# HS-SLA: A Hierarchical Self-Healing SLA Model for Cloud Computing

Ahmad Mosallanejad, Rodziah Atan, Rusli Abdullah, Masrha Azmi Murad, Taghi Javdani
*Faculty of Computer Science and Information Technology, UPM, Malaysia*
*ahmad.upm@gmail.com, {rodziah,rusli, masrah}@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 enforced as a plain documental agreement without any relation to other upper and lower level agreements among different layers of cloud computing. Hence, the cloud computing environment needs the hierarchical and autonomic SLA. This paper proposes the HS-SLA model to enforce the hierarchical self-healing SLA in cloud computing. The self-healing ability includes the SLA monitoring, violation detecting and violation reacting processes. In HS-SLA, the dependent SLAs have a connection to each other hierarchically. This model enables the SLA to check its QoS attributes and notify the current status to dependent SLAs. Additionally, HS-SLA could prevent or propagate the notified violations by an urgent reaction. So, the service providers have a great chance to prevent the violated SLA before sensing by end users. The HS-SLA model is simulated and the experiment results have shown the violation detection and reaction capabilities of the proposed model in a hierarchical cloud environment. Furthermore, in HS-SLA the end users meet the lesser violated SLAs in comparison with the common SLA structure.

## KEYWORDS

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

## 1. INTRODUCTION

Nowadays, cloud computing is an excited paradigm in both research and business area [1, 2]. Many service providers and service consumers have interactions during 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 necessary contract between the service provider and user to define the functionalities and quality of agreed services [5]. Therefore, SLA is a fundamental document which all service invocations are conducted based on its features including service level objective (SLO), their metrics and attributes. This contract is the basic foundation of all relations among service provider and consumer in various layers of cloud computing [6, 7]. Both service provider and consumer should validate their contracts during the service invocations by SLA monitoring [8].

Currently, the most of SLAs are a single XML document which contains functionalities and quality of services (QoS) between specific service provider and consumer. They stand independently with no relation to 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 effective relations between dependent SLAs is a vital challenge which makes the SLA management system inefficient. Having an effective SLA monitoring and a violation reacting system, the hierarchical SLA model is needed based on cloud computing nature [9, 10].

Three types of SLA monitoring systems are available including consumer side, provider side and trusted party side SLA monitoring system [11-13] . Each of these approaches has its own advantages and disadvantages but all of them are based on the centralized monitoring system. In centralized monitoring system, the specific SLA management center in consumer, provider or trusted party side is responsible for SLA monitoring and violation detecting. The most of SLA monitoring systems are applied in service oriented architecture (SOA) and grid computing environments without enough attention given to the cloud computing environment and requirements [14, 15]. Furthermore, the most of SLA management and monitoring systems only

release the SLA violation reports without an effective reaction. 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 paper, hierarchical self-healing (HS)-SLA model is proposed to conduct the SLA monitoring and reacting based on the hierarchical nature of cloud computing. The hierarchical self-healing SLA is designed in the HS-SLA model which each SLA has connected to their related SLAs in different layers of cloud computing. In the proposed model, each SLA is notifying its status to other related SLAs. Moreover, each SLA has the ability of monitoring and reacting independently. So, the SLA can assess the monitoring results itself and notify to the dependent SLAs. This model eliminates the mentioned shortcoming of current SLAs in cloud computing.

The rest of the paper is organized as follows: The related works are stated in Section 2. Afterward, the 3rd Section describes the HS-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 really considering the cloud structure. The main parts of a self - healing system include monitoring and reacting process. So, the related works are divided into three parts: SLA models, SLA monitoring and SLA reacting frameworks.

### 2.1. SLA Model Literature

SLA attributes and its criteria are investigated for cloud computing by Alhamad *et al.* [18]. This study determined the individual SLA metrics for SaaS, PaaS and IaaS separately. Although specific metrics are presented based on their layer requirements, they never considered the SLAs relation and hierarchical nature of cloud computing. Patel *et al.* [19] proposed the WSLA framework for SLA enforcement in cloud computing while this structure already proposed for SOA by Keller and Ludwing [20]. This framework is not enough adapted for cloud needs.

Cloud computing nature and requirements are totally different from SOA therefore SLA management systems cannot directly be brought from SOA. Some other frameworks such as RESTful [21], SALmonADA [22], SLA@SOI [23], [24], [25] and [26] have put aside the hierarchical nature of cloud computing. They have directly applied the SLA structure in the cloud computing area from SOA and grid computing. The SSV architecture proposed by Kertesz *et al.* [27] which contracted the SLA between an individual service provider and consumer after the negotiation process. However, they did not consider the hierarchical relation of SLAs in the cloud environment. Although a research by Undheim *et al.* [28] attempted to deploy the on demand SLAs with QoS details on different levels, it followed the common SLA structure with stated shortcomings. Different techniques of SLA negotiation including direct, broker and market based negotiations are investigated and compared by Grandits *et al.* [29] while they did not have enough attention to the related SLAs in negotiation process. So, the output of these techniques is the isolated agreed SLA. Some other studies such as [30], [21], [31] and [32] also have not considered the upper and lower layer SLAs in negotiation process.

### 2.2. SLA Monitoring Literature

The LoM2HiS framework introduced by Emeakaroha *et al.* [33] as a part of FOSII [22, 34] infrastructure. They tried to map the low level metrics into high level SLA by their monitoring method but the common SLA is used and they followed the centralized monitoring system [35].

In the most of related works, the centralized monitoring system technique is utilized either in provider, consumer or trusted party sides. Mostly, there is a gap between SLA management system and SLA monitoring system which makes the self-healing SLA system inefficient. The LoM2HiS monitoring framework is located in FoSII infrastructure which calculated the SLA attributes value based on the proposed metrics and formula [35]. In this framework, the SLAs consist of attributes, metrics and formulas have to be located in a central repository. Then, another monitoring system is used to measure the SLA validity. Similarly, the proposed SLA monitoring system by Al-Ghuwairi and Cook [36] is installed in a virtual machine to measure the IaaS quality.

The QoSMONaaS approach [37] focused to the QoS monitoring as a service. Although this flexible approach is a third party SLA monitoring system, it still needs the external service for SLA validating. In another study, the LAYSI framework is proposed by Brandic *et al.* [15]. It is one of the few studies which concentrated on layered monitoring for cloud computing [14]. This framework distributed the monitoring process and SLA management into different actors while they should actually be distributed in the hierarchical SLAs. Another related work is DeSVi architecture by Emeakaroha *et al.* [38], it is designed based on LoM2HiS framework which limited for only one data center as an IaaS. Finally, a self-adaptive hierarchical monitoring mechanism is proposed by Katsaros *et al.* [39] which distributed different components between IaaS, PaaS and SaaS layers. However, it is not based on service invocation streams yet and followed the common SLA structure.

### 2.3. SLA Reacting Literature

The violation reaction process is rarely discussed in the mentioned SLA monitoring frameworks. The most of monitoring systems just tried to report the detected violations [22, 23] or assess the penalty cost of service providers [26, 32]. However, an effective reaction process is needed to react against SLA violations.

Some studies such as [40], [41] and [42] utilized the SLAs to manage the provider resources without any reaction against violations. The FoSII and QU4DS frameworks introduced by Emeakaroha *et al.* [33, 43]. They applied the monitoring, analysis, planning and execution loop (MAPE) model to enforce the self-healing cloud computing. This architecture has beneficial for cloud management but not for sudden violation reaction. In another study, the LoM2HiS framework tried to predict the SLA violations by predefining the thresholds. Then, the alerts are notified into the internal knowledge component for possible adjustment in provider resource [35] however there were not any other reaction strategies. On the other hand, LAYSI has propagated the detected SLA violations into upper layer but its communication model was based on subscription between meta-negotiators, brokers and automatic service deployed [15]. This reaction model was complex and followed the plain SLA document structure which is in contrast of self-healing SLA.

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. Having the self-healing SLA in cloud computing, an effective self-monitoring SLA and reaction mechanism are needed.

## 3. HS-SLA MODEL

Having an effective self-healing SLA in cloud computing, HS-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 HS-SLA is the main output of negotiation management system then HS-SLA should be run. Next, the data collector passes the relevant metrics data from resources and lower layer notifications to the HS-SLA. Finally the running HS-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.

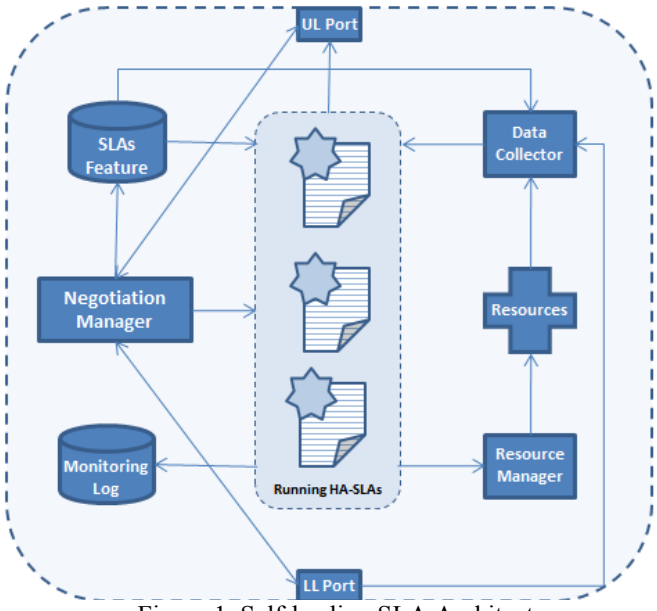



Figure 1. Self-healing SLA Architecture

### 3.2. Hierarchical SLA

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 HS-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 HS-SLA. Figure 2 has depicted the common SLA and HS-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 HS-SLA is a relational contract.

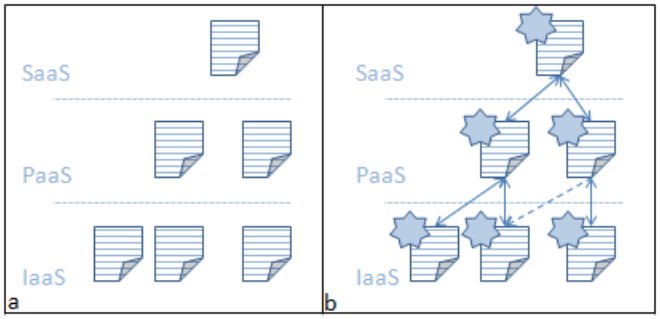



Figure 2. a) Current SLA in cloud computing b) HS-SLA overview

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 HS-SLA could provide a hierarchical self-healing SLA independently. Figure 2b has displayed the HS-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 Process

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 HS-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.

HS-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 HS-SLA structure has the ability of other operations which they will be investigated in future works.

### 3.4. SLA Reacting Process

The related works rarely focus on the SLA based reacting framework while this is a vital issue to sustain the cloud computing reliable. In the HS-SLA model, reacting function also embedded into each SLA. This function reacts against SLA violation and critical value which notified from lower layer. The reaction could be either violation prevention or violation propagation. When any violated SLA notified from lower layers, first of all, current SLA should try to prevent the violation by urgent migrating to another resource or provider. If the fault could not be prevented, current SLA has to propagate the SLA violation alert to the upper layer SLAs. Similarly, the upper layer also tries to prevent or propagate this notified violation. Therefore, hierarchical service providers have a big chance to prevent the violations before affecting the end users. Figure 2b has shown the SLA notifications, violation propagations (arrow) and urgent migration (dashed line). Moreover, threshold value also defined per each attribute for early violation reaction. These threshold values should be proven by data mining and artificial intelligence techniques based on previous reaction results which are not as a part of this research. When the attribute value exceeds the threshold value and current provider could not adjust it, it should be propagated into the upper layer as a critical value notification.

### 3.5. HS-SLA Structure

The current SLA is a plain agreement document but proposed HS-SLA model is a self-healing SLA including monitoring and reacting functionalities. The HS-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 3, line 11. Actually, the array of these attributes is a part of the HS-SLA (Figure3, line 20). Lower layer, upper layer and reserved providers are defined (from 21$^{st}$ to 23$^{rd}$ line) to have an effective hierarchical relation to other SLAs. Figure 3 (from 25$^{th}$ to 33$^{rd}$ line) has shown the functional parts of the HS-SLA consist of monitoring and reacting tasks. Practically, each HS-SLA can monitor itself and react against any

violations. This is a significant contribution of the HS-SLA model as a real self-healing SLA.

```
struct Attribute
{
   int code;
   string name;
   Metric[] metrics;
   string formula;
   float agreedV;
   char agreedOperation;
   float criticalV;

   float assessor()
   {
     …
   }
}
public struct HA_SLA
{
   int code;
   string name;
   Attribute[] attributes;
   int[] lowerSLAs;
   int[] emergency;
   int[] upperSLAs;

   Notification[] Monitoring()
   {
     …
   }
   Notification[] Reacting()
   {
     …
   }
}
```

Figure 3. HS-SLA structure

Normally, the SLA monitoring function checks the value of attributes when the user invokes the service. So the HS-SLA monitoring procedure is running per each service invocation. The monitoring function algorithm is presented in Figure 4.

```
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 4. 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 HS-SLA model, described in Section 3, is simulated to evaluate the proposed model validity. Figure 5 has presented a simulated scenario based on HS-SLA usage in cloud computing. The SLAs from 1 to 4 are the agreements in different layers of cloud computing. Each SLA is a contract about a particular service between specific user and provider. 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 the PaaS provider and SaaS vendor as a client so its service should respond the SaaS requests based on SLA2 attributes. 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 [44], [20], [45] and [46]. As described, the simulated SLAs is depending on each other hierarchically. This simulation is developed by C# language and MySQL database.

This study tries to evaluate the simulated HS-SLA by within-subjects design of experimental methodology used by Emeakaroha *et al.* [38]. In this method, the effects of both HS-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, HS-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 [47, 48] which its validity is confirmed in his study [48, 49]. 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 HS-SLA experimental scenario needs the throughput and respond times of only 4 services as illustrated in Figure 5. Finally 339*8 matrix data is captured from original dataset as presented in appendix.

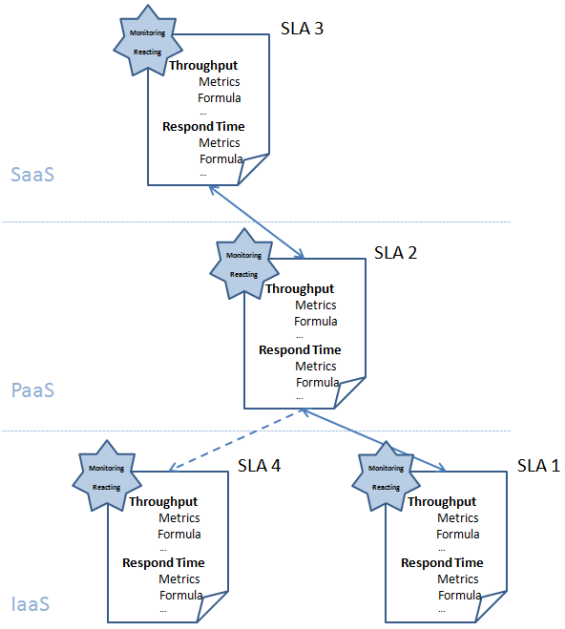



Figure 5. Simulated HS-SLA scenario

The negotiation system is located in service provider as presented in Figure 1. The negotiation process is not a part of this research scope so the default 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 HS-SLA and common SLA are configured by the same metrics, formula, SLO and threshold value.

In both HS-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. After that, SLA2 should monitor the current attribute based on SLA1 notification and respond to the SaaS layer SLA. Finally the 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 HS-SLA has the ability of reaction against notified violations while the common SLA only checks the attributes value. In simulated HS-SLA scenario, PaaS user has migrated to the SLA4 when any violations or critical value reported from SLA1.

The HS-SLA and the common SLA violations are observed in this experiment with 339*8 data. Moreover, this research tried to observe how the increasing of SLA1 violation affects the end user in both HS-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 HS-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.

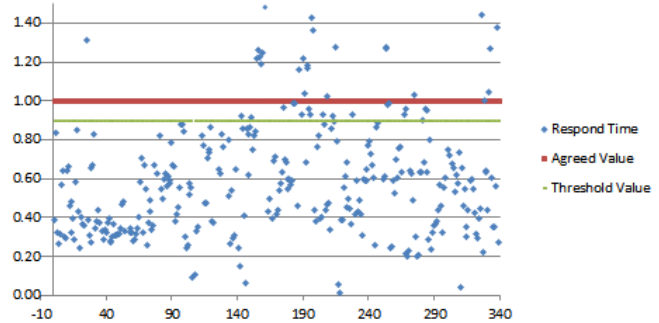
## 5. EVALUATION AND COMPARISION

### 5.1. HS-SLA Monitoring Results

After HS-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 data of Figure 6 are extracted from the recorded HS-SLA output in the monitoring log database. Figure 6 has shown the ability of attributes monitoring in HS-SLA for SLA3. SLA3 is an agreement between the end user and SaaS provider. Both respond time and throughput are the attributes of SLA3 which they have their own agreed value. These attributes are assessed based on their dataset value and the notified attributes from the lower layer provider. The service respond times are presented in Figure 6a for each invocation when the agreed value is less than 1 ms and the threshold value is between

0.9 and 1ms. The HS-SLA detected 49 respond delay which they have taken more than 1ms as illustrated in Figure 6a. Moreover, in 18 invocations the critical value is recorded between 0.9 and 1 ms. On the other hand, Throughput monitoring has depicted in Figure 6b 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 responds time or throughput is exceeded the agreed value. Totally, the HS-SLA monitoring system detected the 83 violated SLA during the 339 invocations which should react against them.

a)




b)

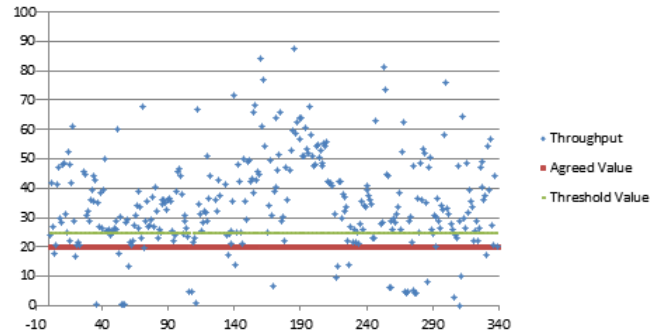



Figure 6. SLA3 monitoring by HS-SLA for a) respond time and b) throughput attributes

## 5.2. Comparison between HS-SLA and common SLA

The HA-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 HS-SLA and common SLA are shown in Figure 7. The number of violations is same in HS-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 HS-SLA can prevent some of the violations before affecting the upper layers. The beneficiations of this reaction are continuing into upper layer and end users as illustrated in SLA3 column. Finally, SLA2 of HS-SLA deducted the violations by 13.68% and the end user sensed 45.75% lesser violations in SLA3 than common SLA. Therefore the reacting ability of HS-SLA deducted the number of violated SLA2 and SLA3 in comparison with the common SLA.

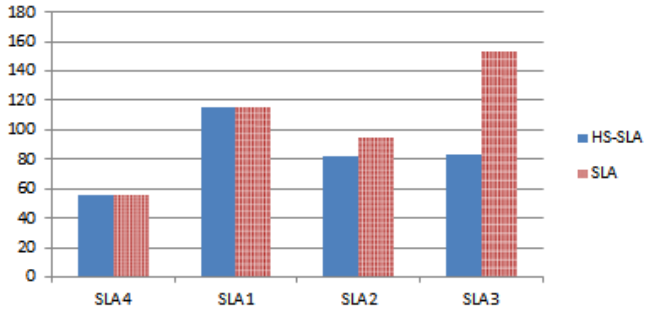



Figure 7. Violated SLAs in HS-SLA and common SLA

### 5.2.2. End user sensation in tension values

Figure 8 has illustrated the number of violations in both HS-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. Actually, Figure 8 has illustrated the HS-SLA and common SLA reactions during these changing.

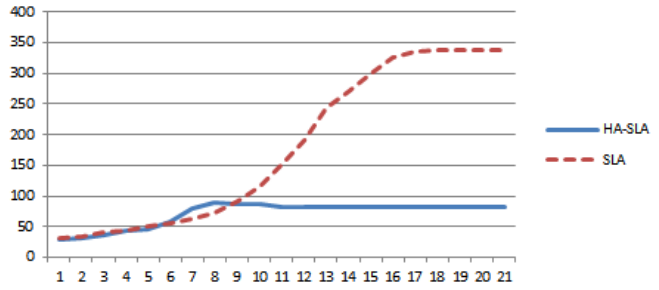



Figure 8. End user sensation in HS-SLA and common SLA when the faults are increasing

The HS-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 HS-SLA does not need any reactions because all SLA1 attribute values are following the agreed value. Surprisingly, the HS-SLA violations are more than common SLA in 7th and 8th periods. In this period, the HS-SLA detects some critical values and migrates to SLA4. In contrast, the SLA1 critical values were not violated but the SLA4 was in violated attribute. Therefore, the violated SLA in HS-SLA becomes more than common SLA in this unsuccessful migration period. Finally the violated SLA is dramatically increased in common SLA when the SLA1 attribute values are hardly deducting. During this period, HS-SLA has a long time successful migration to SLA4 so the end users do not sense the released faults from SLA1. At the final steps, the HS-SLA is stable in 82 violated SLA while the violations of common SLA are increased to 338.

Although in short particular period of time the HS-SLA violations are more than common SLA, this is 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. Having a more reliable cloud computing, the more reaction strategy could be added to this scenario.

## 6. CONCLUSION

According to the literature review, there is a lack of appropriate self-healing SLA model in cloud computing. The most of related works applied the SLA management system from SOA and grid computing to cloud computing while they have the different requirements. This paper proposed a HS-SLA model for SLA monitoring based on the hierarchical nature of cloud computing. It also could react against any critical value or SLA violations to prevent them. Proposed model changed the isolated document SLA to the hierarchical self-healing SLA. The HS-SLA is simulated as the SLAs of IaaS, PaaS and SaaS layers. Each SLA has the ability of its QoS monitoring. Moreover they could react against any SLA violations to prevent or propagate them. Therefore, in the HS-SLA model, service providers have a great chance to prevent the violated SLA before sensing by end users. The proposed model is validated by within-subject design of experimental methodology. Based on experiment results, the HS-SLA monitored the attributes value to detect the violated SLAs and react against them. Finally, The HS-SLA deducted the violated SLA by 45.75% at the end user in comparison with common SLA. In addition, HS-SLA was dramatically reliable by preventing the violation when the IaaS faults were increasing.

Although the HS-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] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA framework for cloud computing," in *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*, 2010, pp. 606-610.

[19] P. Patel, A. Ranabahu, and A. Sheth, "Service Level Agreement in cloud computing," in *Cloud Workshops at OOPSLA*, 2009.

[20] 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.

[21] R. Kübert, G. Katsaros, and T. Wang, "A RESTful implementation of the WS-Agreement specification," in *Proceedings of the Second International Workshop on RESTful Design*, 2011, pp. 67-72.

[22] 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.

[23] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, and R. Yahyapour, "Establishing and monitoring SLAs in complex service based systems," in *Web Services, 2009. ICWS 2009. IEEE International Conference on*, 2009, pp. 783-790.

[24] P. Khanna and B. Babu, "Cloud Computing Brokering Service: A Trust Framework," in *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, 2012, pp. 206-212.

[25] H. P. Borges, J. N. de Souza, B. Schulze, and A. R. Mury, "Automatic generation of platforms in cloud computing," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, 2012, pp. 1311-1318.

[26] A. V. Dastjerdi, S. G. H. Tabatabaei, and R. Buyya, "A dependency-aware ontology-based approach for deploying service level agreement monitoring services in Cloud," *Software: Practice and Experience,* vol. 42, pp. 501-518, 2011.

[27] 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.

[28] A. Undheim, A. Chilwan, and P. Heegaard, "Differentiated Availability in Cloud Computing SLAs," in *Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on*, 2011, pp. 129-136.

[29] P. Grandits, G. Margreiter, A. Juch, and M. Ertl, "Dynamic Negotiation of SLAs in Context of the Cloud Monitoring Frameworks."

[30] S. I. Hashmi, R. Haque, E. Schmieders, and I. Richardson, "Negotiation towards service level agreements: A life cycle based approach," in *Services*

*(SERVICES), 2011 IEEE World Congress on*, 2011, pp. 1-8.

[31] M. C. Wang, X. Wu, W. Zhang, F. Q. Ding, J. Zhou, and G. C. Pei, "A Conceptual Platform of SLA in Cloud Computing," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, 2011, pp. 1131-1135.

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

[33] 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.

[34] M. Dumas, L. García-Bañuelos, A. Polyvyanyy, Y. Yang, and L. Zhang, "Aggregate quality of service computation for composite services," *Service-Oriented Computing,* pp. 213-227, 2010.

[35] V. C. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Cloud resource provisioning and SLA enforcement via LoM2HiS framework," *Concurrency and Computation: Practice and Experience,* 2012.

[36] A. R. Al-Ghuwairi and J. Cook, "Modeling and Enforcement of Cloud Computing Service Level Agreements," *Technical Report,* 2012.

[37] G. Cicotti, S. D'Antonio, R. Cristaldi, and A. Sergio, "How to Monitor QoS in Cloud Infrastructures: The QoSMONaaS Approach," *Intelligent Distributed Computing VI,* vol. 446, pp. 253-262, 2013.

[38] 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.

[39] G. Katsaros, G. Kousiouris, S. V. Gogouvitis, D. Kyriazis, A. Menychtas, and T. Varvarigou, "A Self-adaptive hierarchical monitoring mechanism for Clouds," *Journal of Systems and Software,* 2011.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

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

[49] 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.