Volume 5, Issue 8, August 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Verification of Agent Based System for Monitoring and Controlling SLA in CloudSim

Sarwan Singh
Research Scholar,
Sai Nath University, Jharkhand, India

Manish Arora
Joint Director (Systems)
NIELIT, Chandigarh, India

Abstract*—Cloud computing is an on-demand computing model where Cloud user pays in per usage mode. Cloud user avails services and Cloud service provider provides services. The services are provided on mutually agreed terms and conditions called Service Level Agreements (SLA). These terms and conditions are to be monitored during runtime to avoid any breach of trust between Cloud user and Cloud service provider. A software agent based monitoring tool is implemented and verified to monitor the SLA parameters that includes resources being allocated during runtime and compares runtime value with the agreed parameter values. This paper presents the verification agent developed for Monitoring and controlling SLA, MACSLA. The working of agent is verified using CloudSim where three virtual machines run. The agent generates alerts in case of any violation of resource allocation. This will help both Cloud Service Provider and Cloud user in maintaining trust in the Cloud services being offered.

***Keywords*—Cloud Computing, Service Level Agreements (SLA), CloudSim, Monitoring & Controlling SLA Agent, JADE**

I. INTRODUCTION

Cloud computing is an essential utility in today's computing world. In this on-demand computing model resources are made available on pay per use basis, where trust between the Cloud user and Cloud service provider is maintained by Service Level Agreements (SLA). Any violation in SLA parameters will impact the trust between the two. At runtime various SLA parameters like RAM, Bandwidth MIPS, etc. assigned to Virtual Machine (VM) are monitored by a dynamic agent based system [1] and alerts are generated for any violation noted during run time.

The agent based system for monitoring and controlling SLA is implemented using JADE and CloudSim with back end support database, MySQL. It is difficult to analyse the performance of agent in actual environment, as it involves usage of shared resources with lot of Hardware. Thus simulation tools are becoming important and necessary for evaluation of Cloud computing model and agent based model for calculating efficiency, reliability and performance.

The verification of an agent is very important to ascertain that the design of agent is as per the requirements and it is performing the desired task optimally. The verification process of the agent requires different test cases. In view of this, different test cases have been identified and used in verifying the working of agent for the purpose of monitoring and controlling Multi level SLA.

The paper has been organized into five sections; section II reviews the background and related research, section III briefs the verification approach, section IV does the analysis of output, followed by the conclusion and scope for future work in section V.

II. BACKGROUND & RELATED RESEARCH

a) Monitoring and Controlling SLA using Agent

Service Level Agreement (SLA) is an important document of agreement which maintains the trust relationship between Cloud service provider and Cloud user. This not only describes various SLA parameters whereas guarantees them with remedial actions in case of violation. For this cause a software Agent based methodology proves beneficial, and hence agent based *Monitoring and Controlling SLA (MACSLA)* is modelled [2] [3]. The Monitoring and Controlling Agent (MCA) which orchestrate the whole process and provides graphical interface to Cloud Administrator/User. The MCA has two main processes underneath i.e. Term Collector (TC) and Term Monitor(TM). The main job of Term Collector (TC) is to collect agreed SLA parameters and their predesigned value from the database. On the other hand, Term Monitor(TM) will collect SLA parameter values while Virtual Machine (VM) is running. The MCA will compare the values collected by TC & TM and generate alerts accordingly. The database which stores the pre-designed/pre-negotiated SLA parameter values is implemented using MySQL database [4].

b) Implementation of MACSLA Agent

MACSLA agent is implemented using Cloud simulation tool, CloudSim. CloudSim is a framework which allows, modelling, simulation and experimentation with the Cloud computing infrastructure models [5]. The high level architecture of various entities involved in CloudSim is shown in Figure 1. The Cloud user sends the service request to

Data Center Borker which coordinates with *Cloud Controll*er to create instance of *Host* in *DataCenter*. Cloud Controller may coordinate more than one Data Center. Host has multiple *VMs* running over it with different number of *Cloudlets*.

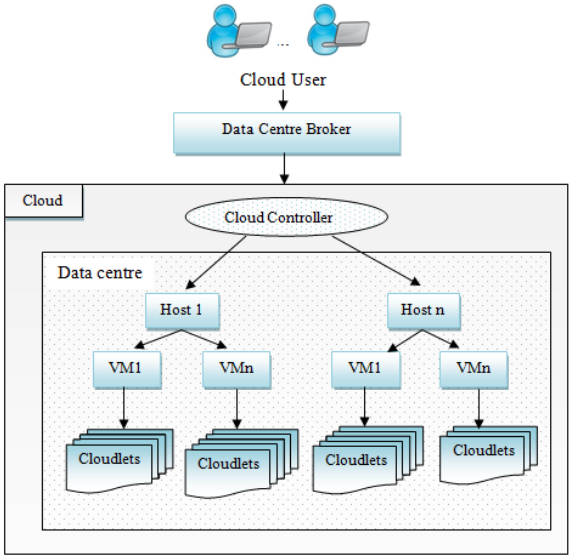


Figure 1 : Architecture of various entities in CloudSim

There are many classes like Cloudlet, Host, Virtual Machine, etc. in CloudSim which serve as building blocks of CloudSim simulator. In MACSLA, classes available with CloudSim simulator are extended to experiment the violation of SLA. Commonly used classes are discussed below:

1) **Cloudlet**: This class models the Cloud-based application, representing pre-assigned instruction length and amount of data transfer. According to the load assigned to Cloudlet, it takes time to complete its computational task.
2) **Data Centre:** In a Cloud computing environment the infrastructure level services offered by Cloud service provider are modelled by this class. This class is used to implement various policies for allocation of bandwidth, memory, and storage devices.
3) **Data Center Broker:** This class mediates between user and Cloud service providers depending on users Quality of Service (QoS) requirements. This class is used to experiment application placement policies.
4) **Host**: This class represents a physical machine with memory, storage, number of cores, etc.
5) **Virtual Machine:** This class models an instance of a VM, which runs on Host, along with other VM (is any). Every VM has its own memory, storage, processor, etc along with provisioning policies.

III. VERIFICATION OF MACSLA

The implemented MACSLA is required to be verified for its functionality. The verification includes identification of test cases, generation of test data and verification the functionality of agent implemented. The implemented agent is verified by integrating in simulator, CloudSim. The verification steps are detailed below :

i) Identification of Test Cases

For the verification purpose three Virtual Machines (VM) are created and deployed on two Hosts. The configurations of VMs and Hots are described in Table 1.

Table 1: configuration details of Virtual Machines and Host

	VM1	VM2	VM3
RAM (MB)	3740	613	870
Core	1	1	1
MIPS	2500	500	1000
Bandwidth	1,00,000	1,00,000	1,00,000

	Host1	Host2
RAM (MB)	4096	4096
Core	2	2
MIPS	1860	2660
Bandwidth	10,00,000	10,00,000

Based on VMs and Hosts, eight test cases are identified to verify MACSLA. The outcome of each test case would be either no violation or violation. The load is varied in each test case to know the outcome. The eight different test cases and their outcome described in Table 2:

Table 2 : Test cases for three Virtual Machines

Test Case	VM1	VM2	VM3
1	Normal Load *No SLA Violation*	Normal Load *No SLA Violation*	Normal Load *No SLA Violation*
2	Over Load *SLA Violation*	Normal Load *No SLA Violation*	Normal Load *No SLA Violation*

3	Normal Load *No SLA Violation*	Over Load *SLA Violation*	Normal Load *No SLA Violation*
4	Normal Load *No SLA Violation*	Normal Load *No SLA Violation*	Over Load *SLA Violation*
5	Over Load *SLA Violation*	Over Load *SLA Violation*	Normal Load *No SLA Violation*
6	Over Load *SLA Violation*	Normal Load *No SLA Violation*	Over Load *SLA Violation*
7	Normal Load *No SLA Violation*	Over Load *SLA Violation*	Over Load *SLA Violation*
8	Over Load *SLA Violation*	Over Load *SLA Violation*	Over Load *SLA Violation*

ii) Test Data

The test data is generated in view of test cases identified and desired outcome. The test data is generated is such a way that it should fall in one of test case identified. The test data generated (randomly) shown in Table 3 including the load on each virtual machine in the form of cloudlets.

Table 3 : Test data for each VM in different Test Cases

Test Case	**VM1** (cloudlet count)	**VM2** (cloudlet count)	**VM3** (cloudlet count)	**Remarks**
1	7	3	5	No violation
2	16	3	5	Violation in VM1
3	7	12	5	Violation in VM2
4	7	3	12	Violation in VM3
5	13	12	5	Violation in VM1 and VM2
6	13	3	14	Violation in VM1 and VM3
7	7	12	12	Violation in VM2 and VM3
8	13	12	10	Violation in VM1, VM2 & VM3

iii) Verifying Functionality of Agent

The MACSLA agent [6] during runtime assigns the Cloudlets to the Virtual Machine(VM) .The code snippet to assign cloudlets to VM is given in Table 4.

Table 4: java code snippet to assign cloudlet to VM

```
...
// setCloudlettoVM(cloudletList, VMID, start offset in cloudletList, // end offset in cloudletList);
setCloudlettoVM(cloudletList, 1, 1, 7);
setCloudlettoVM(cloudletList, 3, 8, 12);
setCloudlettoVM(cloudletList, 2, 13, 15);
...
```

The load on VM 1 is increased by assigning more than 7 cloudlets as shown in the code snippet in Table 5.

Table 5: java code snippet to assign cloudlet to VM

```
...
// setCloudlettoVM(cloudletList, VMID, start offset in cloudletList, // end offset in cloudletList);
setCloudlettoVM(cloudletList, 1, 1, 7);
setCloudlettoVM(cloudletList, 3, 8, 12);
setCloudlettoVM(cloudletList, 2, 13, 15);
setCloudlettoVM(cloudletList, 1, 16, 24);
...
```

IV. ANALYZING THE OUTPUT

The output after applying test data for Test Case 1 is analysed. The cloudlets assigned to each of the VM are given in Figure 2. Under this scenario, none of the VM violates the agreed SLA parameters. The Figure 2 has three graphs w.r.t time for each VM. The first graph is for RAM vs Timestamp, second is for Mips vs Timestamp and third is Bw vs Timestamp. Each graph has two parameters plotted, one representing agreed SLA parameters and other for SLA parameters being allocated to VM on runtime.

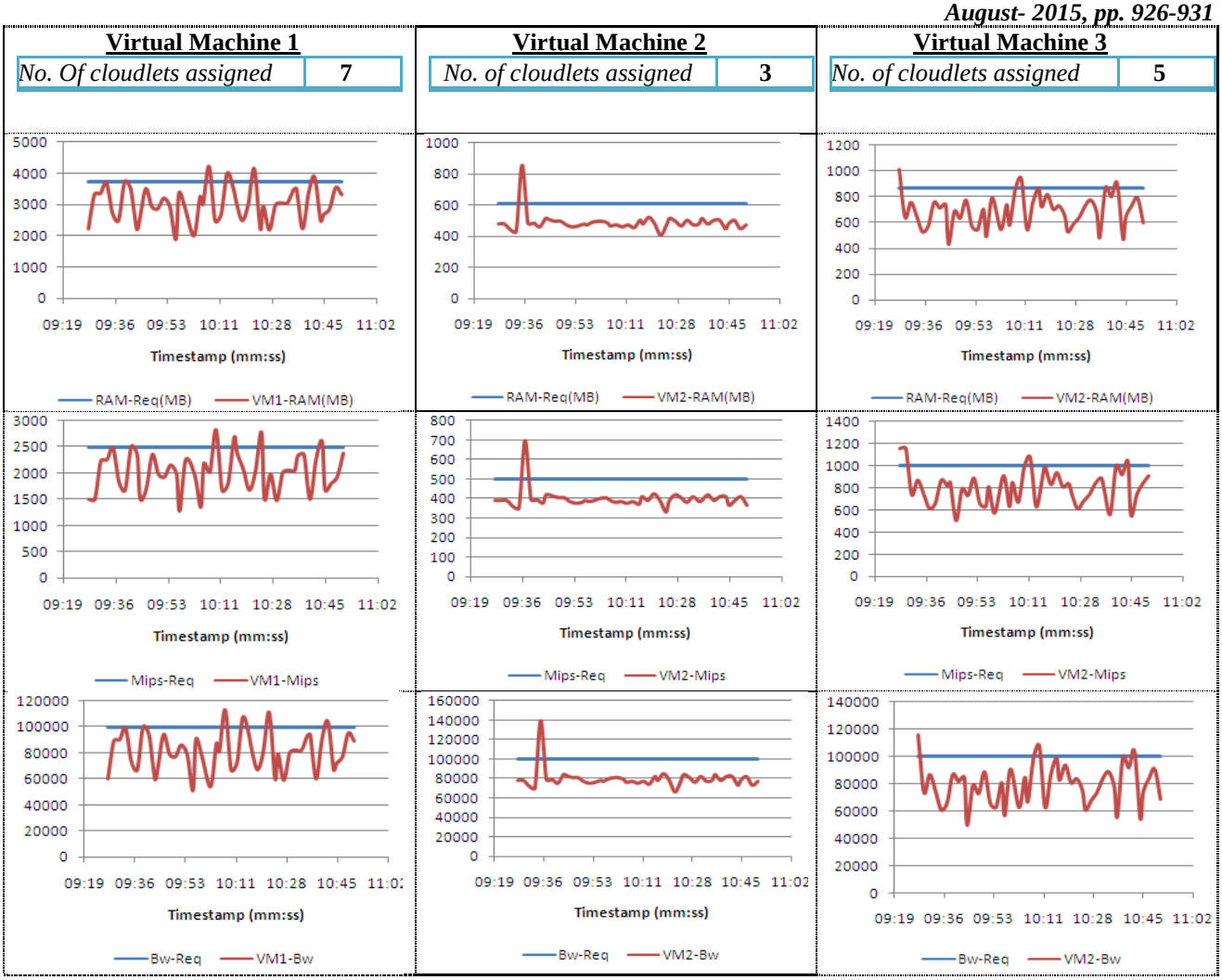


Figure 2 : Test case 1- VM1, VM2,VM3 are not violating SLA

Later, the load in VM1 is increased to verify Test Case 2. The effects of assigning 9 more cloudlets are depicted in Figure 3. The VM2 and VM3 has same load/cloudlets running on them, therefore SLA is not violated and remains as before.

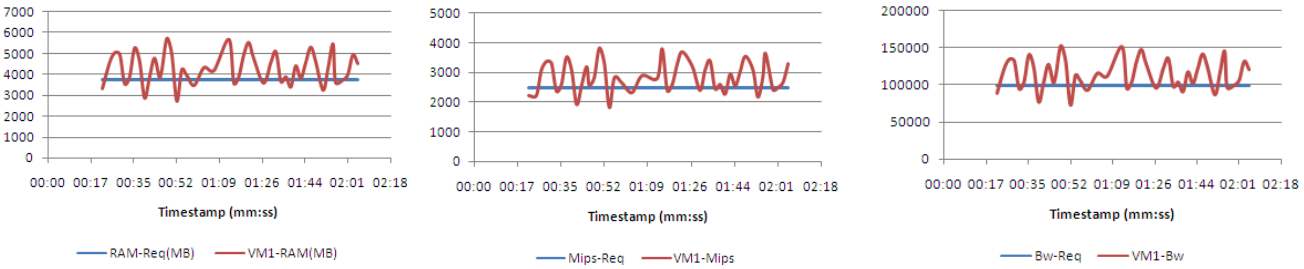
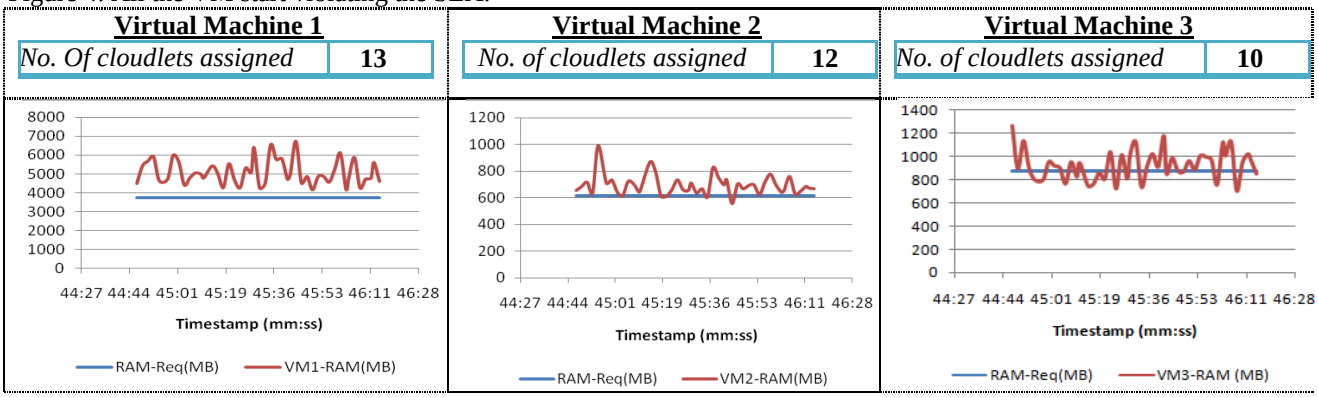


Figure 3 : Test case 2 – VM1 violates the SLA

Similarly, load on VMs are increased as per the Test Cases and further results are analysed as shown in the Figure 4. All the VM start violating the SLA.



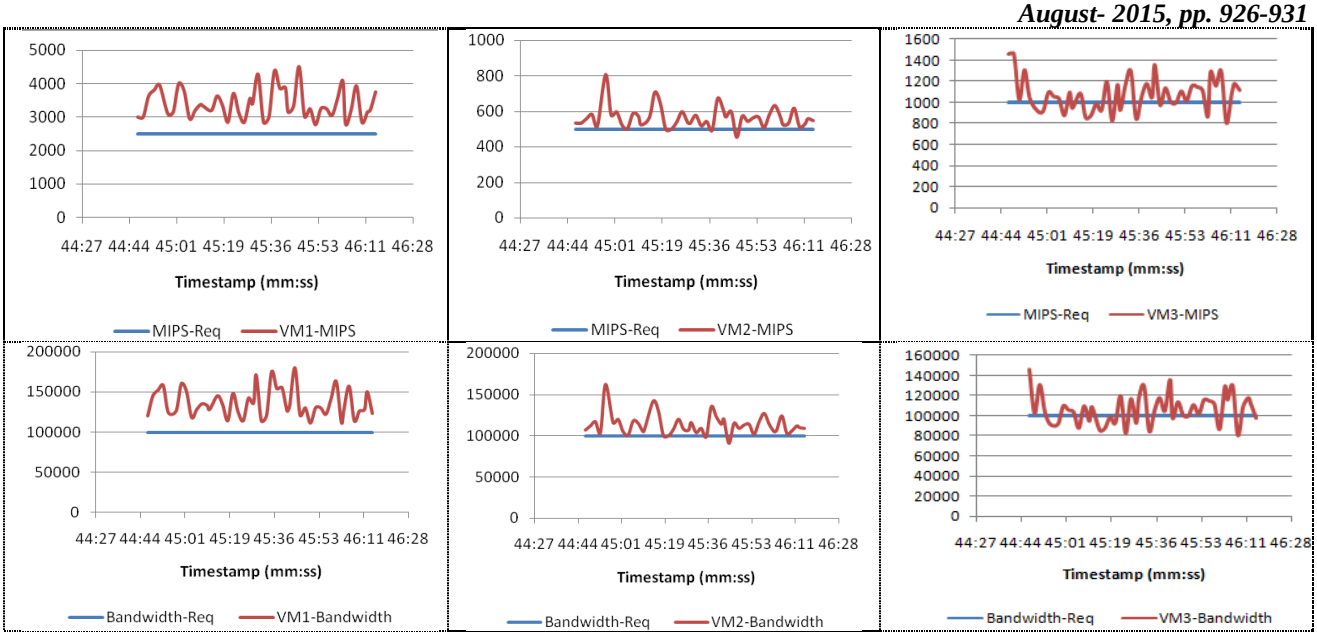


Figure 4 : Test case 8-VM1, VM2,VM3 are violating SLA

The user interface created in JADE [7] as shown in Figure 5 prompts the various SLA parameter values being collected by Term Monitor(TM) and compares it with desired values saved in database by Term Collector (TC). The column showing *Required Value* is saved in database by TC, the *Demanded Value* is saved in database by TM and if any violation the third column shows *true* along with background color changing to red, otherwise it shows false with white background.

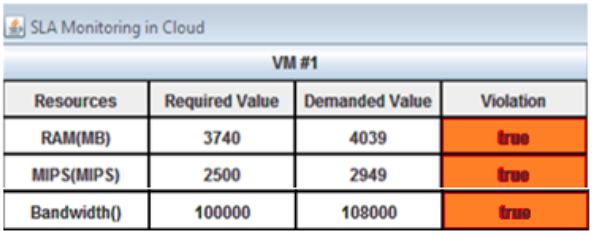


Figure 5 : Agent UI developed in JADE

The eight test cases identified are verified in the simulated environment using CloudSim. It is found that the SLA violation takes place with increase in load on Virtual Machine in context to parameters under observation like RAM, MIPS, and Bandwidth. The complete result for all the test cases with load/Cloudlets being assigned to each VM is shown in Table 6.

Table 6 : Result of SLA violation in different test cases for three Virtual Machines

Test Case	VM1				VM2				VM3			
	Load / Cloudlets	*SLA Violation*			*Load / Cloudlets*	*SLA Violation*			*Load / Cloudlets*	*SLA Violation*		
		RAM	MIPS	Bandwidth		RAM	MIPS	Bandwidth		RAM	MIPS	Bandwidth
1	*7*	No	No	No	3	No	No	No	5	No	No	No
2	*16*	Yes	Yes	Yes	3	No	No	No	5	No	No	No
3	*7*	No	No	No	12	Yes	Yes	Yes	5	No	No	No
4	*7*	No	No	No	3	No	No	No	12	Yes	Yes	Yes
5	*13*	Yes	Yes	Yes	12	Yes	Yes	Yes	5	No	No	No
6	*13*	Yes	Yes	Yes	3	No	No	No	14	Yes	Yes	Yes
7	*7*	No	No	No	12	Yes	Yes	Yes	12	Yes	Yes	Yes
8	*13*	Yes	Yes	Yes	12	Yes	Yes	Yes	10	Yes	Yes	Yes

V. CONCLUSION

The Monitoring And Controlling SLA (MACSLA) has proved that any violation in SLA is highlighted as well as recorded in database which assures Cloud user that he is getting what has been demanded and maintains trust relationship between Cloud user and Cloud service provider.

REFERENCES

[1] Sarwan Singh, Manish Arora , "Implementation of Agent Based System for Monitoring and Controlling SLA in Cloud Environment", IJEBEA, August 2015

[2] Sarwan Singh, Manish Arora , "Design of Agent Based System for Monitoring and Controlling SLA in Cloud Environment", IJATES, Volume No 03, Special Issue No. 01, March 2015

[3] L. Tasquier, S. Venticinque, R. Aversa, and B. Di, "Agent Based Application Tools for Cloud Provisioning and Management", 3rd International Conference on Cloud Computing (CLOUDCOMP 2012), Wien, Austria, September 24–26, 2012

[4] MySQL, Available at https://www.mysql.com/ accessed on August 16, 2015

[5] Rajkumar Buyya , Rajiv Ranjan, Rodrigo N. Calheiros "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities" , {raj, rodrigoc}@csse.unimelb.edu.au, rajiv@unsw.edu.au

[6] K. M. Sim, "Agent-based Cloud Computing", IEEE Transactions On Services Computing, Vol.5, No.4, pp.564,577, Fourth Quarter 2012.

[7] Java Agent Development Framework, Available at http://jade.tilab.com/, accessed on August 16, 2015

