J Supercomput (2013) 64:606–637
DOI 10.1007/s11227-012-0861-z

An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider

Seokho Son · Gihun Jung · Sung Chan Jun

Published online: 25 January 2013

Abstract The number of cloud service users has increased worldwide, and cloud service providers have been deploying and operating data centers to serve the globally distributed cloud users. The resource capacity of a data center is limited, so distributing the load to global data centers will be effective in providing stable services. Another issue in cloud computing is the need for providers to guarantee the service level agreements (SLAs) established with consumers. Whereas various load balancing algorithms have been developed, it is necessary to avoid SLA violations (e.g., service response time) when a cloud provider allocates the load to data centers geographically distributed across the world. Considering load balancing and guaranteed SLA, therefore, this paper proposes an SLA-based cloud computing framework to facilitate resource allocation that takes into account the workload and geographical location of distributed data centers. The contributions of this paper include: (1) the design of a cloud computing framework that includes an automated SLA negotiation mechanism and a workload- and location-aware resource allocation scheme (WLARA), and (2) the implementation of an agent-based cloud testbed of the proposed framework. Using the testbed, experiments were conducted to compare the proposed schemes with related approaches. Empirical results show that the proposed WLARA performs better than other related approaches (e.g., round robin, greedy, and manual allocation) in terms of SLA violations and the provider's profits. We also show that using the automated SLA negotiation mechanism supports providers in earning higher profits.

S. Son · S.C. Jun (✉)
School of Information and Communications, Gwangju Institute of Science and Technology,
123 Cheomdan-gwagiro, Buk-gu, Gwangju 500-712, Republic of Korea
e-mail: scjun@gist.ac.kr

G. Jung
Samsung Electronics Co., Ltd., 129 Samsung-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do 443-742,
Republic of Korea

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

1 Introduction

Cloud computing is a new computing paradigm that provides computing resources as a service through a network to cloud users. A cloud consists of a parallel or distributed system with combinations of interconnected computing resources and virtualized computing resources [1]. There are many existing cloud computing environments, such as Amazon EC2 (Amazon elastic cloud computing) and Amazon S3 (Amazon simple storage service) [2]. In addition to Amazon, Google, IBM, Sun Microsystems, and Microsoft are leading IT companies that have launched cloud computing environments. Considering the interests and business willingness of IT leaders, the cloud computing paradigm is important and essential for various industries.

In cloud computing environments, a cloud service user consumes cloud resources as a service and pays for service use. Before a cloud provider provisions a service to a consumer, the cloud provider and consumer (or broker) need to establish a service level agreement (SLA). The SLA is an agreement that specifies the quality of service (QoS) between a service provider and service consumer, and usually includes the service price, with the level of QoS 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 around the world, cloud service providers have deployed geographically distributed data centers. For example, the Amazon Web Service operates globally distributed data centers that support global cloud consumers. Because the resource capacity of a data center is limited, cloud providers need to distribute the resource load to several data centers for system performance and stability. In addition, the resource load can be distributed in a temporal manner because the resource load in a cloud generally fluctuates by time [3]. It is also hard for cloud providers to handle resource demands that exceed the limited resource capacity, so a load balancing scheme is very important to cloud providers in designing a cloud computing framework, and is directly related to cloud providers' profits. While it is important to design a cloud computing framework that facilitates the efficiency and stability of a system, an SLA-based cloud computing framework that considers the load distribution of a cloud provider that operates geographically distributed data centers has not been sufficiently studied to date.

Hence, we propose a cloud computing framework to facilitate location- and load-aware resource allocation, and implement a testbed of the proposed cloud framework to verify the usefulness of the proposed framework in a case study. The contributions of this paper are: (1) the design of a cloud computing framework to facilitate locational- and load-aware resource allocation, and (2) the implementation of a testbed of the proposed cloud computing framework. Using the proposed system, a cloud consumer can establish the SLA regarding service price and response time through an automated SLA negotiation scheme, and a cloud provider can facilitate load balancing through a pricing strategy. As such, the purpose of this work is to design a cloud computing framework that facilitates resource allocation.

This paper is organized as follows. In Sect. 2, we discuss related works on establishing and managing SLAs in cloud computing environments. Section 3 presents both the SLA negotiation mechanism and location- and load-aware resource allocation. In Sect. 4, we introduce the implementation of a simulation testbed for the proposed framework, and Sect. 5 presents an evaluation of the performance of the proposed framework in terms of the SLA violation. Finally, Sect. 6 concludes and includes a discussion of future research.

2 Related work

In this section, we review existing work on establishing and managing SLAs in cloud computing environments. Moreover, we investigate resource allocation strategies (i.e., virtual machine placement) for distributed data centers and the SLA-based cloud computing framework that includes load balancing in global data centers.

Several proposals in the literature are based on utility functions to dynamically allocate resources. Minarolli and Freisleben [4] proposed a scheme that dynamically allocates resources to virtual machines such that the quality of service constraints are satisfied and operating costs are minimized. In [4], a two-tier resource management approach based on adequate utility functions was presented. Minarolli and Freisleben [4] focused on the CPU as the resource to be allocated, defining the virtual machine (VM) utility function as a linear function of the CPU resources the VM of a consumer gets from the cloud provider. Walsh et al. [5] and Menasce and Bennani [6] present approaches for using utility functions in autonomic computing systems to solve resource allocation tasks. Chase et al. [7] consider utility functions to dynamically allocate resources and save energy by consolidating the load. Van Nguyen et al. [8] propose an autonomic resource controller that finds an optimal number and size of VMs to allocate CPU resources to applications in order to maximize the utility function. The proposed scheme in the current paper (i.e., workload- and location-aware resource allocation scheme, or WLARA) is similar to [4–7] in the sense that WLARA is also based on utility functions to dynamically allocate resources to data centers. However, whereas [4–7] focused on dynamic resource allocation in a data center without considering a global network delay, the focus in this paper on dynamic resource allocation is selecting a data center among globally distributed data centers.

Among the problem solving approaches, [9] seems to be the closest work to the current paper. In [9], large data centers host several application environments (AEs) that are subject to various workloads. The data center servers may need to be dynamically redeployed among the various AEs in order to optimize some global utility function. Similar to the proposed WLARA scheme, the resource allocation scheme in [9] is based on a utility function. However, [9] do not explicitly consider a cloud computing environment. In addition, whereas the present paper considers the response time, including network delay, [9] consider only the workload response time (although they modeled a deliberative workload response time). Bennani and Menasce [9] also do not focus on a resource allocation as selecting a data center among globally distributed data centers.

Researchers have investigated maximizing the profit under SLA. Shi and Hong [10] investigated VM placement in a data center, formulating a multilevel generalized assignment problem for maximizing the profit under SLA and the power budget constraint based on the model of a virtualized data center, applying a first-fit heuristic. Breitgand and Epstein [11] also considered VM placement at data centers, but formulate the problem as a combinatorial auction problem and solve a relaxed linear programming problem with the column generation method. However, whereas [10, 11] focused on resource allocation in a data center to optimize profits, they do not consider the role of dynamic resource allocation to be selecting a data center among globally distributed data centers, guaranteeing SLA including the service response time.

Researchers have also investigated virtual machine placement across multiple cloud providers, such as the work by [12] on optimal virtual machine placement across multiple cloud providers. Chaisiri et al. [12] proposed an optimal virtual machine placement algorithm to minimize the cost for consumers who wants VM services from multiple cloud providers under future demands and price uncertainty. In addition, [13] proposed multiobjective metaheuristics for scheduling applications with high availability requirements and cost constraints across multiple cloud providers. Other researchers have explored cloud brokering mechanisms across multiple providers for the optimized placement of virtual machines across multiple providers [14]. However, whereas [12–14] focused on resource allocation in selecting a cloud provider, there are no considerations of dynamic resource allocation in selecting a data center among globally distributed data centers, guaranteeing the service response time.

In [15], Sotomayor et al. provided a comparison of OpenNebula with several well-known virtual infrastructure managers, including Amazon EC2, vSphere, Nimbus, Eucalyptus, and oVirt. The comparison includes several resource placement policies (i.e., resource allocation policies) of virtual infrastructure managers, such as static-greedy resource selection, round robin resource selection, and placement considering average CPU load. However, whereas the resource placement schemes are focused on selecting a physical machine (PM) to place resources at a data center, they are not focused on resource placement where a cloud service provider operates geographically distributed data centers. With geographically distributed data centers, we need to consider response time violations because of the geographical distance. Buyya et al. investigated energy-aware resource provisioning and allocation algorithms that provision data center resources to client applications to improve the energy efficiency of the data center without violating the negotiated SLA [16]. However, whereas [16] provides some research direction regarding resource allocation, the work does not consider a cloud provider that operates multiple geographically distributed data centers to balance the resource load and response time by geographical distance, and does not show a specific negotiation mechanism.

In [17], Le et al. considered load placement policies on the cooling and maximum data center temperatures among cloud service providers operating multiple geographically distributed data centers. Whereas [17] proposes dynamic load distribution policies that consider electricity-related costs as well as transient cooling effects, there is no focus on the SLA guarantees. Le et al. [17] noted that the placement policies

delay jobs to avoid overheating or overloading data centers and sometimes violate SLAs in their simulation configuration. Moreover, current providers such as Amazon EC2 do not employ sophisticated VM placement policies for global data centers, and consumers themselves manually select a data center at which to place their virtual machines.

There are several automated negotiation mechanisms for grid or cloud resource negotiation (see [18] for a survey). These negotiation mechanisms are designed for price negotiation, but fail to consider other SLA issues such as service time slot and response time. Also, whereas [19–22] proposed SLA-based grid/cloud computing, the schemes in [19–22] do not consider the temporal and locational load distributions of geographically distributed data centers.

As such, we propose a cloud computing framework to facilitate location- and load-aware resource allocation, and implement a testbed of the proposed cloud framework to verify the usefulness of the proposed framework through a case study. The contributions of this paper are: (1) the design of a cloud computing framework to facilitate locational- and load-aware resource allocation (i.e., an initial VM placement scheme), and (2) the implementation of a testbed of the proposed cloud computing framework. Using the proposed system, a cloud consumer can establish the SLA regarding service price, time slot, and response time through an automated SLA negotiation scheme, and a cloud provider can facilitate load balancing using a pricing strategy.

Some of the preliminary results of this work were presented in [23], and the present work significantly and considerably augmented and generalized the preliminary work in [23]. Whereas [23] presented preliminary experimental results for a very small set of simulations, we report a considerably larger set of experiments a well as thorough, detailed analyses of the performance of the burst modes against other related schemes in terms of the number of SLA violations and the provider's profit (Sect. 5). In addition, Son et al. [23] omitted many details on the SLA negotiation mechanism and implementation details for the proposed cloud testbed, while in this paper, a very detailed negotiation mechanism and implementation details are provided in Sects. 3.2 and 4. Moreover, as a short conference paper, Son et al. [23] omitted comparisons with related works.

3 A cloud computing framework to facilitate resource allocation

Figure 1 shows a cloud computing environment that includes the proposed cloud computing framework to provide Infrastructure as a Service (IaaS), where cloud service consumers run their applications on VMs offered by a cloud provider and pay for the virtual resources. This cloud computing environment consists of: (1) cloud service providers, which have multiple geographically distributed data centers and manage data centers through a management system; and (2) cloud service consumers.

Cloud service consumers can find a service provider that has the service the consumer requires through a service brokering agent. Before the service consumer starts to use cloud services, it is necessary to establish an SLA, and the service provider needs a system that guarantees the agreed-upon SLA to the consumer. A primitive way to establish an SLA is to exchange an SLA specification through SLA documents. Whereas this is a way to express detailed requirements, it requires more

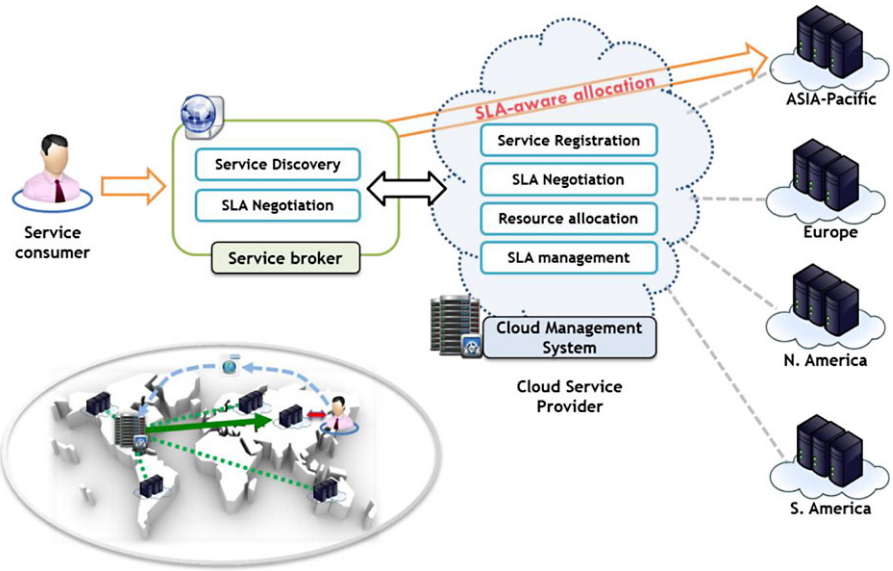


Fig. 1 Cloud computing environment for the proposed cloud computing framework

time and manpower to reach an agreement. Currently, most existing cloud computing providers give SLA options and let consumers select options. However, the variety of SLA options is limited for consumers, so consumers are restricted to specific preferences that meet their utility.

Cloud providers and consumers are independent bodies, and generally have different preferences in leasing or lending services. The most understandable example of this is in regards to the price of a cloud service, as service providers consider the return on investment and hope to lend cloud services at as high of a price as possible, while service consumers hope to reduce the cost of leasing services as much as possible. Negotiation is the process by which a group of agents come to a mutually acceptable agreement on some matter [24]. Using a negotiation mechanism for SLA, a consumer and provider can effectively resolve or narrow such competing perspectives in cloud SLA.

In this paper, therefore, the proposed cloud framework adopts an automated SLA negotiation mechanism to support the establishment of SLA. Whereas the variety of SLA options is limited for consumers within enforced SLA strategies, the different preferences of a consumer and provider can be efficiently narrowed through an automated SLA negotiation mechanism. For example, in regards to agreeing on the price of a service, a negotiation mechanism between a consumer and provider can find an equilibrium state for the price, satisfying both parties. In addition, considering that the primitive way to establish SLA was through document exchange, automated negotiation may help cloud participants conveniently establish SLA. Whereas there should be many SLA issues or options in cloud services in practice, we focus on: (1) service price, and (2) 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 management scheme to guarantee established SLA with a consumer. In the proposed cloud framework, the service response time to consumers is quality of service (QoS). To guarantee the service response time in the agreement with a consumer, the SLA management scheme in the proposed framework considers the geographical distance between a consumer and the distributed data centers of the cloud provider when the provider selects a data center to allocate resources (services) for the consumer.

Therefore, the proposed cloud framework includes the SLA negotiation mechanism, which is designed for the service price and response time issues, and the SLA management scheme for cloud resource allocation. Section 3.1 includes details on the design of the proposed cloud computing framework. The SLA negotiation mechanism for service price, time slot, and response time issues is explained in Sect. 3.2, and Sect. 3.3 includes the design of a location-based resource allocation system.

3.1 Design of the SLA-based cloud computing framework

As shown in Fig. 2, the proposed framework consists of a cloud service broker and cloud service provider. The cloud service broker is an interface between a consumer and cloud service provider. The cloud service broker obtains information on a consumer's service preferences and proceeds with cloud service discovery to find a service that matches the consumer's preferences. After the broker finishes the service discovery, he or she connects with the cloud provider who owns the service that was discovered. The next step is creating the SLA between the cloud service provider and service broker on behalf of the consumer through the SLA negotiation component in Fig. 3. If the negotiation regarding service price and response time is successful, the consumer pays for the service (at the agreed-upon price) and reserves the service with the provider.

A cloud service provider consists of four components: (1) reservation controller, (2) SLA negotiation mechanism, (3) SLA management, and (4) distributed data center. The cloud service provider responds to the service broker's service discovery requests. In the SLA negotiation component (Sect. 3.2 in detail), the provider and broker establish the terms of the SLA. If the automated negotiation is successful (they reach a mutually acceptable agreement regarding price, time slot, and response time), the SLA is established and the provider receives the payment from the consumer through the broker.

The service reservation controller reserves the requested service using the following information: name of the service consumer, service type, start time of service (start time of the reservation), end time of the service (end time of the reservation), and the threshold of the response time to the service reservation queue, which is the reservation list. The reservation controller sequentially checks the reservation queue to initiate the start of the services based on the agreement. The service reservation controller forwards the agreed-upon service response time for services to the SLA management component.

The SLA management component, which is called workload- and location-aware resource allocation scheme (WLARA) (Sect. 3.3 in detail), selects a data center among the distributed data centers to allocate the requested service. The conditions

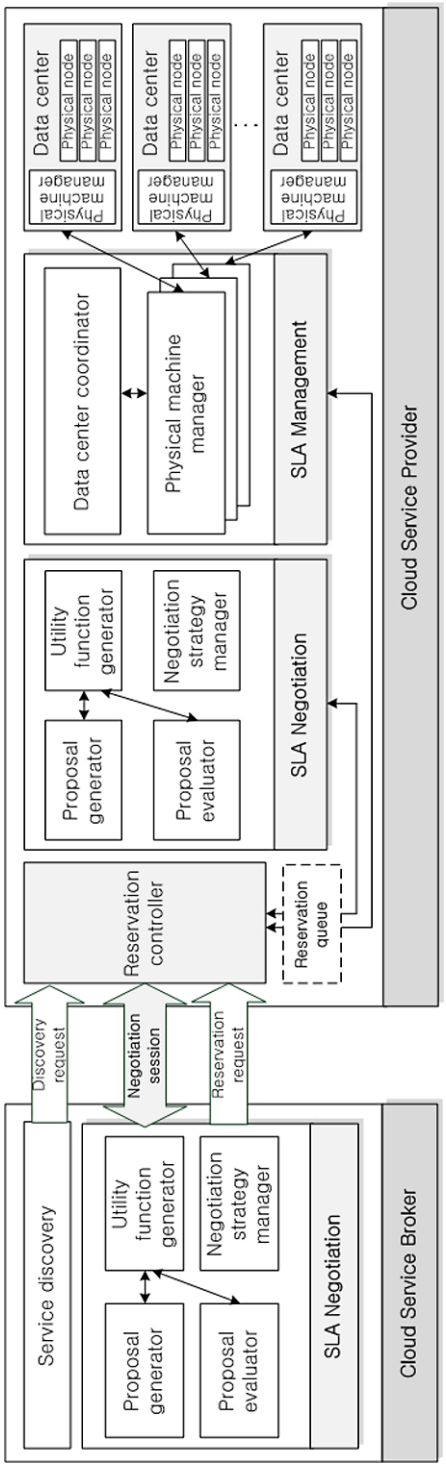


Fig. 2 Design of SLA negotiation and management based cloud computing framework

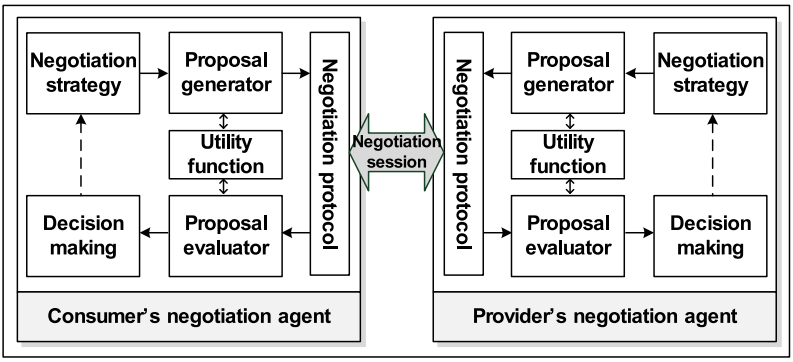


Fig. 3 An automated negotiation mechanism (bilateral)

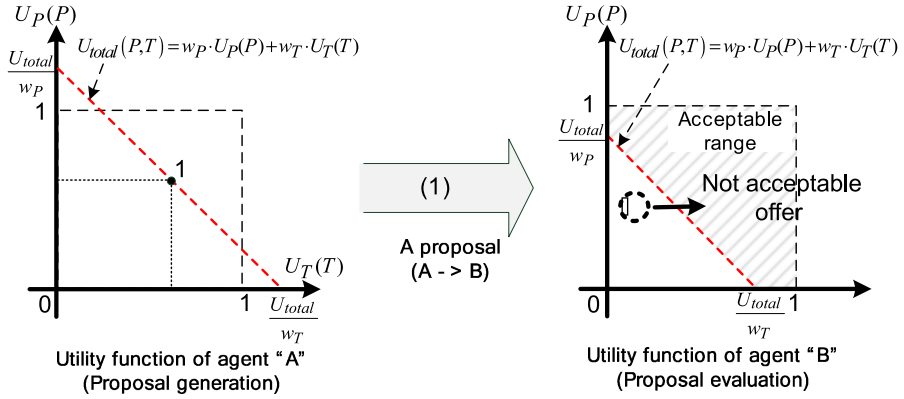
(i.e., utility-based evaluation) of selecting a data center are the data center's workload and specified in the SLA (the management in this paper focuses on service response time). Each data center includes a physical machine manager, who manages the physical computing nodes of a data center to evaluate the average response time of a data center to the service consumer. The SLA management component selects a data center and specific physical computing node using the average response times and the workload.

Furthermore, the SLA management component continuously monitors the response time of the physical computing nodes for resource migration. If a physical computing node is expected to have a hard time meeting a consumer's SLA, the allocated resource for the consumer can be reassigned to a physical computing node that is capable of guaranteeing the consumer's SLA through migration. In conclusion, by using the proposed cloud computing framework, a service provider and service consumer can establish the SLA about service price, time slot, and response time, and the service provider can guarantee the established SLA with a consumer through the SLA management component.

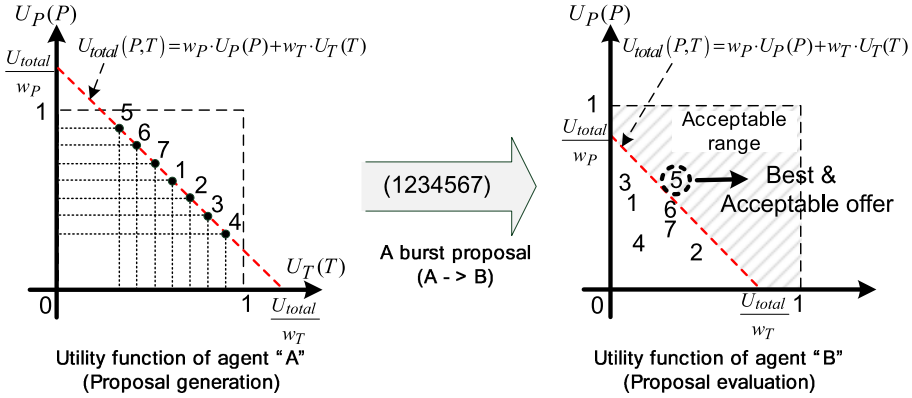
3.2 Automated SLA negotiation mechanism

In general, a negotiation mechanism consists of the negotiation protocol, negotiation strategy, and utility functions (Fig. 3). The negotiation protocol is a set of rules on communication (e.g., possible actions, language, and utterance turn) for negotiations among involved parties. The negotiation mechanism in this work follows Rubinstein's alternating offers protocol [25], which lets agents make counter-offers to their opponents in alternate rounds. Both agents generate counter-offers and evaluate their opponent's offer until either an agreement is made or an agent's negotiation deadline is reached.

Counter-proposals are generated according to the negotiation strategy, which consists of a concession algorithm and tradeoff algorithm. When an agent generates a counter-proposal, the agent needs to concede a proposal because it is hard to reach an agreement without a concession. A concession algorithm determines the amount of concession for each negotiation round [24], and a tradeoff algorithm is required to



(a) Negotiation without the burst proposal



(b) Negotiation without the burst proposal

Fig. 4 A trade-off algorithm to increase the negotiation speed and negotiation utility

generate a proposal in multiissue (i.e., multiattributes) negotiation. In multiissue negotiation, there are multiple issues to negotiate, and the tradeoff algorithm generates a proposal by combining the proposals for individual issues. If the negotiation issues are price and response time, a proposal can be a combination of price and response time (e.g., low price and slow response time, or high price and face response time).

This paper is focused on price, time slot, and response time, so the negotiation mechanism needs to include a tradeoff algorithm. In [26], which is a previous work of this paper, a multiissue negotiation mechanism for price and time slot negotiation was presented. The tradeoff algorithm in [26] called the "burst proposal" was proposed to increase the negotiation speed and negotiation utility (i.e., satisfaction with the negotiation outcome). The proposed cloud computing framework in the current paper is also based on the tradeoff algorithm in [26].

Figure 4 introduces the proposed 'burst proposal' in [26]. With the burst proposal, agents are allowed to concurrently make multiple proposals, which each proposal

consisting of a different pair of price and time slot that generates the same aggregated utility. Therefore, whereas an agent using a typical negotiation mechanism can offer a proposal of a single combination (Fig. 4(a)) at a negotiation round, an agent using a negotiation mechanism with the burst proposal can offer multiple proposals consisting of multiple combinations at a negotiation round (Fig. 4(b)). Figure 4 shows an example of a proposal generation procedure of agent "A" and a proposal evaluation procedure of agent "B." The procedures are represented in utility graphs, and the straight line in utility graphs means the current utility expectation which changes according to a concession algorithm. Figure 4(a) shows a negotiation without the burst proposal. Without the burst proposal, a combination of subproposals (price and time slot) is generated and evaluated. In Fig. 4(a), the proposal from "A" is not acceptable for "B" since the proposal is not satisfied with "B" (i.e., the utility of the proposal from "A" is lower than the current utility expectation of "B"). However, we can observe a different situation with the burst proposal. In Fig. 4(b), "A" can generate and offer seven combinations of sub-proposals (price and time slot). Since one of combinations (the fifth combination) is acceptable to "B," the negotiation can be successful. Experimental results in [26] show the trade-off algorithm is effective to increase the negotiation speed and negotiation utility.

Finally, the utility function $U(x)$ represents an agent's level of satisfaction with a negotiation outcome x. For example, the price attribute can be x, and $U(\text{price})$ represents an agent's level of satisfaction with a negotiation outcome on price. When an agent receives a price proposal, the agent evaluates the price proposal according to the price utility function.

A negotiator (i.e., a cloud participant) specifies the 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 the utility function is $\{0\} \cup [u_{\min}, 1]$, $U(\text{price}) = u_{\min}$ represents the least preferred price (RP), and $U(\text{price}) = 1$ represents the most preferred price (IP). If the price is outside of IP and RP, then $U(\text{price})$ is 0. $u_{\min}$ is the minimum utility that a consumer and provider receive for reaching a deal at their respective reserve prices. To differentiate between not reaching an agreement and reaching an agreement at the reserve price, $u_{\min}$ is defined as 0.01 for the simulation.

To support cloud users in the negotiation on service price, time slot, and response time in the proposed framework, it is necessary to define the utility functions for the negotiable SLA issues. In this paper, the utility functions defined in [26] are adopted (the price utility function $U_P(P)$ and time slot utility function $U_{TS}(TS)$). The time slot utility function defined in [15] supports cloud participants in representing the temporal preferences for leasing/lending cloud services. For example, a consumer can specify the time slot utility function according to the consumer's work schedule, and a provider can specify the time slot utility function according to the expected resource demands by time.

In addition to the price and time slot issues considered in [26], we consider service response time as a negotiation issue for the SLA in the current paper. The service response time represents the minimum response time that a provider offers. Let the initial response time (IRT) and reserve response time (RRT) be the most preferred response time and the least preferred response time, respectively.

The response time (RT) given to a consumer can be evaluated by the response time utility function of a consumer $U_{RT}^{C}(RT)$ as follows:

$$U_{RT}^{C}(RT) = \begin{cases} u_{\min} + (1 - u_{\min}) \cdot |\frac{RRT_C - RT}{RRT_C - IRT_C}|, & IRT_C \leq RT \leq RRT_C, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

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

$$U_{RT}^{P}(RT) = \begin{cases} u_{\min} + (1 - u_{\min}) \cdot |\frac{RT - RRT_P}{IRT_P - RRT_P}|, & RRT_P \leq RT \leq IRT_P, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

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

$$U_{\text{Total}}(P, TS, RT) = \begin{cases} 0, & \text{if either } U_P = 0,\ U_{TS} = 0,\ U_{RT} = 0, \\ w_P \cdot U_P + w_{TS} \cdot U_{TS} + w_{RT} \cdot U_{RT}, & \text{otherwise.} \end{cases} \quad (3)$$

w_P, w_{TS}, and w_{RT} are the preference weights for price, time slot, and response time, respectively. Each weight can be adjusted to represent the importance of the corresponding negotiation issue while the weights satisfy $w_P + w_{TS} + w_{RT} = 1$. For instance, a consumer, who cares about budget only, can assign 1, 0, and 0 to w_P, w_{TS}, and w_{RT}, respectively. According to (3), $U_{\text{Total}}(P, TS, RT) = U_P$ so that the consumer can represent U_P is only the valuable utility for he/she.

In summary, using the SLA negotiation mechanism can narrow the preference differences (i.e., service price, time slot, and response time) between a consumer and provider so that the parties successfully reach a consensus for the SLA.

3.3 Workload- and location-aware resource allocation

A cloud provider needs to properly allocate the provider's resources to a data center for service provisioning so the provider can guarantee the SLA 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 physical machine (PM). If the allocated PM is under a heavy workload, the VM performance will be degraded. In addition, cloud computing services are delivered over the Internet, so reliable delivery is not guaranteed, and the response time may depend on network delays in service delivery. If the distance between a user and a data center is far, the VM response time will be slow. Thus, 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 in order to evaluate the appropriateness of VM placement to data centers.

3.3.1 Utility function

To find an appropriate physical computing resource that fits the user's SLA, we suggest an allocation model considering: (1) the geographical location of users and the provider, and (2) the workload of computing resources. The proposed model uses a utility function to measure the appropriateness of each data center. The utility function is described as follows:

$$U_m = \alpha \cdot U_m^{UL} + \beta \cdot U_m^{RT}. \tag{4}$$

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

The utility function of the data center workload U_m^{UL} is designed to represent the level of workload of a data center when the data center accepts VM placement. We define the data center workload as the average of individual workloads for each PM in the data center. Let W_i, p_i, and a_i be the individual workload of PM_i, the number of the physical threads of CPU in PM_i, and the number of allocated virtual CPUs (vCPUs) in PM_i, respectively, where i is ID of a PM. Then, the individual workload of a PM W_i is represented as follows:

$$W_i = \frac{a_i}{p_i}. \tag{5}$$

In a virtual machine monitor (i.e., virtual machine hypervisor) such as Xen hypervisor [27], a physical thread consists of logical threads, and logical threads are assigned to a VM as vCPUs. In an existing OS system such as Linux, the system evaluates a workload by measuring the number of threads waiting for an allocation to physical threads. Likewise, we assume that the workload of a PM depends on the number of logical threads (i.e., vCPUs) a_i that have been allocated to physical threads p_i in a PM. For example, a PM has eight physical threads ($p_i = 8$) and no vCPU ($a_i = 0$) is allocated to a VM yet. If new VM is required to allocate four vCPUs ($a_i = 4$), the PM is under 50 % of workload according to Eq. (5). If $p_i < a_i$, there will be waiting vCPUs, which means W_i can be over 100 %. Accordingly, the work load in a data center W is the average of individual workloads for each PM in the data center.

To formulate the utility function of the data center workload U_m^{UL}, let W_τ be the upper bound of the workload value, and W_θ and W_C be the expected workload when the provider accepts user's VM in the data center and current workload of the data center, respectively. According to Eq. (6), U_m^{UL} returns a higher utility if a placement does not increase the workload of the data center. When W_θ is higher than W_τ, the utility function returns the minimum utility 0 to prevent an unacceptable VM allocation to the data center.

$$U_m^{UL} = \begin{cases} 1 - \frac{W_\theta - W_C}{W_\tau - W_C}, & W_C \le W_\theta < W_\tau, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Equation (7) describes the utility function for expected response time U_m^{RT}. To formulate the utility function for expected response time, we consider major delays (response delay by the workload and location of the data center) in servicing computational resources through a network. The response time delay by the workload occurs when the current workload is over 1.0 ($W_C > 1.0$) because of threads scheduling, and

the response delay by the location is network latency between the potential user and the data center.

Let T_{SLA} be the response time threshold in the SLA, and T_e and T_C be the expected response time when a user's VM is placed at a data center and the current response time by the workload and location of the data center, respectively. When T_e is not between T_C and T_{SLA}, the utility returns the minimum utility 0 to represent the user request cannot be satisfied with the data center.

$$U_m^{RT} = \begin{cases} 1 - \frac{T_e - T_C}{T_{SLA} - T_C}, & T_C \leq T_e < T_{SLA}, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Based on the utility function in Eq. (4), the provider can determine an appropriate data center for the user's VM in terms of the workload and location between a user and data center.

3.3.2 Resource allocation strategy

When a provider receives a service request from a user, the provider asks the reservation manager (who is in charge of managing resources in regards to allocation and reservation) to evaluate the appropriateness of each data center. To find an appropriate data center, the proposed allocation model utilizes the utility function in Eq. (4). When the reservation manager finds appropriate data centers based on the utility evaluation, the reservation manager asks the coordinator agent (who manages the number of PMs in the data center) to allocate the request of the user. The coordinator then receives the resource allocation request and also finds a PM that has a light workload, because we assume that the data center consists of a number of PMs. Consequently, the request of the user is allocated in the PM that is closest to the user and has a light workload. Therefore, when the provider uses the proposed allocation model, the request of the user is allocated in the PM that guarantees a reasonable response time under the SLA.

4 Implementation

A simulation testbed for the proposed cloud computing framework has been implemented based on JAVA and JADE (Java Agent Development Environment) [28]. JADE is a software framework implemented by JAVA, and is efficient for implementing multi-agent systems because the implementation of JADE concretely follows the Foundation for Intelligent Physical Agents (FIPA) [29] standard, which is a well-defined description of the software agent. The components of the proposed cloud framework were implemented as agents in the simulation testbed, and the agents communicate with each other by exchanging messages among the components using the message types defined in FIFA.

The agents for the framework that follow the interactions described in Fig. 5 are as follows: (1) service broker, (2) service registry, (3) reservation manager, (4) data center coordinator, (5) physical machine manager, and (6) physical machine.

The service broker agent is an implemented object of the service broker in the proposed framework, and the service registry is implemented only for the simulation

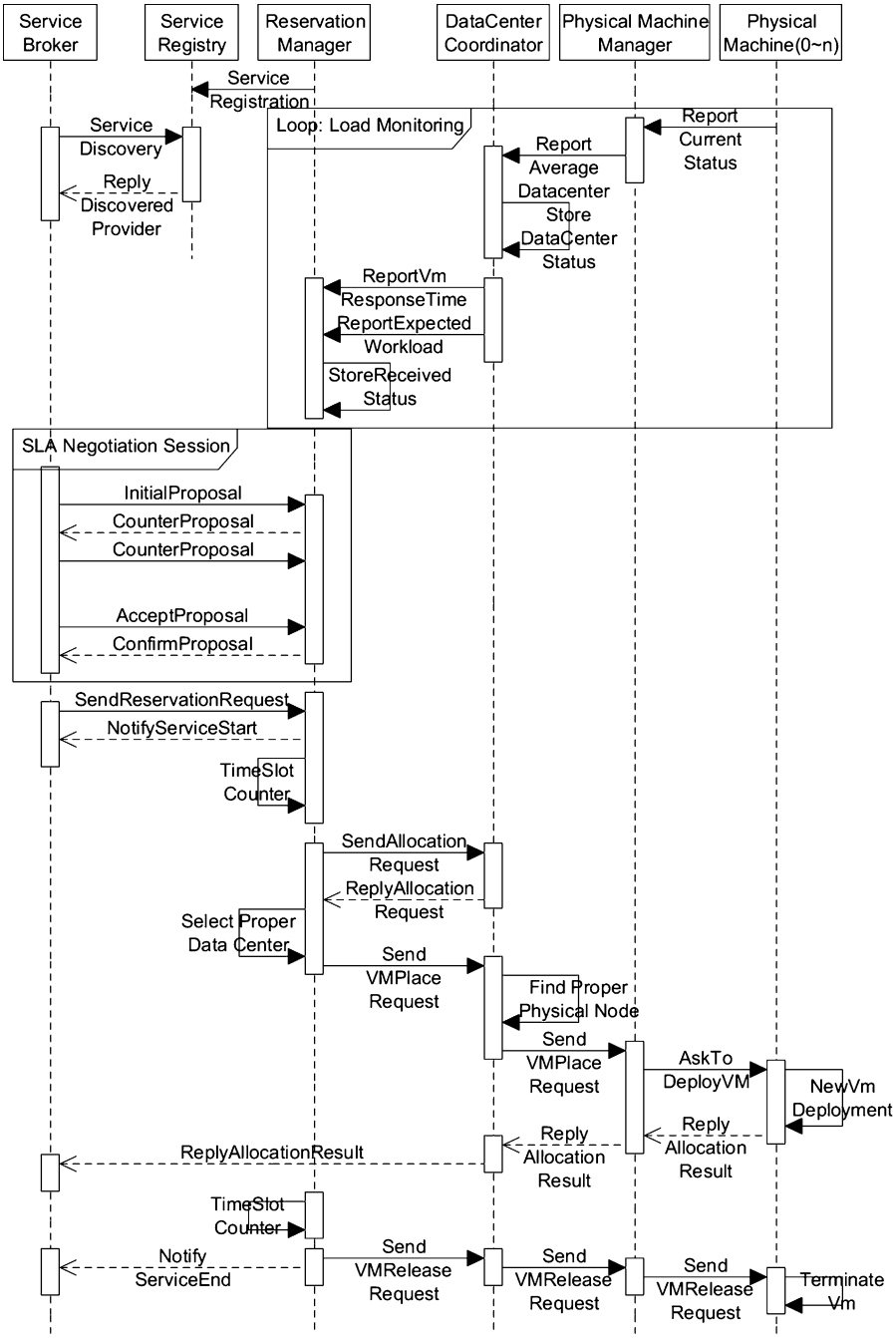


Fig. 5 UML diagram for the implementation

testbed. As cloud service providers, agents can register their services to the service registry (i.e., providers can advertise their services using the service registry). The service broker, on behalf of a service consumer, can search for a service the consumer wants and find providers that lease the services by sending a message to the service registry (for simplicity, the testbed lets the service broker select a provider among the discovered providers). The implementation of the service registry is implemented using the directory facilitator (DF) constructed in JADE in order to keep the agent information.

The other agents (i.e., reservation manager, data center coordinator, and physical machine manager) are the components of a cloud service provider in the proposed framework. The reservation manager delegates the service provider to communicate with the service broker of a consumer. The service reservation manager sends a "service registration" message to register the provider's service list in the service registry, and is made aware of the workload of physical machines in the data centers by managing the monitoring messages. In this procedure, the data center coordinator first receives the monitoring message, and the message is propagated to physical machines through the physical machine manager. The information from the physical machines is then periodically reported to the service reservation manager backward.

After the service broker requests SLA negotiation, the service broker and service provider establish a negotiation session, according to the SLA negotiation mechanism described in Sect. 3.2. When the SLA negotiation is successful and the SLA agreement is completed, the service broker requests a service reservation on behalf of a consumer according to the agreed-upon SLA standard (the service price, the start time of the service, the end time of the service, and the guaranteed service response time), and the service broker pays for the reserved service. The reservation manager updates the reservation queue, which lists reserved services, with the service that consumer requested. The reservation manager continues to check the reservation queue to find services that need to be started according to the start times. When a service needs to be started, the reservation manager sends messages to the internal components (e.g., data center coordinator) to allocate resources for provisioning the service. The implementation of the resource allocation follows the design of SLA management described in Sect. 3.3.1. After the resource allocation procedure is completed, the service manager sends access information (e.g., IP address, username, password, and so on) to the service broker so the consumer can start to use the cloud service. The final stage is when the consumer's reservation is over and the service ends. The reservation manager makes an order to terminate the service so that the physical machines, which provide the resource, release the resource allocation. In addition, the reservation manager sends a "service termination" message to the service broker and ends the accessibility of the consumer.

5 Simulations and empirical results

5.1 Objectives and motivations

A series of experiments were conducted to evaluate and compare the performance of the proposed framework using the testbed described in Sect. 4. The experiments are

Table 1 Input data source

SLA negotiation options	Consumer	Provider
SLA range for price	IP:0.1 ($/hour), RP:1.0 ($/hour)	IP:1.0 ($/hour), RP:0.1 ($/hour)
SLA range for response time	IRT:100 (ms), RRT:250 (ms)	IRT:250 (ms), RRT:100 (ms)

Consumer's resources	Range of values
Number of vCPUs	[1, 32]
Size of storage	[10.0, 1,000.0]
Response time of SLA	[100, 250]
Location	{zone 0, . . . , zone 9}

Physical machine resources	Values
Number of CPUs	{4}
Number of threads per CPU	{8}

Data center specification	Values
Number of physical machines	{100}
Storage capacity	{1,000,000.0}
Number of monitoring agents	{10}
Number of coordinator agents	{1}
Location	{zone 0, . . ., zone 9}

designed for evaluating and comparing the performance of the proposed framework, especially for the proposed resource allocation scheme.

5.2 Performance measure and experimental settings

To assess the performance of the proposed framework, this experiment is focused on evaluating the number of consumers who experience any SLA violation (e.g., slower response time than the response time limit agreed to in the SLA). Also, we show the following: (1) SLA negotiation outcomes in terms of price and response time (note that time slot is omitted to focus on the performance of resource allocation scheme in this evaluation), (2) load distribution to data centers, and (3) average distance between users and data centers.

Table 1 shows the input data source assigned to the components in the testbed. Using the SLA negotiation options in Table 1, all consumers and providers select a preference range for the price and response time, which negotiation agents use to negotiate the price and response time. The negotiation outcome regarding response time is then subjected to the response time limit that the provider guarantees. At the end of each experiment, the response time of each user is calculated and whether the provider violated consumers' SLA is assessed. A consumer agent has three types of resources: (1) number of vCPUs (virtual CPUs of a VM), (2) size of storage, and

Table 2 Experimental constraints

Experimental constraints		Values
Allocation model	*Reservation manager*	Greedy, Random, Round Robin (RR), Ideal Manual (IM), Non-Ideal Manual (NIM), WLARA
	Coordinator and monitoring agent	{Adaptive}
Number of user requests	{1,000}	
Number of data centers	{10}	
Limitation of workload in physical machines	{2.0}	

(3) the VM location (Table 1). The testbed generates random values to assign the types of resources for a consumer's service request according to the input data range. The workload is dependent on how many virtual CPUs are waiting for allocation to the physical CPU in this testbed, so the number of vCPUs affects the workload, and the workload limit of a data center is bounded by 2.0. Table 1 describes the specification of the physical machine and data center. We assume that every data center has the same amount of physical resources (i.e., homogeneous data centers).

Table 2 shows the experimental constraints. As described in Sect. 3.3, the geographical location between a data center and consumer is an important factor that affects the response time. For simulation purposes, we design 10 zones (i.e. "zone 0" to "zone 9") that represent geographical distance. Each data center has a corresponding zone, and consumers can be located in different zones.

For practical experiments, we simulated the network latency among nations using the network latency table from the Verizon global network statistics [30]. We assigned the network latency of each nation to the designated data center. Table 3 shows the global network latency table applied to this experiment. A data center in a zone is aware of the network latency to a consumer in a zone. For example, the network latency between a data center in the USA and a service consumer in Korea is 153 ms. In the case of a consumer and data center in the same zone, we assigned 20 ms to the network latency [30]. Also, we simulated workload pattern to data centers. The workload pattern is based on the arrival rate of consumers' requests. By controlling the request arrival rate periodically, a fluctuating workload pattern is generated during the simulations. The workload pattern includes most of typical workload patterns such as (1) peak workload, (2) off-peak workload, (3) increasing workload, and (4) decreasing workload so that the evaluation of the proposed system is based on reasonable and practical experiments.

Each simulation used 10,000 consumers who request different amounts of resources and are located in different locations. When a consumer agent is generated, a location (i.e., zone) is given to the consumer agent. For more realistic simulation, the distribution of locations to the 10 zones follows a normal distribution (i.e., mean is 4.5, standard deviation is 2.0) so that some data centers face a high-demand situation in the experiments. By controlling consumers' request arrival rates, we simulated both a high load situation (Fig. 6) and low load situation (Fig. 13). In a high load situation, the simulation has conducted for 2,230 minutes, and in a low load situation,

Table 3 Network latency among globally distributed data centers (Verizon global network [30])

	Australia	Hong Kong	India	Japan	Korea	New Zealand	Singapore	UK	USA	Taiwan
Australia	20	138	166	116	153	25	106	300	156	155
Hong Kong	138	20	89	76	43	157	32	234	195	22
India	166	89	20	137	134	185	62	145	269	112
Japan	116	76	137	20	33	142	77	256	186	34
Korea	153	43	134	33	20	175	93	288	153	65
New Zealand	25	157	185	142	175	20	127	306	181	176
Singapore	106	32	62	77	93	127	20	206	201	55
UK	300	234	145	256	288	306	206	20	110	265
USA	156	195	269	186	153	181	201	110	20	148
Taiwan	155	22	112	34	65	176	55	265	148	20

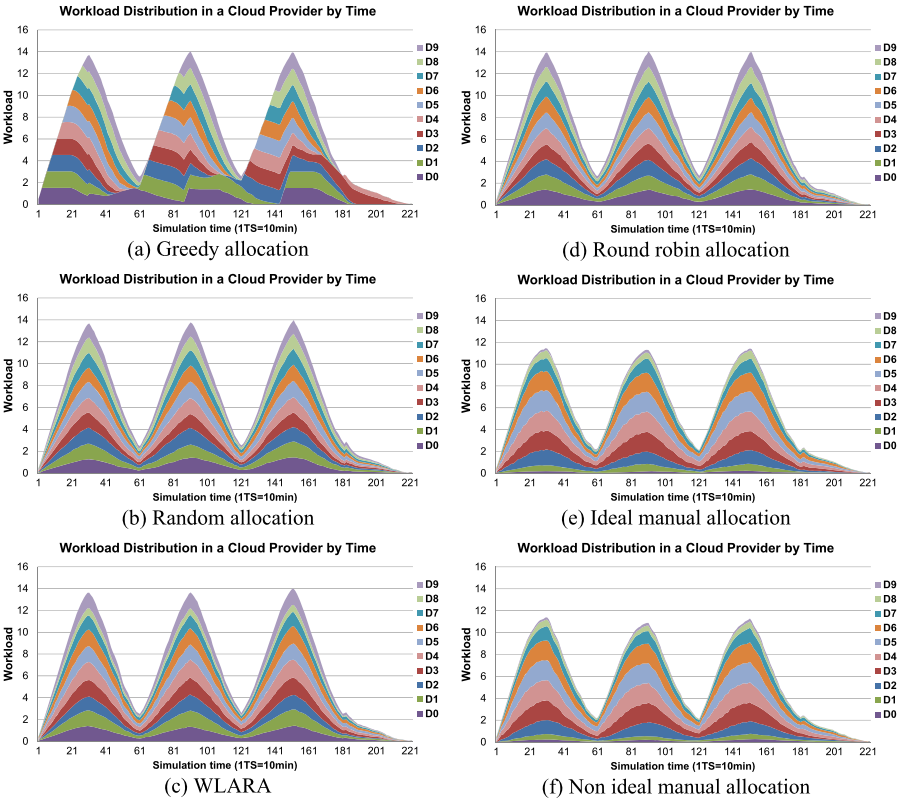


Fig. 6 Overall workload distributions

the simulation has conducted for 3,030 minutes for provisioning services to 10,000 consumers.

The proposed framework, and the resource allocation model (i.e. workload- and location-aware resource allocation, WLARA) more specifically, is compared with the related allocation models that 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 [15], the greedy, random, and round robin (RR) models have been used by Nimbus and Eucalyptus to assign VMs to a physical machine in a data center. In addition, the manual zone selection is a load placement scheme that is similar to those used by current providers, such as Amazon EC2. So, with the manual zone selection, consumers manually select a data center at which to place their virtual machines. The manual zone selection assumes that a human is aware and selects the closest data center, so the manual zone selection used in this simulation places all consumers' VM at a data center deployed in the same location (i.e., zone) as each consumer. Manual zone selection is classified into an ideal case and non-ideal case for the simulation. With the ideal case (i.e., ideal manual selection, IM), a consumer's VM is placed at the closest data center. With the nonideal case (i.e., nonideal manual selection, NIM), a consumer's VM is placed at a close data center by applying

random values that follow a normal distribution in order to simulate situations where users sometimes make a mistake in selecting the closest data center.

5.3 Empirical results and observations

Figures 6 and 7 show the overall trends of the simulation with a high arrival rate of consumers. Figure 6 shows the overall workload distribution to all data centers of a cloud provider by time (0 to 223 time slots substituted with 0 to 2,230 min). Figure 7 shows the overall service response times, including the agreed-upon response time in the SLA and measured network and workload delays for each consumer. The measured response time is the summation of the measured network and workload delays. If the measured response time exceeds the response time agreed to in the SLA, the SLA is violated. In Fig. 7, the agreed-upon response times are spread like a cluster in a strip shape. Figures 6 and 7 show various trends for each resource allocation scheme as follows:

(1) Greedy allocation: Fig. 6(a) shows the overall workload distribution when applying the greedy allocation scheme. The workload distribution to data centers was similar to a staircase. For instance, consumer requests from 1st to 3rd TS were allocated to D0, from 4th to 6th TS were allocated to D1, from 6th to 10th TS were allocated to D2, from 10th to 13th TS were allocated to D3, and so on. This is because a data center is selected and then requests are allocated to the selected data center until the data center faces the workload limit (which is 2.0) with greedy allocation. After the data center reaches the workload limit, the coordinator selects the next data center. In Fig. 7(a), the measured response times of most consumers' requests (which consist of network and workload delays) are relatively longer than the agreed-upon response times. Therefore, we can expect many SLA violations with greedy allocation.
(2) Random allocation: Fig. 6(b) shows the overall workload distribution when applying the random (uniformly) allocation scheme. Whereas the greedy allocation showed a workload distribution similar to a staircase, random allocation showed uniformly distributed workloads to data centers. However, similar to the greedy allocation, most of the measured response times were above the strip of SLAs, as shown in Fig. 7(b).
(3) Round robin allocation (RR): Figs. 6(d) and 7(d) shows the overall workload distribution and overall service response times when applying the round robin allocation scheme, respectively. The results of the round robin allocation are similar to the results of the random allocation. Considering that round robin allocation distributes workloads uniformly, it is reasonable that the round robin and random allocations show similar results.
(4) WLARA: Fig. 6(c) shows the overall workload distribution when applying WLARA. The overall workload distribution is similar to the random and round robin allocations because WLARA considers the workload distribution in allocating resources. However, in Fig. 7(c), WLARA shows quite different results regarding the response time as compared with the other allocation schemes. In Fig. 7(c), most of the measured response times are below or middle of the SLA strip. The response times below the SLA strip indicate that the services did not

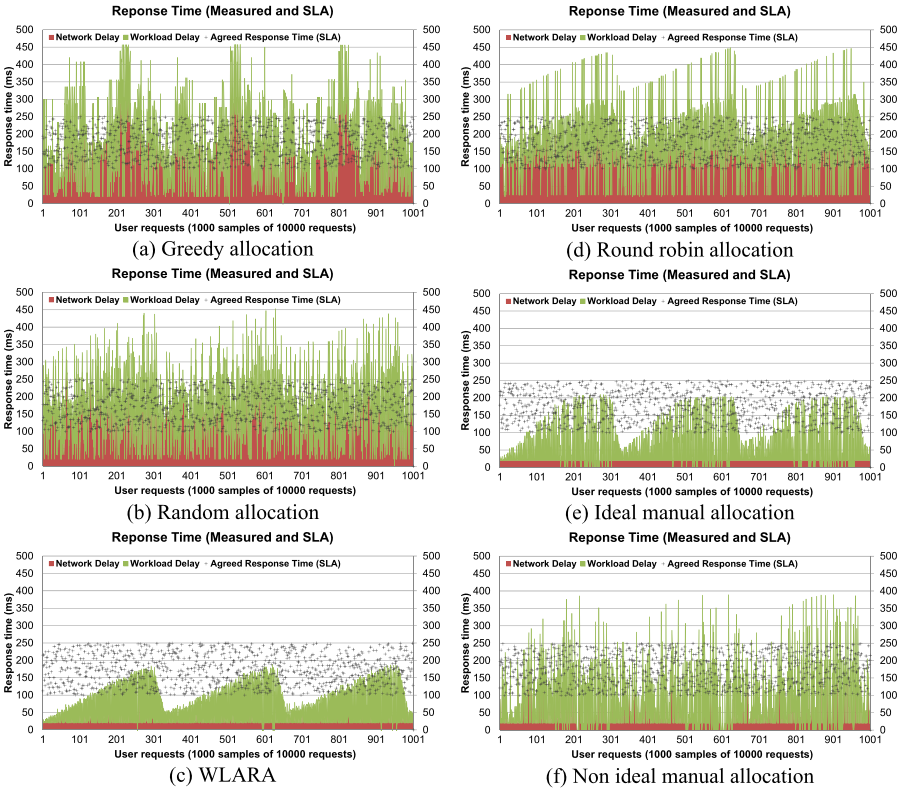


Fig. 7 Overall service response times

violate each SLA, and response times placed in the middle of the SLA strip indicate that the services are manageable. The characteristics of WLARA are shown in Fig. 7(c). Because WLARA is aware of the network delay due to locational differences, the network delay is low. WLARA is also aware of the data center workloads and manages resource allocations, so the workload delay is lower as compared with other schemes.

(5) Ideal manual allocation (IM): In Fig. 7(e), all services delivered to consumers had a 20 ms network delay, which is the lowest network delay in this experiment, as a data center with the same location as a consumer is selected in the IM allocation (e.g., when a consumer is in zone 4, the data center in zone 4 is selected to allocate resources). However, the IM allocation considers only the data center location, which causes problems in the load distribution. As shown in Fig. 6(e), the peak workloads at 30 TS, 90 TS, and 150 TS were around 11, whereas the peak workloads of WLARA shown in Fig. 6(c) were around 14. This is because there were many resource allocation failures with the IM allocation. If too many requests are allocated to one data center, the data center will not be available after the resource capacity runs out. Those requests become allocation failures, leading to diminished profits. In Fig. 6(e), the trend of the measured response

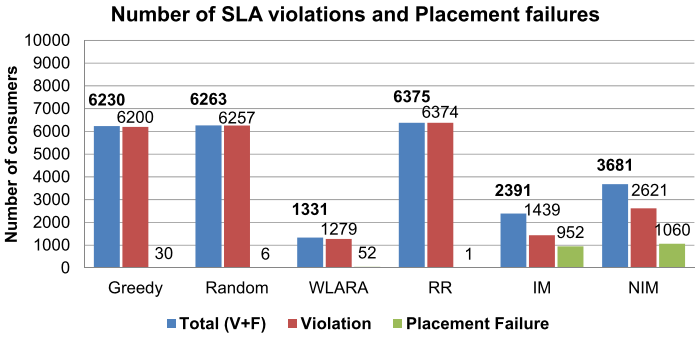


Fig. 8 Total number of SLA violations and placement failures

times is similar to the trend in WLARA, but the IM allocation provided longer response times.

(6) Nonideal manual allocation (NIM): The results of the NIM allocation in Figs. 6(f) and 7(f) show similar trends as the IM allocation because the schemes are similar. However, because we assumed that consumers choose a close data center and it is hard for them to carefully consider network delays, the network delay was higher than the IM allocation. Thus, the measured response times with NIM allocation were longer than those for IM allocation.

Figure 8 shows the comprehensive performance of WLARA and the other schemes in terms of the number of SLA violations and placement failures. Consumers have different response time thresholds according to the outcomes of the negotiated SLA. At the final stage of an simulation, the response time of each consumer's VM is aggregated to check whether it violated the given response time threshold. In Fig. 8, when the provider uses WLARA, the least number of SLA violations (1,297) is guaranteed, whereas the greedy, random, RR, NIM, and IM schemes caused 6,200, 6,257, 6,374, 1,439, and 2,621 SLA violations, respectively. When the provider uses WLARA, there is a low number of placement failures (52), whereas NIM and IM caused 952 and 1,060 placement failures, respectively. Although WLARA shows a higher number of placement failures (52) than the greedy (30), random (6), and RR (1) schemes, there is too much of a performance difference in SLA violations between WLARA and the other schemes, because WLARA considers both workload and response time (including network delay) in the utility function (4). Hence, WLARA can allocate a consumer's request to a PM at a data center that has a lower workload and is close in location in order to guarantee the response time threshold specified in the SLA.

Figures 9 and 10 are presented to analyze the logical reasoning of the performances shown in Fig. 8. Figure 9 shows the average network delay (i.e., geographical distance) between users and the allocated data centers. When the provider uses IM allocation, the average network delay is 20 ms because the IM model always places the VM at the data center with the same location as each user. However, considering the standard deviation of the load distribution to data centers shown in Fig. 10, both NIM and IM perform poorly, causing low performance in terms of SLA violations and the number of allocation failures. When a user request is not acceptable to a selected

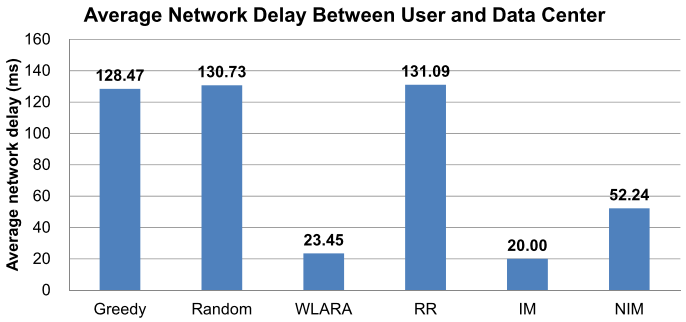


Fig. 9 Average network delay between user and data center (locational difference)

data center due to the capacity limit, the data center may deny allocation. The average network delay with WLARA is 23.45 ms. Although WLARA sometimes does not allocate a user's VM to a data center, which is in the exactly same zone with the user, the model shows a smaller number of SLA violations because it simultaneously considers workload and response time.

Figure 10 shows a detailed graph of Fig. 6 regarding the data center load distribution. Figures 10(a), (b), (c), and (d) show the data center workloads at 80 TS (period when the number of requests is increasing), 90 TS (period when the load is at its peak), and 100 TS (period when the number of requests is decreasing), and the average workload during 0 TS to 223 TS, respectively. Each allocation model shows different trends for the data center load distribution. To represent the trends, we indicated the SD (standard deviation) for each model. In Fig. 10(a), the SD of the greedy, random, WLARA, RR, IM, and NIM schemes are 0.65, 0.11, 0.20, 0.06, 0.61, and 0.65, respectively. Therefore, WLARA (SD: 0.20) shows less bias than the greedy, IM, and NIM schemes (SD: 0.65, 0.61, and 0.65, respectively) and smoothly distributes the loads at the data centers. Similar trends can be observed from the other graphs in Figs. 10(b), (c), and (d). Whereas schemes focused only on load distribution, such as the random and RR allocation models, show a uniform load distribution and small standard deviation, the schemes perform poorly in terms of the average network delay. These allocation models do not consider both workload and location, so they frequently violate SLAs.

Figure 11 shows a profit comparison among the various schemes, where profit is calculated using Eq. (8). An SLA violation incurs a penalty and the penalty follows Eq. (9), where R_{Penalty} is the penalty rate. In this experiment, we applied 0.3 as the penalty rate (i.e., 30 % of the agreed-upon price).

$$\text{Profit} = \begin{cases} 0, & \text{allocation failure,} \\ \text{Price}_{\text{agreed}} \cdot \text{Hours}_{\text{usage}}, & \text{without SLA violation,} \\ (\text{Price}_{\text{agreed}} \cdot \text{Hours}_{\text{usage}}) - \text{Penalty}, & \text{with SLA violation,} \end{cases} \tag{8}$$

$$\text{Penalty} = (\text{Price}_{\text{agreed}} \cdot \text{Hours}_{\text{usage}}) \cdot R_{\text{Penalty}}. \tag{9}$$

In Fig. 11, the profit to the provider is \$21,577, \$21,396, \$25,460, \$21,252, \$22,962, and \$21,532 with the greedy, random, WLARA, RR, IM, and NIM schemes, respectively. WLARA shows the best performance (\$25,460) in terms of profit. A

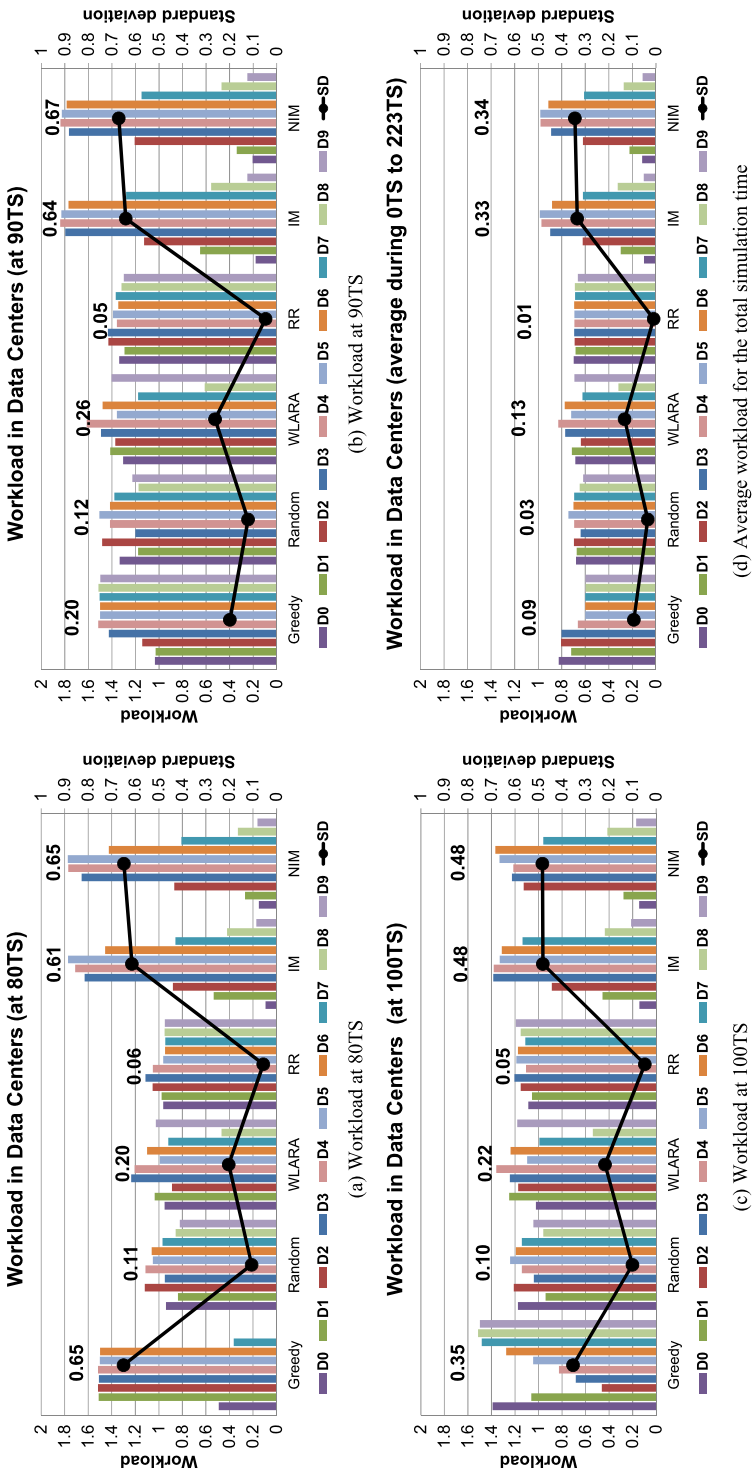


Fig. 10 Workload distribution (at 80 TS, 90 TS, 100 TS, and average of total simulation time)

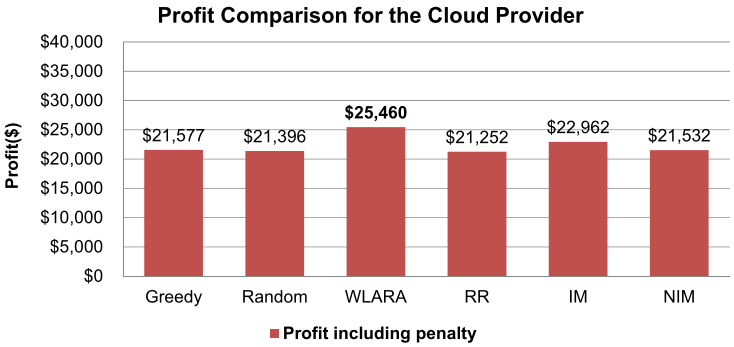


Fig. 11 Profit comparison of WLARA with related schemes

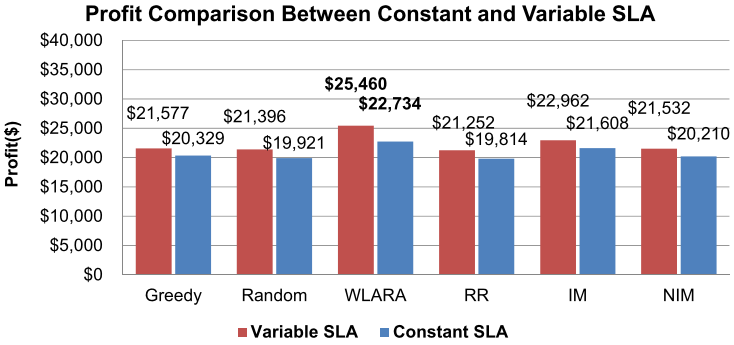


Fig. 12 Profit comparison between constant and variable SLA

cloud provider that adopted WLARA made 15.25 %, 15.96 %, 16.52 %, 9.81 %, and 15.42 % more profits than a cloud provider that adopted the greedy, random, RR, IM, or NIM scheme, respectively. This is because WLARA gives fewer SLA violations, leading to fewer penalties.

In Fig. 12, we show a comparison of profits between constant and variable SLA using the automated negotiation mechanism. For all allocation schemes, the automated negotiation mechanism shows higher profits (average $22,363) than the constant SLA (average $20,769). The SLA negotiation mechanism worked properly, as the response time threshold was low (fast) when the price was low (high). Also, the response times and prices for SLAs were well distributed ($100 <$ Agreed-Upon Response Time < 250; $\$0.1 <$ Agreed-Upon Price $< \$1.0$) according to the given preference range in Table 1.

Figures 13 and 14 show the overall trends of the simulation with the low consumers' arrival rate situation. Figure 13 shows the overall workload distribution for all data centers in a cloud provider by time (0 to 301 Time Slot substituted for 0 to 3,010 min). Figure 14 shows the overall service response times, including the agreed-upon response time in SLA and the measured network and workload delays for each consumer. Figures 13 and 14 show trends that are similar to the results in Figs. 6 and 7 for the high consumers' arrival rate situation. The major difference between

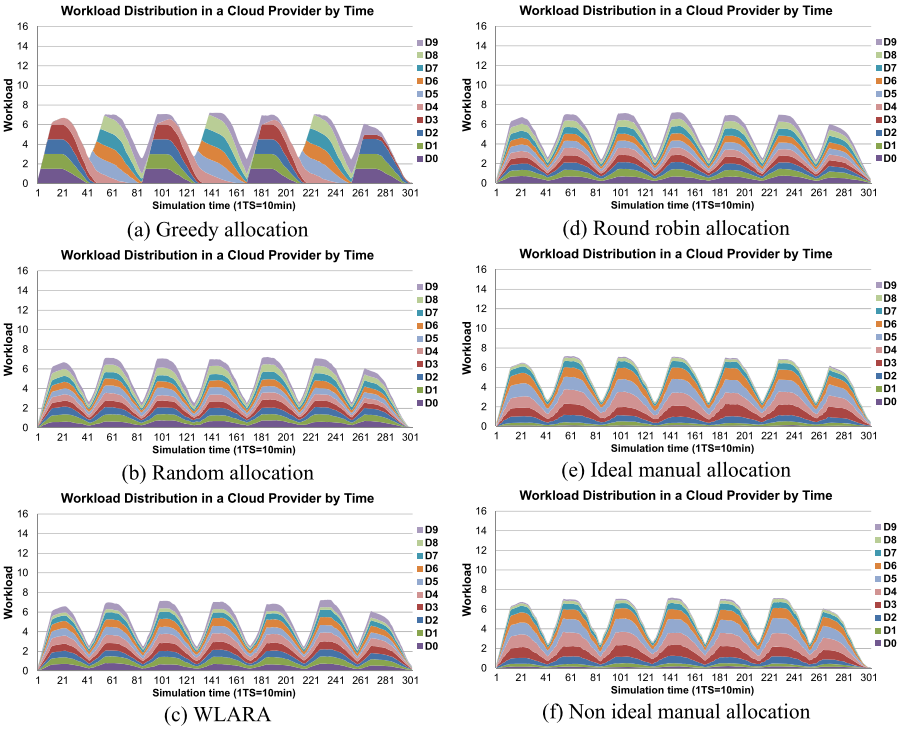


Fig. 13 Overall workload distributions

the low and high arrival rates is that the peak loads in the low arrival rate situation are lower (total load is around 6.3) than the high arrival rate cases (total load is around 14). This is because we fixed 10,000 consumers and changed the arrival rate of consumers. Thus, the low arrival rate situation can simulate low load situations for the allocation schemes.

Figure 15 shows the comprehensive performance of WLARA and the other schemes in terms of the number of SLA violations and placement failures. Consumers have different response time thresholds according to the negotiated SLA outcomes. At the final stage of the experiment, the response time of each consumer's VM is aggregated to check whether it violates the given response time threshold. In Fig. 15, the WLARA model guarantees the least number of SLA violations (1,297), whereas the greedy, random, RR, NIM, and IM models caused 6,200, 6,257, 6,374, 1,439, and 2,621 SLA violations, respectively. When the provider uses the proposed WLARA model, there is a low number of placement failures (52), whereas NIM and IM caused 952 and 1,060 cases of placement failure, respectively. Although WLARA shows a higher number of placement failures (52) than the greedy (30), random (6), and RR (1) models, there is too great of a performance difference in SLA violations between WLARA and the other schemes.

Figure 16 shows a comparison of profits among the various schemes, where profit is calculated using Eq. (8). An SLA violation incurs a penalty and the penalty follows Eq. (9). In Fig. 16, the profit to the provider is \$21,953, \$22,574, \$27,240, \$22,533,

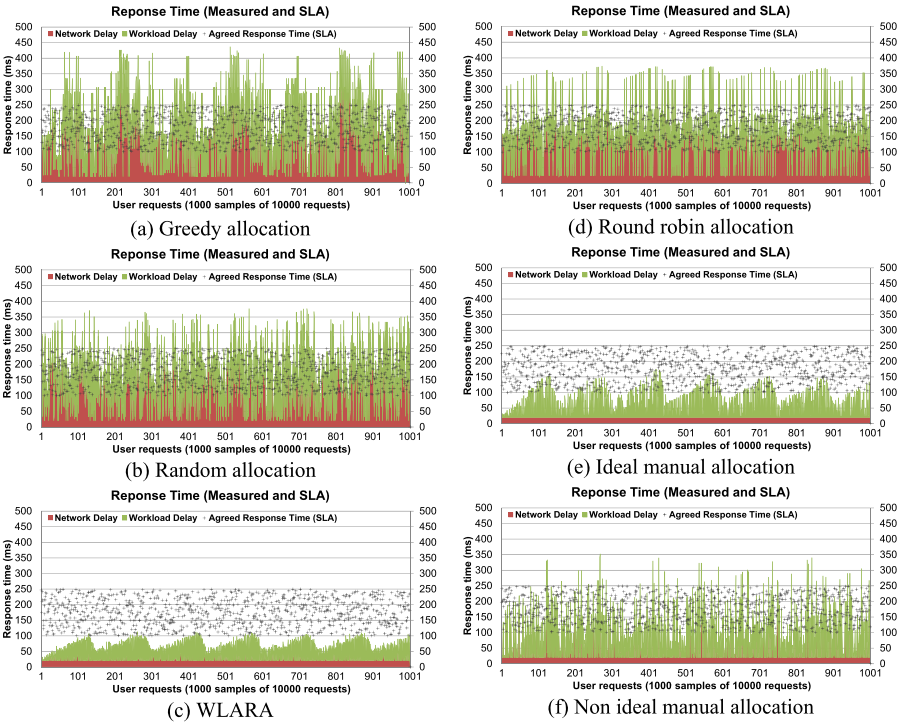


Fig. 14 Overall service response times

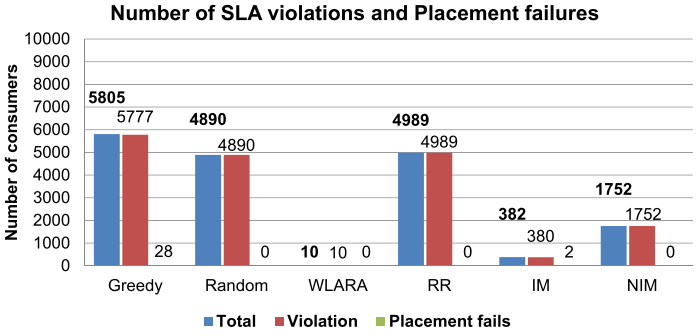


Fig. 15 Total number of SLA violations and placement failures

$26,771, and $25,317 for the greedy, random, WLARA, RR, IM, and NIM models, respectively. WLARA shows the best performance ($27,240) in terms of profit. A cloud provider adopting WLARA made 19.41 %, 17.13 %, 17.28 %, 1.72 %, and 7.06 % more profits than a cloud provider adopting the greedy, random, RR, IM, or NIM model, respectively. This is because WLARA gives fewer SLA violations and leads to fewer penalties.

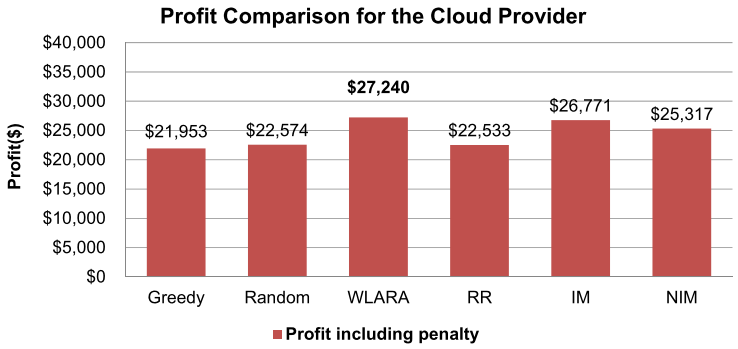


Fig. 16 Profit comparison of WLARA with related schemes

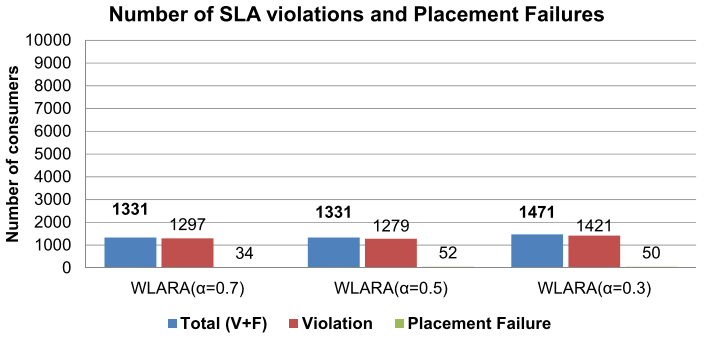


Fig. 17 Total number of SLA violations and placement failures

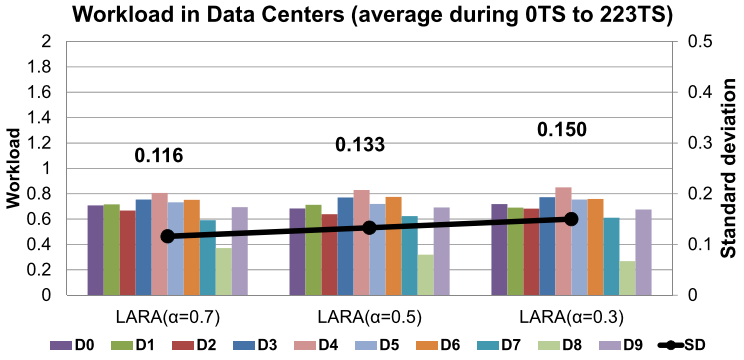


Fig. 18 Workload distribution sample (average of total simulation time)

Finally, Figs. 17, 18, and 19 represent the performance of WLARA with different experimental settings (the weights α and β in (4)). WLARA uses utility function (4), which allows a provider to adjust each preference weight (i.e., placing greater emphasis on workload or location). Hence, the results differ slightly by weight, as shown in

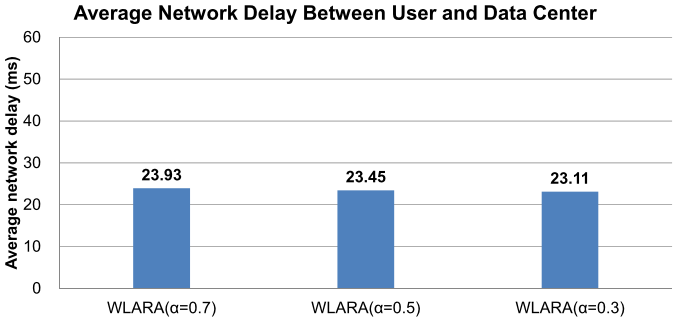


Fig. 19 Average network delay between user and data center (locational difference)

Fig. 17. When the provider uses load-preferred allocation (i.e., load weight α is 0.7, and location weight β is 0.3), the load distribution shows the best results, as in Fig. 18. However, Fig. 19 shows that WLARA ($\alpha = 0.7$) gives a relatively longer distance as compared with other preferences ($\alpha = 0.5$ and $\alpha = 0.3$).

6 Conclusion and future work

In this paper, we propose an SLA-based cloud computing framework to facilitate resource allocation (i.e., virtual machine placement) that takes into account the workload and geographical location of distributed data centers. In this framework, a cloud provider has several data centers that are geographically deployed and we implement a testbed of the proposed cloud framework to verify the usefulness of the proposed framework through simulations. Whereas [4–7, 10, 11] focused on dynamic resource allocation in a data center without considering a global network delay, the focus in this paper on dynamic resource allocation is selecting a data center among globally distributed data centers.

The main functionalities of the proposed cloud computing framework include the SLA negotiation and resource allocation mechanisms. Hence, by using the proposed framework, we expect that cloud computing providers can distribute the resource load (i.e., resource demand) from the locational perspective. In summary, the contributions of this paper are as follows:

(1) A cloud computing framework that includes an automated SLA negotiation mechanism and a workload- and location-aware resource allocation scheme (WLARA) has been designed. Using the automated SLA negotiation mechanism, the service response time in an SLA (i.e., variable SLA) can be controlled to facilitate load distribution in data centers around world through a pricing strategy. In addition, a workload- and location-aware resource allocation scheme (WLARA) provides a utility-based resource allocation to distributed data centers. (2) An agent-based cloud testbed of the proposed framework has been implemented. Using the testbed, experiments were conducted to compare the proposed schemes with related approaches. Empirical results show that the proposed WLARA performs better than other related approaches surveyed in [15] (e.g., random, round robin, greedy, and manual allocation) in terms of SLA violations and the provider's profits. We also show that using

the automated SLA negotiation mechanism supports providers in earning higher profits because the automated SLA negotiation mechanism provides a variable SLA (i.e., a variable response time and price).

In future research, the proposed framework can be extended by taking into account the VM migration functionality and the energy efficiency of data centers so that the proposed framework can adaptively allocate resources according to the effect of VM migrations and reduce the operational cost using an energy efficient resource allocation. In addition, we need to research and classify the negotiable cloud SLA issues to include in the framework.

Acknowledgements 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.

References

1. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616
2. Amazon EC2 (2012) http://aws.amazon.com/ec2. Accessed 1 July 2012
3. Armbrust M, Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zahara M (2010) A view of cloud computing. Commun ACM 53(4):50–58
4. Minarolli D, Freisleben B (2011) Utility-based resource allocation for virtual machines in cloud computing. In: 2011 IEEE symposium on computers and communications (ISCC), pp 410–417. doi:10.1109/ISCC.2011.5983872
5. Walsh WE, Tesauro G, Kephart JO, Das R (2004) Utility functions in autonomic systems. In: First international conference on autonomic computing (ICAC'04), pp 70–77
6. Menasce DA, Bennani MN (2006) Autonomic virtualized environments. In: International conference on autonomic and autonomous systems (ICAS'06), pp 28–38
7. Chase JS, Anderson DC, Thakar PN, Vahdat AM, Doyle RP (2001) Managing energy and server resources in hosting centers. In: Eighteenth ACM symposium on operating systems principles (SOSP'01), pp 103–116
8. Van Nguyen H, Dang Tran F, Menaud J-M (2009) Autonomic virtual resource management for service hosting platforms. In: 2009 ICSE workshop on software engineering challenges of cloud computing (CLOUD'09), pp 1–8
9. Bennani MN, Menasce DA (2005) Resource allocation for autonomic data centers using analytic performance models. In: Second international conference on autonomic computing ICAC 2005, pp 229–240. doi:10.1109/ICAC.2005.50
10. Shi W, Hong B (2011) Towards profitable virtual machine placement in the data center. In: 2011 fourth IEEE international conference on utility and cloud computing (UCC), pp 138–145. doi:10.1109/UCC.2011.28
11. Breitgand D, Epstein A (2011) Sla-aware placement of multivirtual machine elastic services in compute clouds. In: 12th IFIP/IEEE international symposium on integrated network management (IM11), Dublin, Ireland
12. Chaisiri S, Lee B-S, Niyato D (2009) Optimal virtual machine placement across multiple cloud providers. In: Services computing conference, APSCC 2009. IEEE Asia-Pacific, pp 103–110. doi:10.1109/APSCC.2009.5394134
13. Frincu ME, Craciun C (2011) Multi-objective meta-heuristics for scheduling applications with high availability requirements and cost constraints in multi-cloud environments. In: 2011 fourth IEEE international conference on utility and cloud computing (UCC), pp 267–274. doi:10.1109/UCC.2011.43
14. Tordsson J, Montero RS, Moreno-Vozmediano R, Llorente IM (2012) Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Gener Comput Syst 28(2):358–367. doi:10.1016/j.future.2011.07.003

15. Sotomayor B, Montero RS, Llorente IM, Foster I (2009) Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput 13(5):14–22
16. Buyya R, Beloglazov A, Abawajy J (2010) Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. In: The 2010 international conference on parallel and distributed processing techniques and applications (PDPTA2010), pp 6–20
17. Le K, Zhang J, Meng J, Bianchini R, Nguyen TD, Jaluria Y (2011) Reducing electricity cost through virtual machine placement in high performance computing clouds. In: 2011 super computing (SC11), Washington, USA
18. Sim KM (2010) Grid resource negotiation: survey and new directions. IEEE Trans Syst Man Cybern, Part C, Appl Rev 40(3):245–257
19. Paschke A, Dietrich J, Kuhla K (2005) A logic based SLA management framework. In: Semantic web and policy workshop (SWPW), 4th semantic web conference (ISWC 2005), Galway, Ireland
20. Netto MA, Bubendorfer K, Buyya R (2007) SLA-based advance reservations with flexible and adaptive time QoS parameters. In: 5th international conference on service-oriented computing, Vienna, Austria, Sep 2007
21. Brandic I, Music D, Dustdar S (2009) Service mediation and negotiation bootstrapping as first achievements towards self-adaptable grid and cloud services. In: Grids meet autonomic computing workshop (GMAC 2009), In conjunction with the 6th international conference on autonomic computing and communications, Spain, June 2009
22. Foster I, Kesselman C, Lee C, Lindell B, Nahrstedt K, Roy A (1999) A distributed resource management architecture that supports advance reservations and co-allocation. In: 7th international workshop on quality of service (IWQoS'99), London, UK. IEEE Comput Soc, Los Alamitos
23. Son S, Jung G, Jun SC (2012) A SLA-based cloud computing framework: workload and location aware resource allocation to distributed data centers in a cloud. In: The 2012 international conference on parallel and distributed processing techniques and applications (PDPTA2012), Las Vegas, USA (to be appearing)
24. Sim KM (2005) Equilibria, prudent compromises, and the "Waiting" game. IEEE Trans Syst Man Cybern, Part B, Cybern 35(4):712–724
25. Rubinstein A (1982) Perfect equilibrium in a bargaining model. Econometrica 50(1):97–109
26. Son S, Sim KM (2012) A price and time slot negotiation mechanism for cloud service reservations. IEEE Trans Syst Man Cybern, Part B, Cybern 42(3):713–728. doi:10.1109/TSMCB.2011.2174355
27. Xen Hypervisor (2012) http://www.xen.org. Accessed 10 Dec 2012
28. JADE (2012) http://jade.tilab.com. Accessed 1 July 2012
29. FIPA (2012) http://www.fipa.org. Accessed 1 July 2012
30. Verizon IP Latency Statistics (2012) http://verizonbusiness.com/about/network/latency. Accessed 1 July 2012