A SLA-based Cloud Computing Framework: Workload and Location Aware Resource Allocation to Distributed Data Centers in a Cloud

Seokho Son, Gihun Jung, and Sung Chan Jun
School of Information and Communications, Gwangju Institute of Science and Technology, Gwangju, Republic of Korea

Abstract - As the number of users of cloud services increase all over the world, cloud service providers keep deploying geographically distributed data centers more. Since resource capacity of a data center is limited, cloud service providers need to distribute load to data centers. However, when the provider distributes load, SLAs (service level agreements) established with consumers should be guaranteed in an environment where a cloud provider operates geographically distributed data centers. Therefore, this paper proposes a SLA-based cloud computing framework to facilitate temporal and locational load-aware resource allocation. The contributions of this paper include 1) design of a cloud computing framework including an automated SLA negotiation mechanism and a workload and location aware resource allocation (WLARA) and 2) implementation of an agent-based cloud test bed of the proposed framework. Empirical results using the test bed show the proposed WLARA performs better than other related approaches in terms of guaranteed SLAs.

Keywords: Cloud computing, Distributed data centers, Resource allocation, VM placement, SLA negotiation, SLA management

1 Introduction

Cloud computing is a computing paradigm that provides computing resources as services through a network to Cloud users. A cloud consists of a type of parallel or distributed systems that consists of combinations of interconnected computing resources and virtualized computing resources [1]. There are many existing cloud computing environments such as Amazon EC2 and Amazon S3 [2]. Considering such interests and business willingness of IT leaders, cloud computing paradigm is already important and to be essential in various business. In such cloud computing environments, a cloud service user consumes cloud resources as a cloud service and pay for the usage of the service. Before a cloud provider provisions a service to a consumer, the cloud provider and consumer need to establish a SLA in advance. The SLA is an agreement including the level of QoS (quality of service) between a service provider and a consumer. Usually, an SLA includes the price of a service, and the level of QoS can be adjusted by the price of the service. For instance, a cloud provider can charge a higher price to a consumer who requires a high level of QoS.

According to the increased number of cloud service users all over the world, cloud service providers keep implementing geographically distributed data centers. Since the resource capacity is limited in a data center, a cloud providers need to distribute resource load to the data centers for system performance and stability. In addition, resource load can be distributed in temporal manner since resource load in a cloud generally fluctuates by time [3]. Because of the limited resource capacity, it is hard for cloud providers to handle resource demand that exceeds the resource capacity. Therefore, to cloud providers, a load balancing scheme is a very important issue to design a cloud computing framework, and it is directly related with cloud providers’ profit. Whereas it is important to design a cloud computing framework that facilitates efficiency and stability of a system, SLA-based cloud computing framework considering both temporal and locational resource load have not been sufficiently studied so far. Also we need to consider a load distribution for a cloud provider who operates geographically distributed data centers.

In [4], B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster provided a comparison between OpenNebula and several well-known virtual infrastructure managers such as Amazon EC2, vSphere, Nimbus, and so on. The comparison includes several resource placement policies (i.e., resource allocation policies) of virtual infrastructure managers. For example, there are static-greedy resource selection, round robin resource selection, and placement considering average CPU load. However, whereas the resource placement schemes are focused on placing resources in a data center, they are not focused on resource placement in which a cloud provider operates geographically distributed data centers. With geographically distributed data centers, we need to consider response time violation because of geographical distance. In addition, [5] investigated energy-aware resource provisioning and allocation algorithms that provision data center resources to client applications in a way that improves the energy efficiency of the data center. However, whereas [5] guides research directions in resource allocations, [5] does not consider a cloud provider that operates multiple
geographically distributed data centers to balance resource load and response time by geographical distance, and [5] does not include a specific negotiation mechanism.

[6] considers load placement policies on cooling and maximum data center temperatures in cloud providers that operate multiple geographically distributed data centers. Whereas [6] proposes dynamic load distribution policies that consider all electricity-related costs as well as transient cooling effects, [6] does not focus on guarantees of SLA. [6] noted that the policies delay jobs to avoid overheating or overloading data centers and violate SLAs sometimes in their simulation configuration. Moreover current providers such as Amazon EC2 do not employ sophisticated load placement policies, and the consumers themselves manually select a data center to place their virtual machines.

There are several automated negotiation mechanisms for grid or cloud resource negotiation ([7] for a survey). These negotiation mechanisms are designed for price negotiation, but these mechanisms have a lack in considering other SLA issues such as service time slot and response time. Also, they do not consider both temporal and locational load distributions to geographically distributed data centers.

Hence, this paper proposes a cloud computing framework to facilitate temporal and locational load-aware resource allocation, and we implement a test bed of the proposed cloud framework to verify the usefulness of the proposed framework in a case study. The contribution of this paper are 1) design of a cloud computing framework to facilitate temporal and locational load-aware resource allocation, and 2) implementation of a test bed of the proposed cloud computing framework. Using the proposed system, a cloud consumer can establish SLA about service price, time slot, and response time by an automated SLA negotiation scheme. Whereas there should be many SLA issues or options in cloud services in practice, this paper focuses on 1) service price, 2) time slot to specify range of service time, and 3) service response time among several SLA issues in designing the SLA-based cloud framework. In addition to the SLA negotiation mechanism, service providers need a SLA management scheme to guarantee established SLA with a consumer. In the proposed cloud framework considers service response time to consumers as QoS. To guarantee service response time agreed with a consumer, a SLA management scheme in the proposed framework considers geographical distance between a consumer and distributed data centers of a provider when the cloud provider select a data center to allocate resources for the consumer.

In Fig. 1, the proposed framework consists of cloud service broker and cloud provider. Cloud service broker is an interface between a consumer and a cloud provider. The cloud service broker gets information of a consumer’s preference for services and proceeds for cloud service discovery to find a matched service with the consumer’s preference. After the broker finishes service discovery, the broker can be connected to a cloud provider who own the service. The next step is establishing SLA between the provider and the service broker on behalf of the consumer by the SLA negotiation component. If the negotiation is successful regarding service price, time slot, and response time, the broker makes the consumer pay violation in Section 4. Finally, Section 5 concludes this paper with a list of future works.

2 A cloud computing framework to facilitate resource allocations

2.1 SLA-based cloud computing framework

The proposed cloud framework adopts an automated SLA negotiation mechanism to support establishing SLA. Whereas the variety of SLA options is limited for consumers within enforced SLA strategies, the different preferences between a consumer and a provider can be efficiently narrowed with an automated SLA negotiation mechanism. In case of price of a service, a negotiation mechanism between both a consumer and a provider helps to find an equilibrium satisfaction state for price. Whereas there should be many SLA issues or options in cloud services in practice, this paper focuses on 1) service price, 2) time slot to specify range of service time, and 3) service response time among several SLA issues in designing the SLA-based cloud framework. In addition to the SLA negotiation mechanism, service providers need a SLA management scheme to guarantee established SLA with a consumer. In the proposed cloud framework considers service response time to consumers as QoS. To guarantee service response time agreed with a consumer, a SLA management scheme in the proposed framework considers geographical distance between a consumer and distributed data centers of a provider when the cloud provider select a data center to allocate resources for the consumer.
for the service in the agreed price and reserves the service.

A cloud provider consists of four components: 1) reservation controller, 2) SLA negotiation mechanism, 3) SLA management, and 4) distributed data centers. Cloud provider responds to service broker’s service discovery requests. And SLA negotiation component makes a negotiation session between the provider and the broker to establish SLA. If the automated negotiation is successful, which means they came to a mutually acceptable agreement regarding price, time slot, and response time, so that an SLA has been established, the provider receives charge from the consumer through the broker.

Service reservation controller reserves the requested service by input information of the name of consumer, the service type, the start time of service, the end time of service, and the threshold of response time to the service reservation queue which is reservation list. Reservation controller sequentially checks the reservation queue to initiate the start of the services according to the start time of services. It forwards agreed response time for the services to SLA management component.

The SLA management component selects a data center among distributed data centers to allocate requested service. The conditions of selecting a data center are specified in the SLA (response time in this paper). Each data center includes a physical machine manager (PMM), and a PMM manages physical computing nodes in a data center. All PMMs monitor physical computing nodes to evaluate average response time of a data center to the consumer, and SLA management component selects a data center and specific a physical computing node by evaluated average response times. In conclusion, a service provider and a consumer can not only establish SLA about service price, time slot, and response time, but also the service provider can guarantee established SLA with a consumer through the SLA management component.

The SLA management component selects a data center among distributed data centers to allocate requested service. The conditions of selecting a data center are specified in the SLA (response time in this paper). Each data center includes a physical machine manager (PMM), and a PMM manages physical computing nodes in a data center. All PMMs monitor physical computing nodes to evaluate average response time of a data center to the consumer, and SLA management component selects a data center and specific a physical computing node by evaluated average response times. In conclusion, a service provider and a consumer can not only establish SLA about service price, time slot, and response time, but also the service provider can guarantee established SLA with a consumer through the SLA management component.

2.2 Automated SLA negotiation mechanism

In general a negotiation mechanism consists of negotiation protocol, negotiation strategy, and utility functions. Negotiation protocol is a set of rules in a communication (e.g., possible actions, language, and utterance turn) for a negotiation among negotiation parties. As a negotiation protocol, the negotiation mechanism in this work follows Rubinstein’s alternating offers protocol [12] which lets agents make counter-offers to their opponents in alternate rounds. Both agents generate counter-offers and evaluate their opponent's offers until either an agreement is made or one of the agents’ negotiation deadlines is reached. Counter-proposals are generated according to a negotiation strategy that consists of a concession algorithm and a tradeoff algorithm. When an agent generates a counter-proposal, the agent needs to concede a proposal since it is hard to reach an agreement without a concession. A concession algorithm determines the amount of concession for each negotiation round [13]. Also, a tradeoff algorithm is required to generate a proposal in multi-attributes negotiation. In a multi- attributes negotiation, there are multiple issues to negotiate, and a tradeoff algorithm generates a proposal by combining proposals of individual issues; the mechanism adopted a tradeoff algorithm in [14].

The utility function $U(x)$ represents an agent’s level of satisfaction of a negotiation outcome $x$. A negotiator (i.e., a cloud participant) can specifies utility functions. To define a price utility function, the negotiator needs to specify the most preferred price (IP, initial price) and the least preferred price (RP, reserve price). In general, the range of utility function is $[0,1]$, and $U(I P)=u_{m i n}$ represents the least preferred price (RP) and $U(P)=1$ represents the most preferred price (IP). If price is outside of IP and RP, then $U(P)=0$.

To support cloud users negotiation for service price, time slot, and response time in the proposed framework, it is necessary to define utility functions of the negotiable SLA issues. In this paper, utility functions defined in [14] are adopted which are price utility function $U_{p}(P)$ and time slot utility function $U_{TS}(TS)$. Especially, the time slot utility function defined in [4] supports cloud participants to represent temporal preferences in cloud services. In addition to price and time slot issues considered in [14], this paper considers response time as a SLA negotiation issue. The service response time represents the minimum response time that a provider provides. Let $IRT$ (Initial response time) and $RRT$ (Reserve response time) be the most preferred response time and the lease preferred response time respectively. A RT (Response time) given to a consumer can be evaluated by the response time utility function of a consumer $U_{RT}^{C}(RT)$ as follows,

\[
U_{RT}^{C}(RT) = \begin{cases} 
    u_{min} + (1-u_{min}) \frac{RRT_{C} - RT}{RRT_{C} - IRT_{C}}, & \text{if } IRT_{C} \leq RT \leq RRT_{C} \\
    0, & \text{otherwise}.
\end{cases}
\]

Contrarily, a RT given to a provider can be evaluated by the response time utility function of a provider $U_{RT}^{P}(RT)$ as follows,

\[
U_{RT}^{P}(RT) = \begin{cases} 
    u_{min} + (1-u_{min}) \frac{RT - RRT_{P}}{IRT_{P} - RRT_{P}}, & \text{if } RRT_{P} \leq RT \leq IRT_{P} \\
    0, & \text{otherwise}.
\end{cases}
\]

$u_{min}$ is the minimum utility that a consumer and a provider receive for reaching a deal at their reserve prices. To differentiate between not reaching an agreement and reaching an agreement at the reserve price, $u_{min}$ is defined as 0.01.

Finally, the aggregated total utility function $U_{Total}(P,TS,RT)$ that includes price, time slot, and response time is as follows,

\[
U_{Total}(P,TS,RT) = \begin{cases} 
    0, & \text{if either } U_{p} = 0, U_{TS} = 0, U_{RT} = 0 \\
    w_{p} \cdot U_{p} + w_{TS} \cdot U_{TS} + w_{RT} \cdot U_{RT}, & \text{otherwise}.
\end{cases}
\]

$w_{p}, w_{TS}$, and $w_{RT}$ are the preference weights for price, time slot, and response time, respectively. A negotiator can adjust the importance among the issues by differentiating the weights.
where the weights satisfy $w_p + w_z + w_r = 1$. A consumer who cares price only, the consumer can assign 1 to $w_p$ and 0 to the other weights. According to the (3), $U_{\text{total}} = U_r$. The consumer can have a high chance to use a service in a low cost.

So, with the SLA negotiation mechanism, cloud participants can narrow preference differences (i.e., service price, time slot, and response time) so that the parties successfully reach an agreement for SLA.

2.3 Workload and location aware resource allocation

A cloud provider needs to properly allocate the provider’s resources to a data center for a service provisioning so that the provider can guarantee the SLA agreed with a consumer. The response time for users depends on the utilization level of physical nodes in the data center and the location of the data center. The performance of VM depends on the allocated PM (Physical machine). If the allocated PM is under heavy workload, the performance of VM will be degraded. In addition, cloud computing services are delivered over the Internet that does not guarantee reliable delivery. Also, the response time may depend on network delay in service delivery. If the distance between a user and a data center is far, the response time of VM will be slow. So, both workload and geographical location are important factors to guarantee the SLA in terms of response time. WLARA selects a data center according to a utility function to evaluate the appropriateness in VM placement to data centers.

2.3.1 Utility function

To find an appropriate physical computing resource which fits user’s SLA, this paper suggests an allocation model considering 1) the geographical location of users and provider and 2) the expected response time of computing resources. The proposed model uses utility function to measure an appropriateness of each data center. The utility function is described as follows:

$$U_m = \alpha \cdot U_m^{UL} + \beta \cdot U_m^{RT}$$  \hspace{1cm} (4)

In (4), there are two terms in the utility function: 1) machine workload utility $U_m^{UL}$, and 2) expected response time $U_m^{RT}$. Each term is multiplied by the preference weight ($\alpha + \beta = 1.0$) respectively. The weight of each value can indicates that a provider’s preference for placing user VM. If $\alpha \rightarrow 1.0$ and $\beta \rightarrow 0$, the provider emphasizes workload of data center in placing VM. Likewise, if $\beta \rightarrow 1.0$ and $\alpha \rightarrow 0$, the provider emphasizes finding a more close data center.

The utility function (5) for machine workload represents level of workload of a data center when the data center accepts VM placement. Let $W_r$ be the upper bound of workload value, and $W_o$ and $W_e$ be the expected workload in the data center including requested VM placement and current workload of the data center respectively. (5) shows higher utility if the placement does not affect increasing the workload of the data center. When $W_o$ is higher than $W_r$ than the utility function returns the minimum utility 0.

$$U_m^{UL} = \begin{cases} \frac{W_o - W_c}{W_r - W_c}, & W_c \leq W_o < W_r \\ 0, & \text{otherwise.} \end{cases}$$  \hspace{1cm} (5)

(6) describes utility function for expected response time $U_m^{RT}$. Let $T_{SLA}$ be the threshold of response time in SLA, and $T_e$ and $T_c$ be the expected response time when a user’s VM is placed in a data center and current average response time by the workload and location of the data center. This utility function returns values when $T_e$ is in between $T_c$ and $T_{SLA}$. Otherwise, the utility returns the minimum utility 0.

$$U_m^{RT} = \begin{cases} \frac{T_o - T_c}{T_{SLA} - T_c}, & T_c \leq T_e < T_{SLA} \\ 0, & \text{otherwise.} \end{cases}$$  \hspace{1cm} (6)

By utility function (4), the provider can find appropriate data center for user’s VM in terms of the workload and the location between user and data center.

2.3.2 Resource allocation strategy

When a provider receives the request from user, the provider asks to a reservation manager, who is in charge of managing resources to allocation and reservation, for evaluating appropriateness of each data centers. To find proper datacenter, the proposed allocation model uses utility function (4). When the reservation manager finds appropriate datacenters by the evaluation, the reservation manger asks to allocate the request of user to coordinator agent, who is in charge of managing numbers of PMs in the datacenter. Then, the coordinator receives resource allocation request, the coordinator also finds a proper PM that has light workload because we assumes that data center consists of number of PMs. Consequently, the request of user is allocated in the PM that closes to user with light workload. Therefore, when the provider uses the proposed allocation model, the request of user is allocating in the PM that guarantees reasonable response time under SLA.

3 Simulation and empirical results

3.1 Simulation test bed

A simulation test bed for the proposed cloud computing framework has been implemented based on JAVA and JADE (Java Agent Development Environment) [15]. JADE is a software framework implemented by JAVA, and it is efficient to implement multi-agent systems with JADE since the implementation of JADE concretely follows FIPA (Foundation for Intelligent Physical Agents) [16] standard. The components of the proposed cloud framework were implemented as agents in the simulation test bed, and the agents communicate each other by exchanging messages needed among components by using message types defined in FIFA. The agents for the framework are as follows: 1) Service broker, 2) Service registry, 3) Reservation manager, 4) Data center coordinator, 5) PMM, and 6) PM.
3.2 Experimental settings

To show performance of the proposed framework, this experiment is focused on evaluating the number consumers who experience any SLA violation (i.e., slower response time than the response time limit described in SLA). Also, we show 1) SLA negotiation outcomes in terms of price and response time (time slot is omitted to focus on the performance of the resource allocation in this evaluation), 2) load distribution to data centers, and 3) distance between user and data center.

Table 1 shows input data source which is assigned to components in the test bed. Using the SLA negotiation options in Table 1, all consumers and provider select preference range for price and response time so that negotiation agents negotiate price and response time. The negotiation outcome regarding response time is then subjected to the limit of response time that the provider guarantees. At the end of each experiment, the response time of each user was calculated and checks whether the provider violated consumers’ SLA or not. Also, consumer agent can have three different types of resources: 1) the number of vCPUs (virtual CPUs of a VM); 2) size of storage and 3) the VM location (Table 1). The test bed generates random value to assign types of resources as consumer’s service request according to the input data range. Since the workload is depended how many virtual CPUs are waiting for allocate to physical CPU in this test bed, the number of vCPUs affects workload, and the limit of workload to a data center is bounded by 2.0. We assumed all datacenters are homogenous (i.e. same amount of resources).

In Table 1, as we described in Section 2.3, the geographical location between a data center and a consumer is an important factor that affects response time. For simulation purpose, we design ten different zones (i.e. zone 0 to 9) that represent geographical distance. In this experiment, a network delay is increasing gradually according to increasing differences between zones (20ms per each hop in the experiments). Each datacenter has a corresponding zone, and consumers can be in different zones as their locations. For each simulation, 1000 consumers, who request different amount of resources and are located in different locations, have been simulated. When a consumer agent is generated, a location is given to the consumer agent. For more realistic simulation, the distribution of locations to the ten zones follows a normal distribution (i.e. mean is 4.5, standard deviation is 2.0) so that some data centers faced swamped situation with high demand while in the experiments.

The proposed framework especially the resource allocation model (i.e. Workload and Location Aware Resource Allocation, WLARA) is compared with related allocation models, which are widely used in resource allocation (i.e., VM placement): 1) Greedy, 2) Random, 3) Round Robin (RR), and 4) Manual Zone Selection. According to the survey in [4], 1) Greedy, 2) Random, 3) RR have been used by Nimbus and Eucalyptus for placing VM to a PM in a data center. In addition, 4) the manual zone selection is a load placement scheme which is similar to current providers such as Amazon EC2. So, with the manual zone selection, consumers themselves manually select a data center to place their virtual machines. Since the manual zone selection assumes a human is aware and selects the closest data center, the manual zone selection used in this simulation makes all consumers’ VM be placed to a data center deployed in the same location (i.e., zone) with each consumer. Manual zone selection is classified into ideal case and non-ideal case for a simulation purpose. With the ideal case (i.e., ideal manual selection, IM), a consumer’s VM is placed to the closest data center. With the non-ideal case (i.e., non-ideal manual selection, NIM), a consumer’s VM is placed to a close data center by applying random values that follow normal distribution so that we can simulate situations where users sometimes make a mistake in selecting the closest data center.

![Figure 2. SLA negotiation outcomes (price and response time).](image-url)
3.3 Empirical results and observations

Fig. 2 shows SLA negotiation outcomes in terms of service price and response time. According to Fig. 2, SLA negotiation mechanism properly worked since threshold of response time is low (fast, respectively) when the price is low (high, respectively). Also, the response times and the prices for SLAs are well distributed (100<Agreed Response Time<250; 0.1<Agreed Price<1.0) according to the given preference range in Table 1. In addition, Fig. 3 shows the performances of all resource allocation models. According to the experiments, the proposed allocation model shows the best performance in terms of the number of SLA violations and allocation failures; 2) a balanced resource loads distribution and distance of data center. The detailed observation is described as follows.

Fig. 3 (a) shows the number of SLA violations and allocation failures. The consumers have different threshold of response time according to the outcomes of negotiated SLA. At the final stage of an experiment, the response time of each consumer’s VM is aggregated to check whether it violate given response time threshold or not. In Fig. 3 (a), when the provider uses the proposed model WLARA, it guarantees the least number of SLA violations (170) and no allocation failure (0) whereas Greedy, Random, RR, NIM, and IM occurred 753, 637, 624 342, 288 SLA violations, respectively. This is because WLARA is considering both workload and response time including network delay by the utility function (4). Hence, WLARA can allocate consumer’s request to a PM in a data center that has less workload and is in a close location to guarantee the threshold of response time in SLA.

Fig. 3 (b) shows load distribution in data centers. Each allocation model shows different trends of load distribution in data centers. To represent the trends, we indicated standard deviations for each model. WLARA shows less biased and smoothly distributes loads in data centers. Since the user’s request is distributed to follow the random normal distribution (i.e. mean is 4.5, standard deviation is 2.0), the provider may be requested more for allocating user’s VM at certain location of data center. Hence, the load distribution trend has basically in a triangle shape like WLARA and IM. The worst case is greedy model. It allocated user requests by order of data centers. Therefore, the load distribution is biased. In Fig. 3 (b), random and RR allocation model shows slight uniform distribution of load and less standard deviation. But, as shown in Fig. 3 (a), since these allocation models are not considering workload and location, the models violated SLAs too frequently.

Fig. 3 (c) shows the average geographical distance between users and allocated data centers. When the provider uses IM
selection, the average distance is zero because the IM model always places VM to the data center which has the same location with each user. IM selection is slightly better than the NIM in SLA violations. However, considering standard deviation of load distribution to data centers in Fig. 3 (b), both NIM and IM show bad performances. This may lead low performances in SLA violations and the number of allocation failures. When user request is not acceptable to selected data center due to the capacity limit, the data center may deny to allocation. The average distance with WLARA was 0 to 1. Although WLARA sometimes does not allocate user’s VM to exactly same location, WLARA shows less number of SLA violations because WLARA considers both workload and response time together.

Fig. 4 represents the performance among WLARA with different experimental settings (the weights α and β in (4)). WLARA uses the utility function (4) that allows provider to adjust each weight of preference (i.e. considering workload more or location more). Hence, the results are slightly different by the weight. In Fig. 4, when the provider uses load-prefer allocation (i.e. load weight α is 0.7, and location weight β is 0.3), the distribution of load shows also best results in Fig. 4 (a). However, Fig. 4 (b) shows WLARA (α =0.7) gives relatively longer distance than with other preferences.

4 Conclusion and future work

This paper proposed a SLA-based Cloud computing framework to facilitate temporal and locational load-aware resource allocation in which a cloud provider has several data centers geographically deployed., and we implement a test bed of the proposed cloud framework to verify the usefulness of the proposed framework in a case study.

The main functionalities of the proposed cloud computing framework include a SLA negotiation mechanism and a resource allocation mechanism. Hence, by using the proposed framework, we expect cloud computing providers can facilitate distributing resource load (i.e., resource demand) in temporal and locational perspectives. In summary the contributions of this paper are as follows: 1) Design of a cloud computing framework to facilitate temporal and locational load aware resource allocation in a cloud computing environment which is similar to Amazon EC2 that has multiple data centers geographically distributed. 2) Implementation of an agent-based cloud computing test bed according to the proposed cloud computing framework. The evaluation in this paper was focused on the performance of the resource allocation algorithm. The empirical result shows WLARA performs better than other related schemes such as greedy, random, round robin, and manual selection like Amazon EC2 in terms of the number of SLA violations.

Using the proposed system, a cloud consumer can establish SLA about service price, time slot, and response time by an automated SLA negotiation scheme, and a cloud provider can facilitate load balancing by a pricing strategy. As future works, the proposed framework requires a thorough evaluation that includes cost effectiveness, resource load, average resource utilization, and so on. In addition, we need to research and classify negotiable cloud SLA issues to be included in the framework considerations.

5 Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2010-0026438) and by PLSI supercomputing resources of Korea Institute of Science and Technology Information.

6 References