International Journal of Digital Information and Wireless Communications (IJDIWC) 4(1): 43-52
The Society of Digital Information and Wireless Communications, 2014 (ISSN: 2225-658X)

A Hierarchical Self-Healing SLA for Cloud Computing

Ahmad Mosallanejad, Rodziah Atan, Masrah Azmi Murad, Rusli Abdullah,
Faculty of Computer Science and Information Technology, UPM, Malaysia
ahmad.upm@gmail.com, {rodziah, masrah, rusli}@upm.edu.my

ABSTRACT

The service level agreement (SLA) is a mutual contract between the service provider and consumer which determines the agreed service level objective (SLO). The common SLA is a plain documental agreement without any relation to other dependent SLAs during the different layers of cloud computing. Hence, the cloud computing environment needs the hierarchical and autonomic SLA. This paper proposes the SH-SLA model to generate a hierarchical self-healing SLA in cloud computing. The self-healing ability contains the SLA monitoring, violation detecting and violation reacting processes. In SH-SLA, the related SLAs communicate with each other hierarchically. The SLA would be able to check its QoS and notify the recent status to dependent SLAs. Furthermore, SH-SLA could prevent or propagate the notified violations by an urgent reaction. Consequently, the service providers have a great chance to prevent the violated SLA before sensing by end users. The SH-SLA model is simulated and the experiment results have presented the violation detection and reaction abilities of the proposed model in cloud computing. Besides, the end users meet the lesser violations in SH-SLA than the common SLA.

KEYWORDS

service level agreement, cloud computing, self-monitoring SLA, hierarchical SLA, self-healing SLA

1. INTRODUCTION

Currently, cloud computing is an excited topic in both business and research area [1, 2]. Many service providers and consumers have dealings among the different layers of cloud computing consist of software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) [3, 4]. Service level agreement (SLA) is a contract between the service provider and consumer to determine the quality and functionality of agreed services [5]. So, SLA is a fundamental document which covers the service level objective (SLO), their attributes and metrics. This agreement is the basis of relations between service provider and consumer in different layers of cloud computing [6, 7]. Both service provider and consumer need to monitor the agreed services for valdating the SLA [8].

Currently, the most of SLAs are a single XML document which covers functionalities and quality of services (QoS) between specific service provider and consumer. They do not have a connection with other related SLAs while cloud computing is a hierarchical environment. SLAs in SaaS will be failed if its related SLA in PaaS is violated. So, lack of an effective relations between dependent SLAs is a vital challenge which makes the SLA management system inefficient. To have an effective SLA monitoring and a violation reacting system, a hierarchical SLA model is needed based on cloud computing nature [9, 10].

Three types of SLA monitoring systems are available including provider side, consumer side and trusted party side SLA monitoring system [11-13]. Each of these approaches has its own advantages and disadvantages but all of them have a centralized monitoring system. In centralized monitoring system, the specific SLA management center in provider, consumer or trusted party side is in charge for SLA monitoring and violation detecting. The most of SLA monitoring systems are applied in grid computing and service oriented architecture (SOA) environments therefore they are not compatible enough for cloud computing [14, 15]. Furthermore, the most of SLA management and monitoring systems only present the report of SLA violations without any reaction. Likewise, The current structure of SLA and SLA monitoring system is not suitable for cloud computing nature without self-healing SLA feature [16, 17]. In order to have the reliable cloud

computing services, an effective SLA monitoring and the violation reacting model is unavoidable.

In this study, hierarchical self-healing (SH)-SLA model is proposed to enforce the SLA monitoring and violation reacting in cloud computing. In SH-SLA, each SLA has connected to their related SLAs in different layers of cloud computing so each SLA is able to notify its status to other related SLAs. Additionally, each SLA has the ability of monitoring and reacting independently. The SLA can assess the monitoring results itself and notify to the dependent SLAs.

The rest of the paper is organized as follows: The related works are presented in Section 2. Then, the 3rd Section describes the SH-SLA compositions. Section 4 presents the experiment method then the proposed model is evaluated in Section 5. Finally, the conclusion and future work are presented in 6th Section.

2. RELATED WORKS

Many studies have investigated the SLA management systems, but only a few of them covered the SLA enforcement in cloud computing. The most of related works applied the SLA models from other environments such as SOA and grid computing into cloud computing without enough considering to the cloud requiermnets. The main parts of a self-healing system consist of monitoring and reacting procedures.

Some related works such as QoSMONaaS [18] and SLAMonADA [19] focused on SLA monitoring system to detect the SLA violations without considering to the reaction process. These proposed platforms measured the QoS value in running time to release a report about SLA validation. A. Kertesz *et. al*. (2011) and P. Varalakshmi *et. al.* (2011) also proposed the SLA monitoring system to detect the SLA violations [20] [21]. They count the detected violations in order to assess the penalty cost which provider should pay to the service consumer. Some other related works are such as [22], [23] and [24] checked the SLA validation to manage the provider resources effectively. They also did not consider the violation reaction issue in their studies.

There are a few related works to react against SLA violations in cloud computing. FOSII [25], QU4DS [26] and LoM2HiS [27] which are significant in this work. All of these frameworks engaged the monitoring, analysis, planning and execution (MAPE) loop to react against violations [28]. They applied the centralized monitoring and reacting systems in cloud computing, for any ecosystem deploying it. They periodically collected the QoS values from could infrastructure and compare the collected data with SLO. If any violation is detected, the reaction plan is designed based on historical knowledge. Finally the reaction is executed by resource management system. Reaction plan was normally the provider resource reconfiguration or Virtual Machines (VM) replacement. These proposed healing systems are not suitable for agile reactions due to several separated procedures which are to be done sequentially to react against detected violations. Furthermore, a central system is enforced to monitor all SLAs and react against all detected violations. These events are forecasted to be hard for an education ecosystem in terms of maintaining and adjusting to suitable services and platforms.

This literature review has illustrated that the related works and proposed solutions did not suitably cover the cloud computing requirements. The most of related works have transmitted the SLA monitoring framework into cloud computing from SOA and grid computing. Although a few effective SLA monitoring and reacting mechanisms have presented, they have followed the common SLA structure yet.

3. SH-SLA MODEL

Having an effective self-healing SLA in cloud computing, SH-SLA model is proposed including innovative SLA structure, effective SLA monitoring and reacting methods. These abilities need an infrastructure in each service provider to work properly. Although these facilities are completely integrated, they are introduced in next sub-sections separately.

3.1. Self-healing SLA Architecture

To have a hierarchical self-healing SLA, an especial architecture is needed. Each service provider, in different layers of cloud computing, needs to have fundamental components as illustrated in Figure 1. The lower layer (LL) port and upper layer (UL) port are the communication gates to other service providers and service consumers in lower and upper layers of cloud computing. First of all, the negotiation management system specifies the SLA features after the negotiation process. Actually, the SH-SLA is the main output of negotiation management system then SH-SLA should be run. Next, the data collector passes the relevant metrics data from resources and lower layer notifications to the SH-SLA. Finally the running SH-SLA assesses the SLA attributes value and records them to the monitoring warehouse. Moreover, any necessary reaction could be done by the resource manager and notification could be sent to the upper layer consumers.

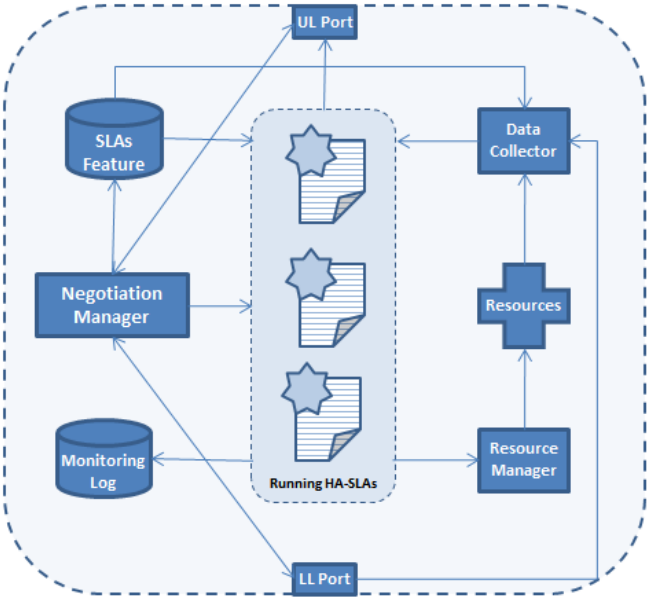

Figure 1. Self-healing SLA Architecture

3.2. Hierarchical SLA in Cloud

This model suggests the hierarchical SLA as a foundation of SLA monitoring and reacting processes. SLAs are the basis of all interactions between service providers and consumers which should be inspected in a monitoring system. Current SLA is a document including the information of service functionalities and QoS. Proposed SH-SLA has two important contributions versus common SLAs: firstly it defines the hierarchical relations secondly this is a self-healing SLA. It is hierarchical because the dependent SLAs are connected to each other. It is self-healing SLA because the both monitoring and reacting functions are allocated inside the SH-SLA. Figure 2 has depicted the common SLA and SH-SLA to present their differences. The current SLA does not have any fields for connecting to the related SLAs while they need to have a hierarchical structure in cloud computing during the different layers. Current SLA, introduced in related works, is an isolated document but SH-SLA is a relational contract.

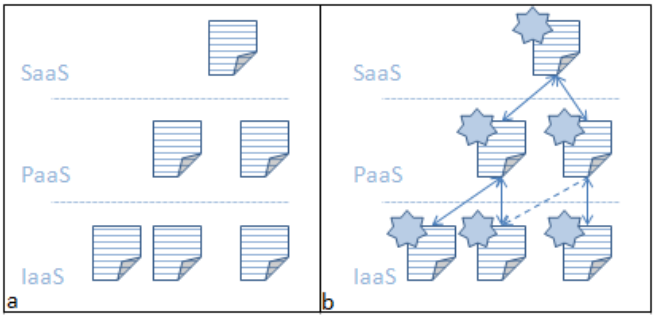

Figure 2. a) Current SLA in cloud computing b) SH-SLA overview

Many SLAs in different layers of cloud computing are contracted. Figure 2 has illustrated some related SLAs during SaaS, PaaS and IaaS which the upper layer SLAs relay to the related lower layer SLAs. If any SLA in IaaS be violated, all dependent SLAs in PaaS and SaaS will be failed. Figure 2a has shown the common SLA in cloud computing which they are not designed to have a hierarchical relation between related SLAs. A few developed frameworks have tried to build these connections by their management system but they are not reasonable when SH-SLA could provide a hierarchical self-healing SLA independently. Figure 2b has displayed the SH-SLA relations which the related upper and lower layer SLAs have been linked in each SLA contents. Moreover some reserved relations are defined for urgent invocation against critical SLA

violation. This reserved relation is illustrated in Figure 1b as a dash line.

3.3. SLA Monitoring Procedures

The SLA monitoring is an important activity for both service provider and service consumer. The service provider utilizes the SLA monitoring systems to manage and economize their resources. On the other hand, service consumer wants to confirm the agreed QoS in SLA. Besides, the SLA monitoring process is the first part of the self-healing SLA to detect the SLA violations. Related monitoring frameworks are discussed in literature review which most of them had a central monitoring approach. In contrast, the proposed SH-SLA has located the monitoring function in each SLA as a part of the distributed monitoring framework. Each monitoring function evaluates the current value of attributes based on their metrics and formula. The monitoring function returns the notification consists of attributes state and their value. They could be recorded in provider or consumer side and also could be used for any relevant reactions.

SH-SLA model changes the passive SLA document to the active SLA identity. Figure 2b insists on self-healing ability of each SLA by a star icon. In each SLA, the star icon indicates to the all operations of SLA such as monitoring and reacting functions. Each SLA could manage, monitor and react by itself. Actually, SLA monitoring and reacting methods are the scope of this research while the SH-SLA structure has the ability of other operations which they will be investigated in future works.

3.4. SLA Reacting Procedures

In proposed method, each service has its own monitoring and reacting instance as shown in Figure 3. SLA contents include attribute, SLO, threshold and related SLAs address. The threshold value is utilized to prevent the SLA violations before happening. The address of related SLAs is used to notify any emergency alert into relevant clients and providers. The attribute listener receives the specific QoS value related to the current SLA from data collector. The QoS value is compared with SLO which is recorded in SLA contents. If the QoS value exceeds the threshold value, the violation reacting procedure will be activated.

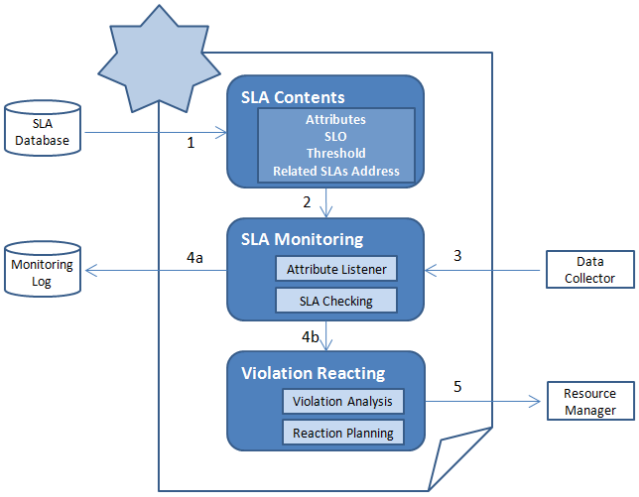

Figure 3. Self-healing components

The violation reaction method is proposed to be as part of HA-SLA. Three Reaction (3R) strategies are defined for violation reaction including: internal resource reconfiguration, external service invocation, and detected violation propagation. If notified QoS from monitoring part exceeds threshold value, it will be propagated into all related consumers and providers as a violated or critical situation. The alerts propagation can help the upper layer providers to react against the notified violation earlier. Sequentially, current provider tries to react against detected violation by internal resource reconfiguration or external service invocation. First, the provider applies the restarting, replacing and reconfiguring methods to the software and hardware resources in order to revive the violated service. If this reaction was unsuccessful, the provider will invoke the external service from another provider which has a same functionality with the violated service. Indeed, the external service is reserved as a spare service to respond the consumers in violation period.

3.5. SH-SLA Developement

The current SLA is a plain agreement document but proposed SH-SLA model is a self-healing SLA including monitoring and reacting

functionalities. The SH-SLA structure code has illustrated in Figure 3 as a first building block of hierarchical self-healing SLA. In this version, some SLA features are avoided because they are not necessary for monitoring and reacting framework and they are out of this paper scope. The SLA attribute structure is defined in lines from 1 to 10 including the definition of metrics, formula, agreed value and threshold value. The threshold value is assigned for agile reaction against any probable violation. Each attribute is able to assess its validity by assessor function as illustrated in Figure 4, line 11. Actually, the array of these attributes is a part of the SH-SLA (Figure4, line 20). Lower layer, upper layer and reserved providers are defined (from 21st to 23rd line) to have an effective hierarchical relation to other SLAs. Figure 4 (from 25th to 33rd line) has shown the functional parts of the SH-SLA consist of monitoring and reacting tasks. Practically, each SH-SLA can monitor itself and react against any violations. This is a significant contribution of the SH-SLA model as a real self-healing SLA.

```
1.  struct Attribute
2.  {
3.     int code;
4.     string name;
5.     Metric[] metrics;
6.     string formula;
7.     float agreedV;
8.     char agreedOperation;
9.     float criticalV;
10.
11.    float assessor()
12.    {
13.      …
14.    }
15. }
16. public struct HA_SLA
17. {
18.    int code;
19.    string name;
20.    Attribute[] attributes;
21.    int[] lowerSLAs;
22.    int[] emergency;
23.    int[] upperSLAs;
24.
25.    Notification[] Monitoring()
26.    {
27.      …
28.    }
29.    Notification[] Reacting()
30.    {
31.      …
32.    }
33. }
```

Figure 4. SH-SLA structure

Normally, the SLA monitoring function checks the value of attributes when the user invokes the service. So the SH-SLA monitoring procedure is running per each service invocation. The monitoring function algorithm is presented in Figure 5.

```
Notification[] Monitoring()
        {
Notification[] notify = new Notification[attributes.Count];
Boolean ReactState =false;

for (int i = 0; i <attributes.Count; i++)
            {
                notify[i] = new Notification();
                notify[i].senderCode= CurrentSLAcode;
                notify[i].AttributeCode = attributes[i].code;
                notify[i].AttributeV = attributes[i].assessor();

if (notify[i].AttributeV ∉ attributes[i].agreedV)
                {
                    notify[i].state = -1;
                    ReactState= true;
                }
else if ( notify[i].AttributeV ∈ attributes[i].criticalV)
                {
                    notify[i].state = 0;
                    ReactState= true;
                }
else if (notify[i].AttributeV ∈ attributes[i].agreedV)
                    notify[i].state = 1;
            }

if( ReactState == true )
                    Reacting(notify);

MonitoringLogDatabase.Insert(notify);

return notify;
        }
```

Figure 5. The monitoring function algorithm

The monitoring function takes the metrics value and the SLA status of the lower layer as a parameter from the data collector. After the SLA attributes measuring, the results are notified to the upper layer SLA. Moreover, any detected violation and critical value are passed to the reacting function to prevent or propagate them. The violation prevention method could follow either internal or external reaction strategies. The service provider migration and the resource replacing are the samples of external and internal reactions respectively.

4. METHODOLOGY

The SH-SLA model, described in Section 3, is simulated to evaluate the proposed model. Figure 6 has presented the simulated scenario based on SH-SLA usage in cloud computing. The predefined SLAs from 1 to 4 are the agreements in different layers of cloud computing. Each SLA is a contract about a specificr service between service provider and consumer. SLA1 is a contract

between IaaS provider and PaaS vendor as a client. Service of SLA4 is also located in IaaS but as a reserved service for emergency invocations. SLA2 is a contract between PaaS provider and SaaS vendor as a client. Finally SLA1 is an agreement between SaaS provider and the end user. SLA3 is dependent on SLA2 moreover SLA2 is dependent on SLA1 hierarchically. In this scenario, each SLA includes respond time and throughput attributes. These attributes are selected because they are the popular attributes which many researchers already focused on them such as [29], [30], [31] and [32]. As described, the simulated SLAs are depending on each other hierarchically.

This study tries to evaluate the simulated SH-SLA by within-subjects design of experimental methodology used by Emeakaroha *et al.* [33]. In this method, the effects of both SH-SLA and plain SLA are observed separately on the same data. This experimental design is selected because this research attempts to compare the effects of two different treatments, SH-SLA and common SLA, on the same situation. Finally the number of violated SLA is observed during the invocations.

This study used the throughput and respond time dataset collected from Zheng [34, 35] which its validity is confirmed in his study [17, 35-37]. This dataset includes the throughput and respond time of 5825 services which they are invoked by 339 users. Therefore, 339 throughputs and respond times are collected for each service. On the other hand, the SH-SLA experimental scenario needs the throughput and respond times of only 4 services as illustrated in Figure 6. Finally 339*8 matrix data is captured from original dataset.

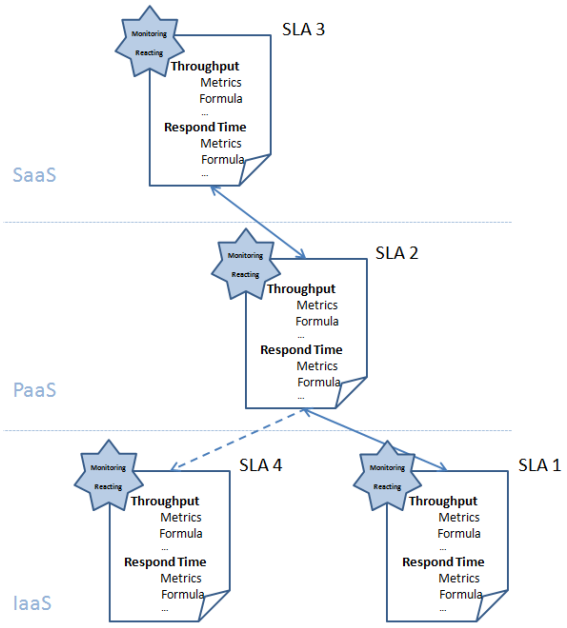

Figure 6. Simulated SH-SLA scenario

The negotiation system is located in service provider as presented in Figure 1 however this process is not a part of this research scope. A predefined agreed value is assumed between service provider and service consumer as a SLO. These agreed values are estimated based on normal attribute value achieved from the dataset. Both SH-SLA and common SLA are configured by the same metrics, formula, SLO and threshold value.

In both SH-SLA and common SLA simulation, firstly the SLA1 should assess the current attribute values as a lowest layer SLA then notify the monitoring results to the upper layer SLA. Afterwards, SLA2 should monitor the current attribute based on SLA1 notification and respond to the SaaS layer SLA. Finally SLA3 assesses the final attribute values based on lower layer attributes value and notify to the end user. This procedure is frequently repeated per each invocation. Moreover, the SH-SLA has the ability of reaction against notified violations while the common SLA only checks the attributes value. In simulated SH-SLA scenario, PaaS user has migrated to the SLA4 when any violations or critical value reported from SLA1.

The SH-SLA and the common SLA violations are observed in this experiment with 339*8 data. Furthermore, this research tried to observe how the increasing of SLA1 violation

affects the end user in both SH-SLA and common SLA. For this purpose, the first experiment is repeated when the value of SLA1 attributes is changing from 100% to -100%. This experiment is started with 100% increased attribute data in SLA1 to observe its effects on SLA3 in both SH-SLA and common SLA. The test is repeated frequently when the SLA1 attributes are decreasing by 10% in each time. The last test is done by 100% deducted SLA1 attributes.

5. EVALUATION AND COMPARISION

5.1. SH-SLA Monitoring Results

After SH-SLA implementation, the experiment has been done and its results confirmed the validity of this model. During the 339 service invocations all attributes of different SLAs are monitored. The Figure 7 has presented the SH-SLA output in the monitoring log database. The output of SH-SLA monitoring for SLA3 is also shown in Figure. SLA3 is an agreement between the end user and SaaS provider. The respond time and throughput are the attributes of SLA3 which they have their specific agreed value. These attributes are assessed based on their dataset value and the notified attributes from the lower layer provider. The respond time of service is presented in Figure 6a for each invocation when the agreed value is less than 1 millisecond and the threshold value is between 0.9 and 1 millisecond. The SH-SLA detected 49 respond delay which they have taken more than 1 millisecond as illustrated in Figure 7a. Moreover, the respond time was in critical range in 18 invocations. On the other hand, Throughput monitoring has depicted in Figure 7b which the throughput should be more than 20 and threshold area is between 25 and 20. During the service invocations, 39 violated throughputs are detected and 26 throughput values are observed in critical area. However, SLA3 is violated if either respond time or throughput is exceeded the agreed value. Totally, the SH-SLA monitoring system detected 83 number of violated SLAs during the 339 invocations which should react against them.

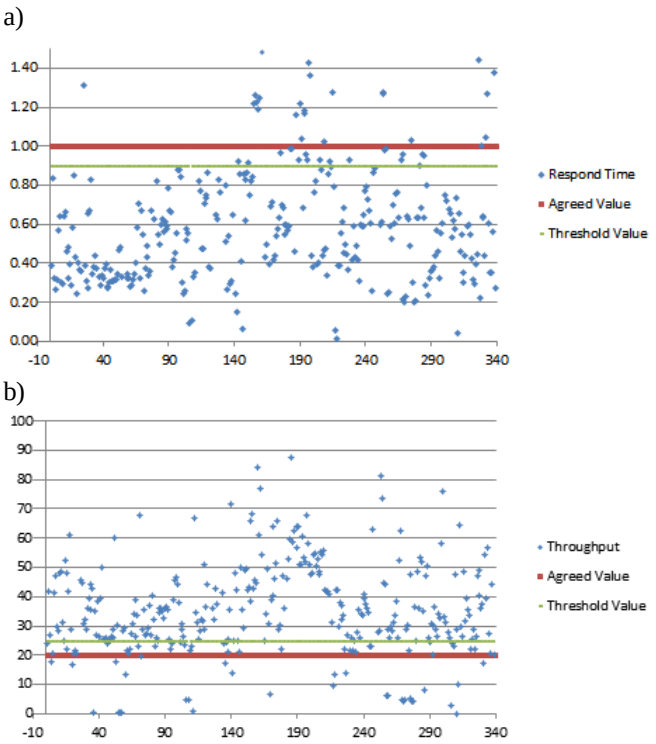

Figure 7. SLA3 monitoring by SH-SLA for a) respond time and b) throughput attributes

5.2. Comparison between SH-SLA and common SLA

The SH-SLA and common SLA are compared regarding two aspects: the number of violated SLAs and the sensation of violation by end user in critical situation.

5.2.1. Violated SLAs

The notified violations in both SH-SLA and common SLA are shown in Figure 8. The number of violations is same in SH-SLA and common SLA for SLA4 and SLA1 because they do not have the ability of reaction and migration to another service provider as shown in Figure 5. On the other hand, the SLA2 has the ability of migration from SLA1 to SLA4 in critical situations therefore SH-SLA can prevent some of the violations before affecting the upper layer. The beneficiations of this reaction are continuing into upper layer and end users as illustrated in SLA3 column. Finally, SH-SLA deducted the violations by 13.68% in SLA2 and the end user sensed 45.75% lesser violations in SLA3 than common

SLA. Therefore the reacting ability of SH-SLA deducted the number of violated SLA2 and SLA3 in comparison with the common SLA.

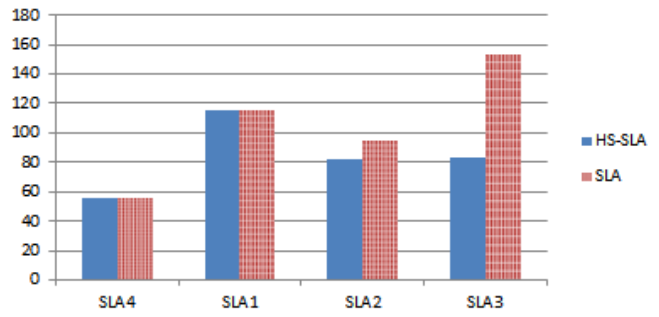

Figure 8. Violated SLAs in SH-SLA and common SLA

5.2.2. End user sensation in tension values

Figure 9 has illustrated the number of violations in both SH-SLA and common SLA at the end user (SLA3) when the lowest layer faults are increased. For this purpose, the tension values are simulated from the real SLA1 attribute value. The experiment is started with 100% increased real SLA1 attributes value. Then, the increased attribute values are deducted step by step by 10% deduction. Finally SLA1 attribute values are arrived to -100% of real value. Figure 8 has illustrated the SH-SLA and common SLA reactions during this changing.

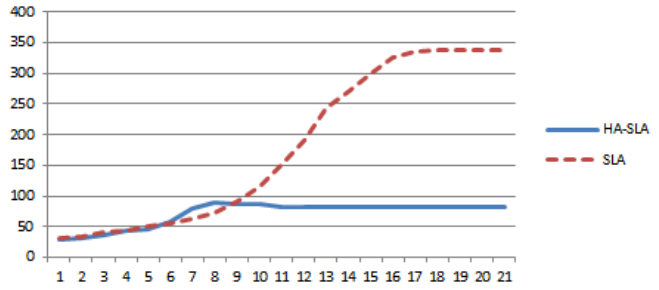

Figure 9. End user sensation in SH-SLA and common SLA when the faults are increasing

The SH-SLA and common SLA have a same result when the SLA1 attribute values are in excellent level (from 1st to 6th). During this period, the SH-SLA does not need any reactions because all SLA1 attribute values are following the agreed value. Surprisingly, the SH-SLA violations are more than common SLA in 7th and 8th periods. In this period, the SH-SLA detected some critical values and migrated to SLA4. In contrast, the critical values of SLA1 were not violated while the SLA4 was in violated attribute. Therefore, the violated SLA in SH-SLA became more than common SLA in this unsuccessful migration period. Finally the violated SLA is dramatically increased in common SLA when the SLA1 attribute values were hardly deducting. During this period, SH-SLA had a long time successful migration to SLA4 so the end users have not sensed the released faults from SLA1. At the final steps, the SH-SLA was stable in 82 violated SLA while the violations of common SLA are increased to 338.

Although in short particular period of time the SH-SLA violations were more than common SLA, this was a rare special situation which could be moderated by effective migration decisions. Moreover, only one migration alternative just for SLA2 is considered in this experiment scenario as an alternative reaction. To have a more reliable cloud services, more reaction strategies can be added to this scenario.

6. CONCLUSION

The most of related works applied the SLA monitoring system from SOA and grid computing to cloud computing while they have the different requirements. This paper proposed SH-SLA model for SLA monitoring based on the hierarchical nature of cloud computing. It also could react against any critical value and SLA violations to prevent them. Proposed model changed the plain SLA to the hierarchical self-healing SLA. Each SLA is able to monitor its QoS and react against violation. So, service providers have a great chance to prevent the SLA violations before sensing by end users. The SH-SLA is simulated to be run in IaaS, PaaS and SaaS layers. The proposed model is validated by within-subject design of experimental methodology. The experiment results indicated that the SH-SLA deducted the violated SLA by 45.75% at the end user in comparison with common SLA. Moreover, SH-SLA was more reliable when the IaaS faults were increasing.

Although the SH-SLA is successfully validated, the reacting decisions should be improved and other preventive strategy should be considered in future work.

AKNOWLEDGEMET

This project is funded by the Exploratory Research Grant Scheme, Ministry of Higher Education of Malaysia. Project no.: ERGS/1/2012/TK06/UPM/02/46.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "A view of cloud computing," *Communications of the ACM*, vol. 53, pp. 50-58, 2010.

[2] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, 2008, pp. 5-13.

[3] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, 2010, pp. 27-33.

[4] W.-T. Tsai, X. Sun, and J. Balasooriya, "Service-oriented cloud computing architecture," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, 2010, pp. 684-689.

[5] S. G. Gómez, J. L. Rueda, and A. E. Chimeno, "Management of the Business SLAs for Services eContracting," in *Service Level Agreements for Cloud Computing*, ed: Springer, 2011, pp. 209-224.

[6] B. P. Rimal, A. Jukan, D. Katsaros, and Y. Goeleven, "Architectural requirements for cloud computing systems: an enterprise cloud approach," *Journal of Grid Computing*, vol. 9, pp. 3-26, 2011.

[7] M. J. Buco, R. N. Chang, L. Z. Luan, C. Ward, J. L. Wolf, and P. S. Yu, "Utility computing SLA management based upon business objectives," *IBM Systems Journal*, vol. 43, pp. 159-178, 2004.

[8] M. Comuzzi, J. Vonk, and P. Grefen, "Measures and mechanisms for process monitoring in evolving business networks," *Data & Knowledge Engineering*, vol. 71, pp. 1-28, 2012.

[9] I. Ul Haq and E. Schikuta, "Aggregation patterns of service level agreements," in *Proceedings of the 8th International Conference on Frontiers of Information Technology*, 2010, p. 40.

[10] O. Mola and M. A. Bauer, "Towards Cloud Management by Autonomic Manager Collaboration," *International Journal of Communications, Network and System Sciences*, vol. 4, pp. 790-802, 2011.

[11] B. S. ARAB and A. A. A. GHANI, "EMPLOYING PERFORMANCE COUNTERS AND SOFTWARE WRAPPER FOR MEASURING QOS ATTRIBUTESOF WEB SERVICES," *Journal of Theoretical and Applied Information Technology*, vol. 34, 2011.

[12] E. Badidi, "A framework for brokered Service Level agreements in SOA environments," in *Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on*, 2011, pp. 37-42.

[13] R. Jurca, B. Faltings, and W. Binder, "Reliable QoS monitoring based on client feedback," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 1003-1012.

[14] I. Ul Haq, I. Brandic, and E. Schikuta, "Sla validation in layered cloud infrastructures," *Economics of Grids, Clouds, Systems, and Services*, pp. 153-164, 2010.

[15] I. Brandic, V. C. Emeakaroha, M. Maurer, S. Dustdar, S. Acs, A. Kertesz, and G. Kecskemeti, "LAYSI: A Layered Approach for SLA-Violation Propagation in Self-manageable Cloud Infrastructures," in *Computer Software and Applications Conference Workshops*, 2010, pp. 365-370.

[16] D. Weyns, B. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, and K. M. Göschka, "On patterns for decentralized control in self-adaptive systems," in *Software Engineering for Self-Adaptive Systems II*, ed: Springer, 2013, pp. 76-107.

[17] Y. Dai, Y. Xiang, and G. Zhang, "Self-healing and Hybrid Diagnosis in Cloud Computing," in *Cloud Computing*, ed: Springer, 2009, pp. 45-56.

[18] G. Cicotti, S. D'Antonio, R. Cristaldi, and A. Sergio, "How to Monitor QoS in Cloud Infrastructures: The QoSMONaaS Approach," in *Intelligent Distributed Computing VI*, ed: Springer, 2013, pp. 253-262.

[19] C. Muller, M. Oriol, M. Rodriguez, X. Franch, J. Marco, M. Resinas, and A. Ruiz-Cortes, "SALMonADA: A platform for monitoring and explaining violations of WS-agreement-compliant documents," in *Principles of Engineering Service Oriented Systems (PESOS), 2012 ICSE Workshop on*, 2012, pp. 43-49.

[20] A. Kertesz, G. Kecskemeti, and I. Brandic, "Autonomic sla-aware service virtualization for distributed systems," in *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, 2011, pp. 503-510.

[21] P. Varalakshmi, K. Priya, J. Pradeepa, and V. Perumal, "SLA with Dual Party Beneficiality in Distributed Cloud," in *Advances in Computing and Communications*, ed: Springer, 2011, pp. 471-479.

[22] H. Liu, D. Xu, and H. K. Miao, "Ant Colony Optimization Based Service Flow Scheduling with Various QoS Requirements in Cloud Computing," in *Software and Network Engineering (SSNE), 2011 First ACIS International Symposium on*, 2011, pp. 53-58.

[23] T. A. L. Genez, L. F. Bittencourt, and E. R. M. Madeira, "Workflow scheduling for SaaS/PaaS cloud providers considering two SLA levels," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, 2012, pp. 906-912.

[24] X. Wang, Z. Du, and Y. Chen, "An adaptive model-free resource and power management approach for multi-tier cloud environments," *Journal of Systems and Software*, 2012.

[25] C. Muller, M. Oriol, M. Rodríguez, X. Franch, J. Marco, M. Resinas, and A. Ruiz-Cortes, "SALMonADA: A platform for monitoring and explaining violations of WS-agreement-compliant documents," in *Principles of Engineering Service Oriented Systems (PESOS), 2012 ICSE Workshop on*, 2012, pp. 43-49.

[26] A. L. Freitas, N. Parlavantzas, and J. L. Pazat, "A QoS assurance framework for distributed infrastructures," in *Proceedings of the 3rd International Workshop on Monitoring, Adaptation and Beyond*, 2010, pp. 1-8.

[27] V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level metrics to high level SLAs-LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," in *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, 2010, pp. 48-54.

[28] M. Maurer, I. Breskovic, V. C. Emeakaroha, and I. Brandic, "Revealing the MAPE loop for the autonomic management of cloud infrastructures," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, 2011, pp. 147-152.

[29] B. Abrahao, V. Almeida, J. Almeida, A. Zhang, D. Beyer, and F. Safai, "Self-adaptive SLA-driven capacity management for internet services," in *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, 2006, pp. 557-568.

[30] A. Keller and H. Ludwig, "The WSLA framework: Specifying and monitoring service level agreements for web services," *Journal of Network and Systems Management*, vol. 11, pp. 57-81, 2003.

[31] M. N. Bennani and D. A. Menasce, "Resource allocation for autonomic data centers using analytic performance models," in *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, 2005, pp. 229-240.

[32] V. Stantchev and C. Schröpfer, "Negotiating and enforcing qos and slas in grid and cloud computing," *Advances in Grid and Pervasive Computing*, pp. 25-35, 2009.

[33] V. C. Emeakaroha, M. A. S. Netto, R. N. Calheiros, I. Brandic, R. Buyya, and C. A. F. De Rose, "Towards autonomic detection of sla violations in cloud infrastructures," *Future Generation Computer Systems*, 2011.

[34] Y. Dai, Y. Xiang, and G. Zhang, "Self-healing and Hybrid Diagnosis in Cloud Computing," *Cloud Computing*, pp. 45-56, 2009.

[35] Z. Zheng. *Distributed Reliability Assessment Mechanism for Web Services*. Available: http://www.wsdream.net/

[36] Y. Zhang, Z. Zheng, and M. R. Lyu, "Exploring latent features for memory-based QoS prediction in cloud computing," in *Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on*, 2011, pp. 1-10.

[37] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed qos evaluation for real-world web services," in *Web Services (ICWS), 2010 IEEE International Conference on*, 2010, pp. 83-90.