work by Wong et al. (2019) has proven the feasibility of using horizontal scaling as a preventive measure and fault tolerance mechanism of replication, as rectification for QoS violations. In this paper, we propose a scaling and fault tolerance (SFT) algorithm and evaluate its effectiveness based on CloudSim Plus (Filho et al., 2017), a toolkit with libraries for the simulation of cloud computing scenarios. A total of four scenarios were used in the simulation to evaluate the effectiveness of the proposed SFT algorithm. One scenario provided a preventive measure for probable cloud QoS violation; another a remedial measure for certain cloud QoS violation; while the other two monitored consideration of over- or under- provisioning of virtual machines (VMs). One example for monitoring the provisioning of VM is by reducing the number of idle VM based on real-world QoS measurement of cloud services, as discussed by Zheng et al. (2014).

Our experimental results show that the proposed SFT algorithm performs effectively (close to a 90%–100% effective rate) in providing preventive or remedial measures and reducing the number of VMs when they are not needed, consequently guaranteeing QoS performance, as defined in the cloud services SLA. Additionally, the 16 decision rules determine QoS violation at four levels, namely no violation, normal, probable violation and certain violation; unlike many other works, that only define violations as normal or violation, the four levels are able to detect and predict whether a violation will occur or not before it actually happens. The SFT algorithm takes appropriate action to prevent the occurrence of actual violation, or rectifies it if violation has occurred. Together with the 16 decision rules, it thus contributes towards another aspect of detection, prediction, prevention and rectification measures with regard to response time and throughput for cloud QoS violations.

Unlike the work of Aruna and Aramudhan (2016), which includes cost in its proposed method of using fuzzy sets to shortlist providers based on the QoS agreed in the SLA, the SFT algorithm does not include the cost factor in resolving QoS violations. This will be left for future work.

Scalability is defined as the handling of increasing workloads by allocating more resources to the system (Lehrig et al., 2015). There are two general scalability approaches, namely horizontal and vertical scaling. Horizontal scaling involves adding or removing VMs to spread the load across multiple distributed VMs, while vertical scaling involves increasing and decreasing the power of an existing VM by means of more memory (RAM), storage (HDD/SSD), or processors (CPUs). In this paper, the focus is on application scalability, which is defined as the maintenance of cloud services application performance goals by avoiding QoS violation events when the workload submitted by users increases (Kuperberg et al., 2011).

Fault tolerance, as defined by Ganesh et al. (2014), is the ability of the cloud environment to handle unanticipated changes, such as hardware failure, software defects or network congestion. Two standard policies, namely proactive and reactive fault tolerance, can be used for real-time cloud applications. Proactive fault tolerance can predict faults, errors and failures, and once a suspicious component has been detected, it will be replaced proactively. Proactive fault tolerance techniques include pre-emptive migration, software rejuvenation and self-healing.

Reactive fault tolerance reduces the effect of failure on applications being executed when the failure effectively occurs. Examples of reactive fault tolerance techniques are check pointing or restart, replication, job migration and task resubmission. In this paper, the focus is on

implementing a reactive fault-tolerance policy on computation failure, which involves hardware or infrastructure failure.

2. METHODOLOGY

This section presents an overview of our system incorporated with the proposed SFT algorithm. Based on the work of Wong et al. (2018), 16 decision rules were derived for the detection and prediction of cloud QoS violations. In the study by Wong et al. (2019), adaptive mechanisms, such as horizontal scaling and fault tolerance mechanisms, were proven to be feasible in preventing and rectifying such violations. In this paper, we propose a scaling and fault tolerance (SFT) algorithm which adopts the adaptive mechanisms to provide preventive and remedial measures for handling cloud QoS violations, as well as for monitoring over- or under-provisioning of resources in the cloud environment; please refer to Figure 1 for an overview of the system. The system architecture design includes a cloud broker, a number of servers hosting several running virtual machines (VMs) in a data centre, with other functions such as cloud monitoring, cloud QoS detection and prediction, and cloud QoS violation adaptive mechanisms with the proposed SFT algorithm.

A service level agreement (SLA) is established between the cloud service provider and cloud consumer to guarantee cloud service performance and availability. QoS metrics are used to measure the hosted cloud services in the data centre, with consideration of performance and availability to ensure that QoS requirements are met and to prevent cloud QoS violation occurrences. Such violation might occur due to events such as under-provisioning and computation failure, such as faulty VMs, which could disrupt the daily operations of enterprises or organizations using business or productivity software on the cloud.

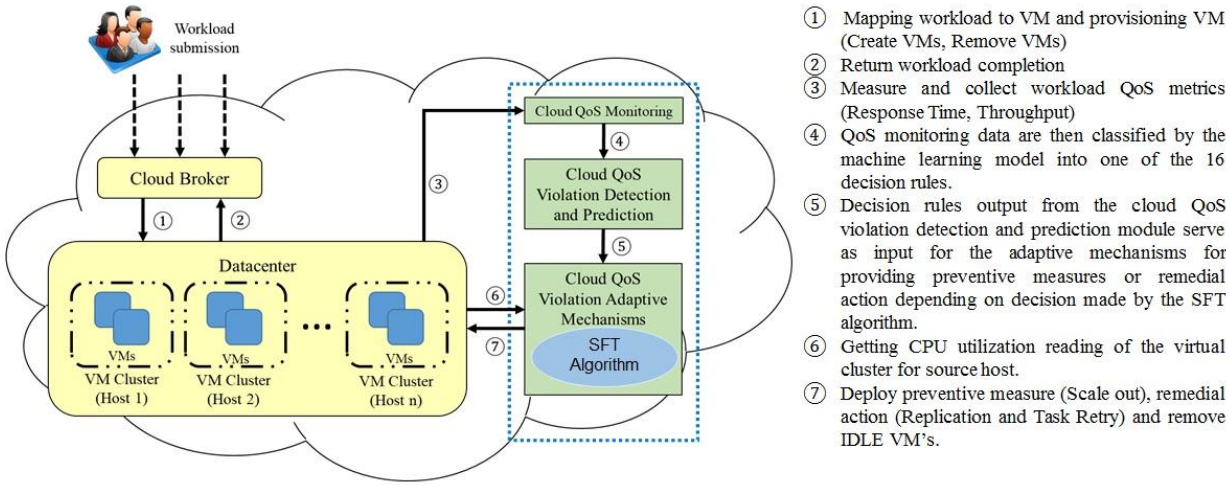

Figure 1 Overview of the proposed system architecture

Basically, a cloud broker is responsible for managing the use, performance and delivery of cloud services, as well as negotiating the relationships between cloud providers and cloud consumers (Mell & Grance, 2011). The datacenter is the infrastructure to host several servers deployed for running applications and processing workload submitted.

2.1. Cloud QoS Monitoring

With reference to Figure 1 (Steps 1-3), a workload request is submitted and received by the cloud broker for processing. Subsequently, this request is allocated to a running VM by the scheduler. The VM, hosted in a server located in the datacenter, processes the workload and upon completion returns a response. The cloud QoS monitoring function is responsible for measuring and collecting QoS metrics, such as response time and throughput.

2.2. Cloud QoS Violation Detection and Prediction

The cloud QoS detection and prediction (Figure 1, Step 4) function introduced by Wong et al. (2018) derives 16 decision rules to determine QoS violation conditions based on response time and throughput. The raw metrics of these, being in a quantitative range of values, are then categorized into linguistic terms. Based on the fact that the relationship between response time and throughput is in inverse order, the 16 decision rules, which are categorized into four main decision outputs of certainly no violation, normal, probable violation and certain violation, as shown in Table 1, are thus derived. The categorization of response time and throughput in linguistic terms, with the classification and derivation of the 16 decision rules, has been proven to be effective through a support vector machine classification multiplier.

Table 1 The 16 decision rules

Rule#	Response Time (seconds) - linguistic category	Response Time (seconds) - range of values	Throughput (kbps) - linguistic category	Throughput (kbps) - range of values	Decision Rules
1	Short	0.112–0.339	High	> 0.881	Certainly No Violation
2	Short	0.112–0.339	Normal	0.377–0.880	Normal
3	Normal	0.340–3.479	High	> 0.881	Normal
4	Normal	0.340–3.479	Normal	0.377–0.880	Normal
5	Short	0.112–0.339	Low	0.124–0.376	Probable violation
6	Normal	0.340–3.479	Low	0.124–0.376	Probable violation
7	Long	3.480–4.340	High	> 0.881	Probable violation
8	Long	3.480–4.340	Normal	0.377–0.880	Probable violation
9	Long	3.480–4.340	Low	0.124–0.376	Probable violation
10	Short	0.112– 0.339	Very Low	0–0.123	Certain violation
11	Normal	0.340–3.479	Very Low	0–0.123	Certain violation
12	Long	3.480–4.340	Very Low	0–0.123	Certain violation
13	Very Long	> 4.341	High	> 0.881	Certain violation
14	Very Long	> 4.341	Normal	0.377–0.880	Certain violation
15	Very Long	> 4.341	Low	0.124–0.376	Certain violation
16	Very Long	> 4.341	Very Low	0–0.123	Certain violation

For example, in Table 1, Row 2 for rule #1, the response time is in the short time region, with a range of values of 0.112–0.339 s, while throughput is in the high value zone, with a range of values greater than 0.881 kbps. This decision rule determines that the QoS is in the 'certainly no violation' condition.

Based on these four main decision output categories, the main concern would be when QoS is in a probable (Table 1, Rows 5-9) or certain (Table 1, Rows 10-16) violation state. This means that when QoS is in a probable violation state, preventive measures should be deployed to prevent further quality downgrade to the certain violation state, instead bringing the situation back to the normal state. Likewise, when QoS is in a certain violation state, remedial action

should be taken immediately to rectify the situation and bring QoS back, if not to normal, but to at least a probable violation state.

2.3. Cloud QoS Violation Adaptive Mechanism

Based on the 16 decision rules, we therefore propose a Scaling and Fault Tolerance (SFT) algorithm implemented with adaptive mechanisms to provide preventive and remedial measures for the two QoS areas of most concern, probable and certain violation conditions (Figure 1, Steps 5-6).

The SFT algorithm in pseudo code form is shown in Figure 2. It first takes the four decision outcomes of certain violation, probable violation, normal and certainly no violation as input for determining QoS violation conditions at 15 second intervals throughout the duration of the running of the cloud service. The algorithm then provides preventive measures, remedial action or decision scales based on the QoS decision outcome. For example, for a real-life SaaS, if the workload submitted is being processed over a long time span due to under-provisioning, the SFT algorithm will deploy a preventive measure of horizontal scaling to scale out the number of VMs to balance the workload, hence guaranteeing the SaaS QoS. Additionally, the algorithm could be applied to homogenous VMs without any modification required; refer to the flowchart in Figure 3 (reading from top to bottom, left to right), for the flow of the SFT algorithm.

Scaling and Fault Tolerance (SFT) Algorithm

Input: *D*: Decision rules of cloud QoS detection and prediction model
(3 : Certain Violation)
(2 : Probable Violation)
(1 : Normal)
(0 Certainly No Violation)
TCurrent: Timestamp of the current run time for every 15s interval
NWorkload_r: Total number of workloads running at *TCurrent*
CPUUtilization: Total CPU Utilization on a scale of [0 to 1]
VMAllocated : VMs allocated for processing workload **i**
VMReplicate : Replication of the faulty VM of *VMAllocated*
VMstatus : Status of the VM
VMTotal : Total Number of VMs in a virtual server host

Output: Scaling and fault tolerance decision

```
1.  For every running workload i of NWorkload_r at TCurrent
2.      If (D = 3) then
3.          If (CPUUtilization >= 0.5) then
4.              VMAllocated + 2 //Scale out by creating and adding two more VMs to the virtual server host
5.          else
6.              Cancel workload i and workloads that are scheduled to run on VMAllocated
7.              Destroy VMAllocated for processing workload i
8.              Replicate new VMReplicate for provision and schedule by system.
9.              Submit VMReplicate for provision and schedule by system
10.             Resubmit workload i and run with VMReplicate
11.     If (D = 2) then
12.         VMAllocated + 1 //Scale out by creating and adding one more VM to the virtual server host
13.     If (D = 0) then
14.         For each VM j in VMTotal
15.             If (VMstatus = IDLE and more than 30 sec upon IDLE) then
16.                 Remove VM //Scale in by removing idle VM
```

Figure 2 Pseudo procedure of the SFT Algorithm

As seen in Figure 3, after the SFT has determined the decision outcome to be certain violation (CV), it then checks for CPU utilization. When this is fully utilized (CPUUtilization = 1), the preventive measure of horizontal scaling is deployed by adding two new VMs to the hosting server, which is to cater for under-provisioning events caused by workload fluctuation due to a larger processing request. However, when CPU utilization is not fully utilized (CPUUtilization < 1), this might indicate an event such as a faulty VM, so the remedial action of replication or task retry is deployed. Additionally, any workload that has been scheduled to run on a faulty VM will be evicted and the faulty VM will be destroyed. Subsequently, a new replicate VM which has the same configuration and image of the faulty VM will be provisioned, started and added to the hosting server. Workload that has been scheduled to be processed with the faulty VM will be resubmitted to the replicate VM. Either the remedial or preventive measure process will end if there is no further workload running on the VM; otherwise, it will route back to the start.

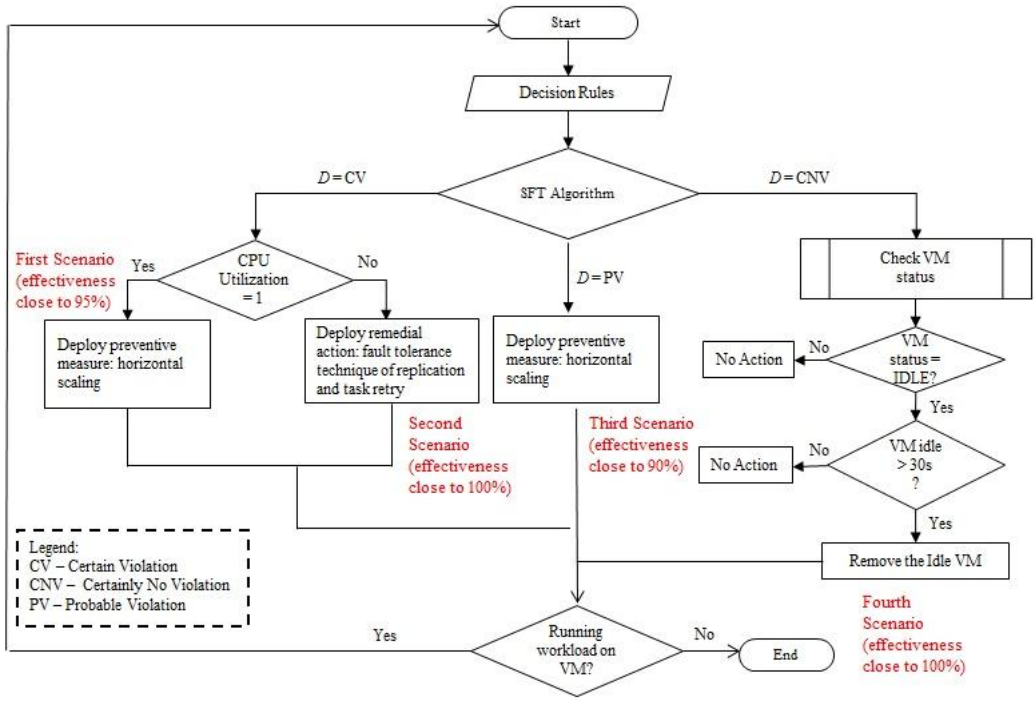

Figure 3 Flowchart for the SFT algorithm

In the event that the decision outcome is probable violation (PV), the preventive measure of horizontal scaling by provisioning a new VM to the hosting server is deployed. This is to cater for under-provisioning caused by the large workload being submitted. Similarly, the process will end if there is no further workload running on VM; otherwise, it will also route back to start.

In the case that the decision outcome is certainly no violation (CNV), the system will check the VM status. If a VM is found to be idle for more than 30 seconds, only then will it be destroyed, otherwise nothing is done. The 30 second waiting interval is to ensure that the VM remains idle in the CNV condition with no new workload submission to provide for continuous availability of the cloud services. The process will route back to the start if there is still workload running on the VM.

3. EXPERIMENTAL SETUP

In order to evaluate the effectiveness of the proposed SFT algorithm, CloudSim Plus (Filho et al., 2017) was used to simulate workloads resembling real-world QoS cloud measurement, as seen in Table 2, which shows a description of the workload based on decision rule distribution.

Referring to Table 2, Columns 2-3 represent the configuration for generating workload which resembles the workload pattern of running web services. Workload request is defined as the number of million instructions per second (MIPS) needed by the VMs for execution of the workloads. Workload size is the size of the workload to be processed. Columns 4-5 show the response time and throughput values computed based on their workload characteristics on one VM. Column 6 represents the frequency distribution of workload characterized by the decision rule stated in Column 7. The last column, Column 8, represents the percentage of each category of decision output; for example, 14%, 71%, 9% and 6% are the percentage distribution for certainly no violation, normal, probable violation and certain violation respectively. These distributions closely resemble real-world QoS measurement of cloud services (Zheng et al., 2014).

To illustrate this further, for example in Table 2, Row 2, the workload request of 22000MIPS accounts for 58% of the workload frequency distribution for the decision rule of certainly no violation. The decision outcome of certainly no violation contributes 14% of total distributions to the real-world QoS measurement of cloud services. A point to note is that not all workloads based on the 16 decision rules could be simulated. For example, short response time and very low throughput (Table 1, Rule#10) do not resemble a real-world QoS measurement of any cloud service.

Table 2 Workload generation based on QoS decision rules

No	Workload Request (MIPS)	Workload Size (kb)	Response Time (s)	Throughput (kbps)	Workload frequency distribution (%)	Decision rules	Decision rules frequency distribution (%)
1	22000	0.3	0.327	0.916	58	Certainly No Violation	14
2	21600	0.3	0.322	0.932	20		
3	20700	0.3	0.308	0.973	8		
4	18000	0.3	0.268	1.119	22		
5	24600	0.3	0.366	0.819	43	Normal	71
6	31500	0.3	0.469	0.639	27		
7	37000	0.3	0.551	0.544	15		
8	44500	0.3	0.663	0.453	11		
9	50500	0.3	0.752	0.399	4		
10	59900	0.3	0.892	0.336	44	Probable Violation	9
11	85600	0.3	1.275	0.235	18		
12	105000	0.3	1.563	0.192	14		
13	128000	0.3	1.906	0.157	14		
14	149000	0.3	2.219	0.135	10		
15	240000	0.3	3.574	0.084	52	Certain Violation	6
16	460000	0.3	6.849	0.044	32		
17	640000	0.3	9.530	0.032	14		
18	911000	0.3	13.565	0.022	2		

3.1. Experimental Setup

All the simulations were created using CloudSim Plus (Filho et al., 2017), a cloud toolkit for generating cloud computing infrastructures and application services, with the sub datasets derived from the WS-DREAM dataset, which consists of real-world QoS evaluation results from 142 users on 4,500 web services over 64 different time slices (Zheng et al., 2014). Under this simulated environment, virtual machines (VMs) resembling cloud resources were made available by the real cloud provider; for example, t2.medium of Amazon EC2 instances (AWS, 2018) was selected for use. This dual-core Intel Xeon 2.49 GHz CPU can execute 63000 million instructions per second (MIPS), with 4096 MB memory and 100Mbps network bandwidth. The number of VMs used was set at one at the start of the simulation as a baseline, and this number was kept constant for all the experiments unless it was being scaled out by decisions made by the SFT algorithm. The baseline for response time and throughput follows the normal transaction workload, as shown in Table 2 (Items 5-9). To simulate the provisioning of both VMs and workloads, time-shared policy (Calheiros et al., 2010) was implemented. In this policy, the processing power (CPU Cores) is concurrently shared by the VMs across the same time frame. For our experiments, each CPU Core was shared by two VMs, each taking up the respective equally divided workload simultaneously. To simulate a faulty VM, CloudSim Plus was used to provide a fault injection class to inject a faulty VM during the simulation runtime.

3.2. The Four Scenarios for Simulation

The experiments conducted were based on four scenarios. The SFT algorithm was executed before and after the simulation of each scenario in order to gather relevant results for comparison

and evaluation; please refer to Figures 4a–4d for the timeline of simulation and submission of workloads.

The first scenario involved submission of a workload (Table 2, Rows 15-18) to the system every 15 seconds for 10 occurrences, as shown in Figure 4a. In this scenario, a preventive measure was deployed to handle the workload causing certain QoS violation without any faulty VM event. This experiment was repeated 30 times, with a total of 270 (10 occurrences excluding the first x 30) random workload submissions, creating a certain QoS violation state in order to test the SFT.

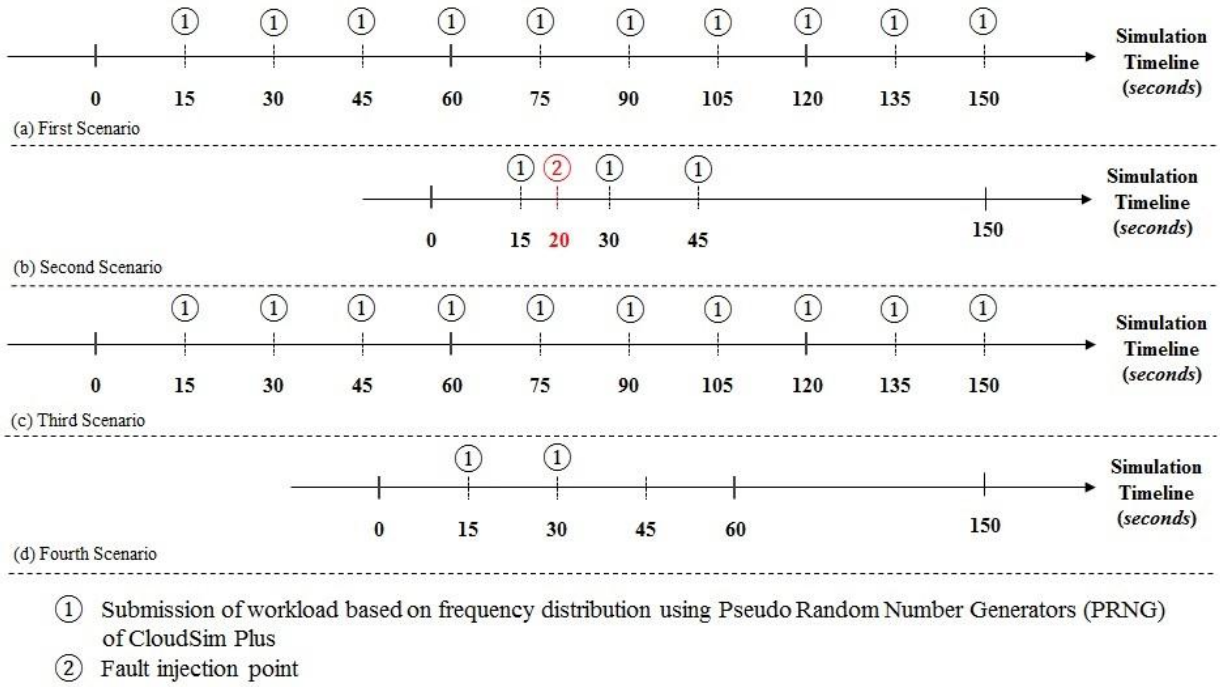

Figure 4 Simulation and workload submission timeline

The second scenario comprised submission of a workload (Table 2, Rows 5-9) to the system every 15 seconds for three occurrences, as shown in Figure 4b. The injection of a faulty VM was deployed at the 20th second of the simulation run time to cause a certain QoS violation condition. In this scenario, VM failure was injected to simulate an event causing a certain QoS violation condition in order to test the capability of the SFT in deploying remedial action to handle the faulty VM. This experiment was repeated 30 times, with random submission of workloads.

In the third scenario, a workload (Table 2, Rows 10-14) was submitted to the system every 15 seconds for 10 occurrences, as shown in Figure 4c. This was to simulate workloads for probable violation events and to test the SFT algorithm in handling preventive measures. The experiment was repeated 30 times, with random submission of workloads.

The fourth scenario involved submission of a workload (Table 2, Rows 1-4) to the system every 15 seconds for two occurrences, as shown in Figure 4d. The aim of this scenario was to simulate workloads in order to investigate the capability of the SFT algorithm in handling over-provisioning cases when the system detects a QoS certainly no violation condition. This experiment was repeated 30 times, with random submission of workloads.

3.3. Performance Evaluation of SFT Algorithm

The performance of the SFT algorithm was evaluated separately for each scenario using performance metrics such as average response time, average throughput, number of VMs applied and effective rate. Average response time represents the mean response time value collected for

each point of time from all the repeated experiments. Average throughput represents the mean throughput value, also collected for each point of time from all repeated experiments. The number of VMs applied indicates the total number of VMs running in the host server. The effectiveness of the algorithm will vary according to the scenario being evaluated. Finally, the effective rate of the algorithm is determined by how successful it is in preventing probable QoS violation or rectifying cases of certain violation. The effective rate can also mean how successful the algorithm is in removing idle VMs when no workload has been submitted for 30 seconds when in a certainly no violation condition.

For deploying preventive measures, the SFT adds two more VMs (scales out) to the hosting server when handling certain violation events without a faulty VM. Conversely, for remedial measures, it applies fault tolerance techniques such as replication when handling certain violation events with a faulty VM. The preventive measure of scaling out (adding one additional VM) to the hosting server is deployed by SFT for probable violation events. This is to achieve the goal of maintaining cloud QoS requirements. Removing an idle VM is executed through SFT when a certainly no violation event is detected and there is no submission of workload for more than 30 seconds. As a result, all possible decision outcomes of probable violation, certain violation and certainly no violation scenarios are fully covered for performance evaluation.

4. RESULTS AND DISCUSSION

This section presents the experimental results based on the simulation of the four scenarios. For the results of the first scenario, please refer to Figures 5a, 5b and 5c, which show average response time, average throughput and number of VMs applied with CPU utilization respectively before and after implementation of the SFT algorithm from all the 30 repeated experiments conducted. As can be seen from Figure 5, the algorithm detected the occurrence of certain violation at time 15 s and started to deploy a preventive measure by adding two more VMs to the hosting server. By balancing the workload with newly added VMs, this therefore prevented any certain violation cases from occurring in subsequent time frames. However, it was observed that on 13 occasions (3 when the workload request was at 640000MIPS and 10 when at 911000MIPS), the SFT was not able to deploy preventive measures to bring the QoS state back to probable violation or normal. Therefore, in general, SFT was effective 95% of the time with regard to preventive measures in the first scenario. This result is still satisfactory, as these two workload requests seldom occur.

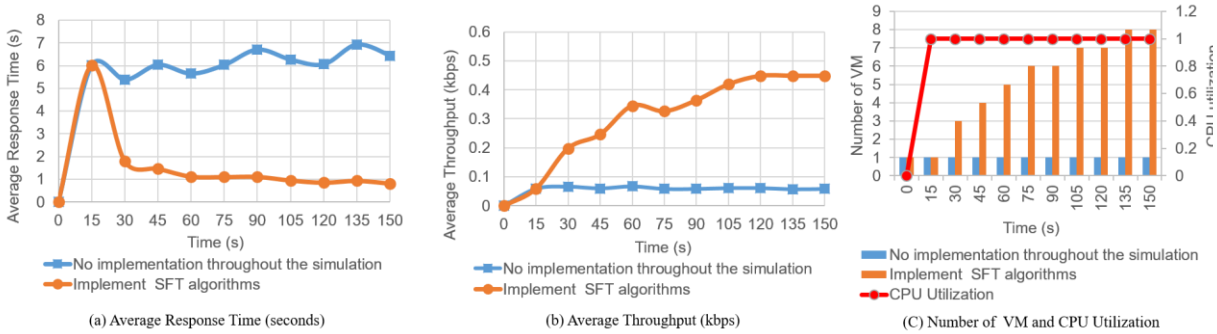

Figure 5 Experimental results for first scenario

The experimental results of the second scenario can be seen in Figures 6a, 6b and 6c, which display the average response time, average throughput and number of VMs applied with CPU utilization respectively before and after implementation of the SFT algorithm in all the repeated experiments conducted. As shown in Figure 6, remedial action was taken to replicate the faulty VM when SFT detected a certain violation occurrence at the 20th second point. Re-submission of the workload attached to the faulty VM was made at the 31st second by SFT when average

response time (Figure 6a), throughput (Figure 6b) and full CPU utilization (Figure 6c) resumed operation. Based on the observations, SFT was able to perform replication for all the faulty VMs, thus achieving a 100% effective rate for remedial action in the case of certain violation events.

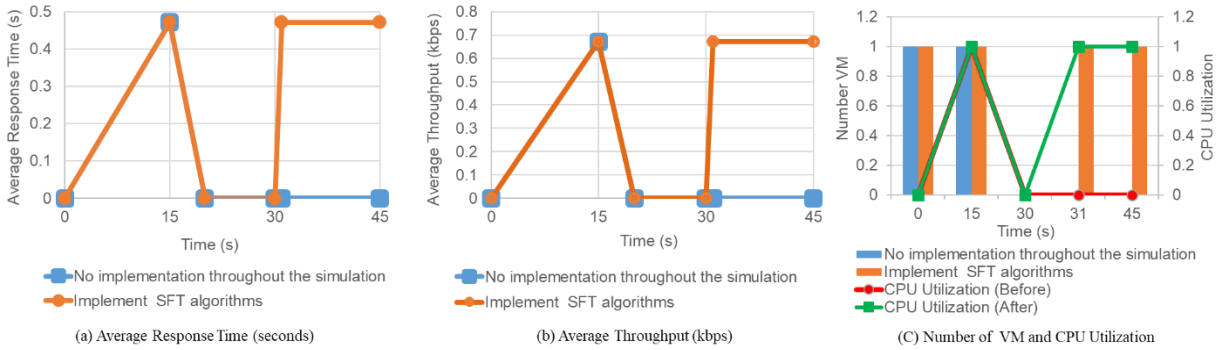

Figure 6 Experimental results for second scenario

The experimental results of the third scenario are shown in Figures 7a and 7b, which display the average response time and average throughput before and after implementation of the SFT algorithm from all the repeated experiments conducted. As can be seen from Figure 7, the algorithm detected the occurrence of probable violation at the 15th second and then deployed a preventive measure by adding an additional VM to the hosting server. Therefore, balancing the workload after the addition of a new VM prevented a further system downgrade to the certain violation condition. However, based on the observations made from all the repeated experiments, it was discovered that the SFT algorithm was not able to deploy preventive measures on 30 occasions when the workload request was at 128000MIPS (15 times) and 149000MIPS (15 times). Therefore, it is considered to be effective close to 90% of the time in relation to preventive measures in the third scenario. This result remains satisfactory, as these two workload requests only contribute to around 2% of probable violation events out of the total of 100%, including other normal, certainly no violation and certain violation events.

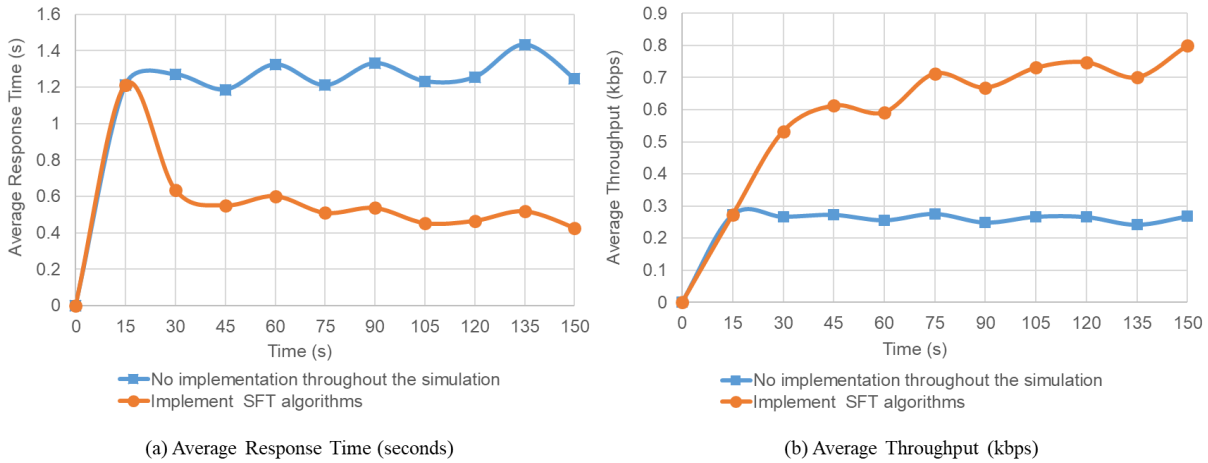

Figure 7 Experimental results for third scenario

For the experimental results of the fourth scenario, please refer to Figure 8, which displays the number of VMs applied before and after implementation of the SFT algorithm in all the repeated experiment conducted. As can be seen from the figure, the SFT algorithm waited for 30 seconds (at the 60th second), only then removing the VM which was in an idle state accordingly. From the observations made for all the repeated experiments, SFT was effective 100% of the time in the fourth scenario.

A point to note is that most of the data centres consist of configurable computing resources, both physical and virtual, which are connected and accessible through broad networks. This allows cloud service providers to pool multiple computing resources together from data centres in different locations to serve multiple consumers using a multi-tenant model, and to scale computing resources according to cloud consumer demand (Mell & Grance, 2011). Hence, to dynamically adjust scaling in or out, the resources should not represent an issue.

However, in our future research, the SFT algorithm should include a feature to intelligently identify heavy or light workloads processed by each cloud application. In this way, heavy workloads could be dynamically allocated to more, or just the right number of, VMs for processing the workload without encountering QoS violation when the cloud platform is already fully occupied by the running of other cloud applications. Likewise, the lighter workloads of other applications could be allocated to fewer, or just the minimum number of, VMs for processing the workload without encountering any QoS violations.

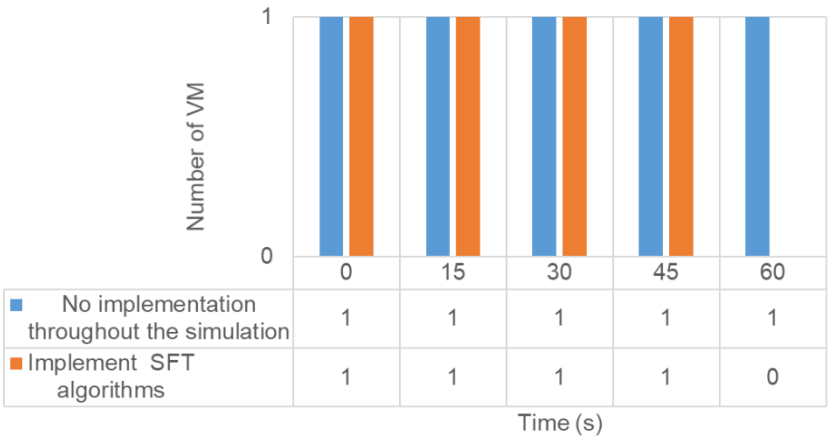

Figure 8 Experimental results for fourth scenario

5. CONCLUSION

In this paper, we have presented the design and implementation of a system that can perform VM-scaling, replication and task retry. We developed a scaling and fault tolerance (SFT) algorithm to deploy preventive measures or take remedial action based on QoS decision outcomes with regard to response time and throughput. Experiments based on four scenarios to measure the effectiveness of the algorithm in handling events such as faulty VMs and over- and under-provisioning were conducted. Our experimental results show that the algorithm was effective 90% to 100% of the time when handling probable violation events using a scaling technique as the preventive measure; when taking remedial action using replication and task re-submission as fault tolerance techniques; and in resolving over-provisioning. The SFT algorithm, together with the 16 decision rules, thus contributes to an additional aspect of detection, prediction, prevention and rectification measures of response time and throughput for cloud QoS violations.

6. REFERENCES

Aruna, L., Aramudhan, M., 2016. Framework for Ranking Service Providers of Federated Cloud Architecture using Fuzzy Sets. *International Journal of Technology*, Volume 7(4), pp. 643–653

AWS (Amazon Web Services), 2018. Amazon EC2 T2 Instances –Amazon Web Services, Inc. Available Online at https://aws.amazon.com/ec2/ instance-types/t2/, Accessed on July 20, 2018

Bardsiri, A.K., Hashemi, S.M., 2014. QoS Metrics for Cloud Computing Services Evaluation. *International Journal of Intelligent Systems and Applications*, Volume 6(12), pp. 27–33

Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R., 2010. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software Practice and Experience*, Volume 41(1), pp. 23–50

Filho, M.C.S., Oliveira, R.L., Monteiro, C.C., Inacio, P.R.M., Freire, M.M., 2017. CloudSim Plus: A Cloud Computing Simulation Framework Pursuing Software Engineering Principles for Improved Modularity, Extensibility and Correctness. *In*: Proceedings of IFIP/IEEE International Symposium on Integrated Network Management, 8-12 May, 2017, pp.400–406

Ganesh, A., Sandhya, M., Shankar, S., 2014. A Study on Fault Tolerance Methods in Cloud Computing. *In*: Proceedings of IEEE International Advance Computing Conference (IACC), pp.844–849

Kuperberg, M., Herbst, N., Kistowski, J.V., Reussner, R., 2011. Defining and Quantifying Elasticity of Resources in Cloud Computing and Scalable Platforms. *In*: Karlsruhe Reports in Informatics, Volume 16, pp. 1–17

Lehrig, S., Eikerling, H. Becker, S., 2015. Scalability, Elasticity, and Efficiency in Cloud Computing: A Systematic Literature Review of Definitions and Metrics. *In*: Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures - QoSA '15, pp. 83–92

Mell, P., Grance, T., 2011. *The NIST Definition of Cloud Computing*. NIST Special Publication 800-145, pp. 1–3

Wong, T.S., Chan, G.Y., Chua, F.F., 2018. A Machine Learning Model for Detection and Prediction of Cloud Quality of Service Violation. *In*: International Conference on Computational Science and Its Applications (ICCSA), LNCS, pp. 498–513

Wong, T.S., Chan, G.Y., Chua, F. F., 2019. Adaptive Preventive and Remedial Measures in Resolving Cloud Quality of Service Violation. *In*: Proceedings of IEEE 33rd International Conference on Information Networking (ICOIN 2019), 9-11 January 2019, Kuala Lumpur, Malaysia, pp. 473–479

Zheng, Z., Zhang, Y., Lyu, M.R., 2014. Investigating QoS of Real-World Web Services. *In:* IEEE Transactions on Services Computing, Volume 7(1), pp.32–39