Guaranted QOS with Profit Maximization Scheme in CloudComputing

Anusha K S
Dept. of CSE
GMIT, Bharathinagar

Archana H C
Dept. of CSE
GMIT, Bharathinagar

Jeevitha H M
Dept. of CSE
GMIT, Bharathinagar

Sindhu P
Dept. of CSE
GMIT, Bharathinagar

Nandish Kumar D
Assoc. Prof., & Head, Dept. of CSE
GMIT, Bharathinagar

Abstract—As an effective and efficient way to provide computing resources and services to customers on demand, cloud computing has become more and more popular. From cloud service providers' perspective, profit is one of the most important considerations, and it is mainly determined by the configuration of a cloud service platform under given market demand. However, a single long-term renting scheme is usually adopted to configure a cloud platform, which cannot guarantee the service quality but leads to serious resource waste. In this paper, a double resource renting scheme is designed firstly in which short-term renting and long-term renting are combined aiming at the existing issues. This double renting scheme can effectively guarantee the quality of service of all requests and reduce the resource waste greatly. Secondly, a service system is considered as an M/M/m+D queuing model and the performance indicators that affect the profit of our double renting scheme are analyzed, e.g., the average charge, the ratio of requests that need temporary servers, and so forth. Thirdly, a profit maximization problem is formulated for the double renting scheme and the optimized configuration of a cloud platform is obtained by solving the profit maximization problem. Finally, a series of calculations are conducted to compare the profit of our proposed scheme with that of the single renting scheme. The results show that our scheme can not only guarantee the service quality of all requests, but also obtain more profit than the latter.

Index Terms—Cloud computing, guaranteed service quality, multiserver system, profit maximization, queuing model, service-level agreement, waiting time.

1 INTRODUCTION

As an effective and efficient way to consolidate computing resources and computing services, clouding computing has become more and more popular [1]. Cloud computing centralizes management of resources and services, and delivers hosted services over the Internet. The hardware, software, databases, information, and all resources are concentrated and provided to consumers on-demand [2]. Cloud computing turns information technology into ordinary commodities and utilities by the the pay-per-use pricing model [3, 4, 5]. In a cloud computing environment, there are always three tiers, i.e., infrastructure providers, services providers, and customers (see Fig. 1 and its elaboration in Section 3.1). An infrastructure provider maintains the basic hardware and software facilities. A service provider rents resources from the infrastructure providers and provides services to customers. A customer submits its request to a service provider and pays for it based on the amount and the quality of the provided service [6]. In this paper, we aim at researching the multiserver configuration of a service provider such that its profit is maximized.

Like all business, the profit of a service provider in cloud computing is related to two parts, which are the cost and the revenue. For a service provider, the cost is the renting cost paid to the infrastructure providers plus the electricity cost caused by energy consumption, and the revenue is the service charge to customers. In general, a service provider rents a certain number of servers from the infrastructure providers and builds different multiserver systems for different application domains. Each multiserver system is to execute a special type of service requests and applications. Hence, the renting cost is proportional to the number of servers in a multiserver system [2]. The power consumption of a multiserver system is linearly proportional to the number of servers and the server utilization, and to the square of execution speed [7, 8]. The revenue of a service provider is related to the amount of service and the quality of service. To summarize, the profit of a service provider is mainly determined by the configuration of its service platform.

To configure a cloud service platform, a service provider usually adopts a single renting scheme. That's to say, the servers in the service system are all long-term rented. Because of the limited number of servers, some of the incoming service requests cannot be processed immediately. So they are first inserted into a queue until they can handled by any available server. However, the waiting time of the service requests cannot be too long. In order to satisfy quality-of-service requirements, the waiting time of each incoming service request should be limited within a certain range, which is determined by a service-level agreement (SLA). If the quality of service is guaranteed, the service is fully charged, otherwise, the service provider serves the request for free as a penalty of low quality. To obtain higher revenue, a service provider should rent more servers from the infrastructure providers or scale up the server execution speed to ensure that more service requests are processed with high service quality. However, doing this would lead to sharp increase of the renting cost or the electricity cost. Such increased cost may counterweight the gain from penalty reduction. In conclusion, the single renting scheme is not a good scheme for service providers. In this paper, we propose a novel renting scheme for service providers, which not only can satisfy quality-of-service requirements, but also can obtain more profit. Our contributions in this paper can be summarized as follows.

- A novel double renting scheme is proposed for service providers. It combines long-term renting with short-term renting, which can not only satisfy quality-of-service requirements under the varying system workload, but also reduce the resource waste greatly.
- A multiserver system adopted in our paper is modeled as an $M/M/m+D$ queuing model and the performance indicators are analyzed such as the average service charge, the ratio of requests that need short-term servers, and so forth.
- The optimal configuration problem of service providers for profit maximization is formulated and two kinds of optimal solutions, i.e., the ideal solutions and the actual solutions, are obtained respectively.
- A series of comparisons are given to verify the performance of our scheme. The results show that the proposed Double-Quality-Guaranteed (DQG) renting scheme can achieve more profit than the compared Single-Quality-Unguaranteed (SQU) renting scheme in the premise of guaranteeing the service quality completely.

The rest of the paper is organized as follows. Section 2 reviews the related work on profit aware problem in cloud computing. Section 3 presents the used models, including the three-tier cloud computing model, the multiserver system model, the revenue and cost models. Section 4 proposes our DQG renting scheme and formulates the profit optimization problem. Section 5 introduces the methods of finding the optimal solutions for the profit optimization problem in two scenarios. Section 6 demonstrates the performance of the proposed scheme through comparison with the traditional SQU renting scheme. Finally, Section 7 concludes the work.

2 Related Work

In this section, we review recent works relevant to the profit of cloud service providers. Profit of service providers is related with many factors such as the price, the market demand, the system configuration, the customer satisfaction and so forth. Service providers naturally wish to set a higher price to get a higher profit margin; but doing so would decrease the customer satisfaction, which leads to a risk of discouraging demand in the future. Hence, selecting a reasonable pricing strategy is important for service providers.

The pricing strategies are divided into two categories, i.e., static pricing and dynamic pricing. Static pricing means that the price of a service request is fixed and known in advance, and it does not change with the conditions. With dynamic pricing a service provider delays the pricing decision until after the customer demand is revealed, so that the service provider can adjust prices accordingly [9]. Static pricing is the dominant strategy which is widely used in real world and in research [2, 10, 11]. Ghamkhari *et al.* [11] adopted a flat-rate pricing strategy and set a fixed price for all requests, but Odlyzko in [12] argued that the predominant flat-rate pricing encourages waste and is incompatible with service differentiation. Another kind of static pricing strategies are usage-based pricing. For example, the price of a service request is proportional to the service time and task execution requirement (measured by the number of instructions to be executed) in [10] and [2], respectively. Usage-based pricing reveals that one can use resources more efficiently [13, 14].

Dynamic pricing emerges as an attractive alternative to better cope with unpredictable customer demand [15]. Macías *et al.* [16] used a genetic algorithm to iteratively optimize the pricing policy. Amazon EC2 [17, 18] has introduced a ”spot pricing” feature, where the spot price for a virtual instance is dynamically updated to match supply and demand. However, consumers dislike prices to change, especially if they perceive the changes to be ”unfair” [19, 20]. After comparison, we select the usage-based pricing strategy in this paper since it agrees with the concept of cloud computing mostly.

The second factor affecting the profit of service providers is customer satisfaction which is determined by the quality of service and the charge. In order to improve the customer satisfaction level, there is a service-level agreement (SLA) between a service provider and the customers. The SLA adopts a price compensation mechanism for the customers with low service quality. The mechanism is to guarantee the service quality and the customer satisfaction so that more customers are attracted. In previous research, different SLAs are adopted. Ghamkhari *et al.* [11] adopted a stepwise charge function with two stages. If a service request is handled before its deadline, it is normally charged; but if a service request is not handled before its deadline, it is dropped and the provider pays for it due to penalty. In [2, 10, 21], charge is decreased continuously with the increasing waiting time until the charge is free. In this paper, we use a two-step charge function, where the service requests served with high quality are normally charged, otherwise, are served for free.

Since profit is an important concern to cloud service providers, many works have been done on how to boost their profit. A large body of works have recently focused on reducing the energy cost to increase profit of service providers [22, 23, 24, 25], and the idle server turning off strategy and *dynamic CPU clock frequency scaling* are adopted to reduce energy cost. However, only reducing energy cost cannot obtain profit maximization. Many researchers investigated the trade-off between minimizing cost and maximizing revenue to optimize profit. Both [11] and [26] adjusted the number of switched on servers periodically using different strategies and different profit maximization models were built to get the number of switched on servers. However, these works did not consider the cost of resource configuration.

Chiang and Ouyang [27] considered a cloud server system as an $M/M/R/K$ queuing system where all service requests that exceed its maximum capacity are rejected. A profit maximization function is defined to find an optimal combination of the server size R and the queue capacity K such that the profit is maximized. However, this strategy has further implications other than just losing the revenue from some services, because it also implies loss of reputation and therefore loss of future customers [3]. In [2], Cao *et al*. treated a cloud service platform as an *M/M/m model*, and the problem of optimal multiserver configuration for profit maximization was formulated and solved. This work is the most relevant work to ours, but it adopts a single renting scheme to configure a multiserver system, which cannot adapt to the varying market demand and leads to low service quality and great resource waste. To overcome this weakness, another resource management strategy is used in [28, 29, 30, 31], which is cloud federation. Using federation, different providers running services that have complementary resource requirements over time can mutually collaborate to share their respective resources in order to fulfill each one's demand [30]. However, providers should make an intelligent decision about utilization of the federation (either as a contributor or as a consumer of resources) depending on different conditions that they might face, which is a complicated problem.

In this paper, to overcome the shortcomings mentioned above, a double renting scheme is designed to configure a cloud service platform, which can guarantee the service quality of all requests and reduce the resource waste greatly. Moreover, a profit maximization problem is formulated and solved to get the optimal multiserver configuration which can product more profit than the optimal configuration in [2].

3 THE MODELS

In this section, we first describe the three-tier cloud computing structure. Then, we introduce the related models used in this paper, including a multiserver system model, a revenue model, and a cost model.

3.1 A Cloud System Model

The cloud structure (see Fig. 1) consists of three typical parties, i.e., infrastructure providers, service providers and customers. This three-tier structure is used commonly in existing literatures [2, 6, 10].

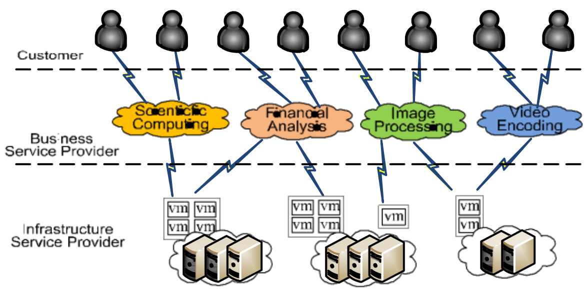


Fig. 1: The three-tier cloud structure.

In the three-tier structure, an infrastructure provider the basic hardware and software facilities. A service provider rents resources from infrastructure providers and prepares a set of services in the form of virtual machine (VM). Infrastructure providers provide two kinds of resource renting schemes, e.g., long-term renting and short-term renting. In general, the rental price of long-term renting is much cheaper than that of short-term renting. A customer submits a service request to a service provider which delivers services on demand. The customer receives the desired result from the service provider with certain service-level agreement, and pays for the service based on the amount of the service and the service quality. Service providers pay infrastructure providers for renting their physical resources, and charge customers for processing their service requests, which generates cost and revenue, respectively. The profit is generated from the gap between the revenue and the cost.

3.2 A Multiserver Model

In this paper, we consider the cloud service platform as a multiserver system with a service request queue. Fig. 2 gives the schematic diagram of cloud computing [32].

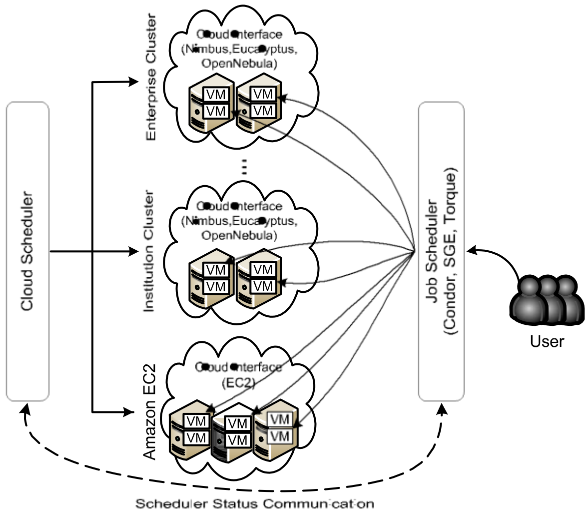


Fig. 2: The schematic diagram of cloud computing.

In an actual cloud computing platform such as Amazon EC2, IBM blue cloud, and private clouds, there are many work nodes managed by the cloud managers such as Eucalyptus, OpenNebula, and Nimbus. The clouds provide resources for jobs in the form of virtual machine (VM). In addition, the users submit their jobs to the cloud in which a job queuing system such as SGE, PBS, or Condor is used. All jobs are scheduled by the job scheduler and assigned to different VMs in a centralized way. Hence, we can consider it as a service request queue. For example, Condor is a specialized workload management system for compute-intensive jobs and it provides a job queueing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their jobs to Condor, and Condor places them into a queue, chooses when and where to run them based upon a policy [33, 34]. Hence, it is reasonable to abstract a cloud service platform as a multiserver model with a service request queue, and the model is widely adopted in existing literature [2, 11, 35, 36, 37].

In the three-tier structure, a cloud service provider serves customers' service requests by using a multiserver system

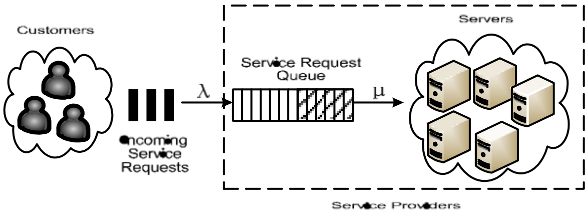


Fig. 3: The multiserver system model, where service requests are first placed in a queue before they are processed by any servers.

which is rented from an infrastructure provider. Assume that the multiserver system consists of m long-term rented identical servers, and it can be scaled up by temporarily renting short-term servers from infrastructure providers. The servers in the system have identical execution speed s (Unit: billion instructions per second). In this paper, a multiserver system excluding the short-term servers is modeled as an *M/M/m* queuing system as follows (see Fig. 3). There is a Poisson stream of service requests with arrival rate λ, i.e., the interarrival times are independent and identically distributed (i.i.d.) exponential random variables with mean $1/\lambda$. A multiserver system maintains a queue with infinite capacity. When the incoming service requests cannot be processed immediately after they arrive, they are firstly placed in the queue until they can be handled by any available server. The first-come-first-served (FCFS) queuing discipline is adopted. The task execution requirements (measured by the number of instructions) are independent and identically distributed exponential random variables r with mean $\bar{r}$ (Unit: billion instructions). Therefore, the execution times of tasks on the multiserver system are also i.i.d. exponential random variables $x = r/s$ with mean $\bar{x} = \bar{r}/s$ (Unit: second). The average service rate of each server is calculated as $\mu = 1/\bar{x} = s/\bar{r}$, and the system utilization is defined as $\rho = \lambda/m\mu = \lambda/m \times \bar{r}/s$.

Because the fixed computing capacity of the service system is limited, some requests would wait for a long time before they are served. According to the queuing theory, we have the following theorem about the waiting time in an *M/M/m* queuing system.

Theorem 3.1. The cumulative distribution function (cdf) of the waiting time W of a service request is

$$F_W(t) = 1 - \frac{\pi_m}{1-\rho} e^{-m\mu(1-\rho)t}, \quad (1)$$

where

$$\pi_m = \frac{(m\rho)^m}{m!} \left[\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!(1-\rho)} \right]^{-1}.$$

Proof 3.1. We have known that the probability distribution function (pdf) of the waiting time W of a service request is

$$f_W(t) = (1-P_q)u(t) + m\mu\pi_m e^{-(1-\rho)m\mu t},$$

where $P_q = \pi_m/(1-\rho)$ and $u(t)$ is a unit impulse function [2, 38]. Then, $F_W(t)$ can be obtained by straightforward calculation.

3.3 Revenue Modeling

The revenue model is determined by the pricing strategy and the server-level agreement (SLA). In this paper, the usage-based pricing strategy is adopted, since cloud computing provides services to customers and charges them on demand. The SLA is a negotiation between service providers and customers on the service quality and the price. Because of the limited servers, the service requests that cannot be handled immediately after entering the system must wait in the queue until any server is available. However, to satisfy the quality-of-service requirements, the waiting time of each service request should be limited within a certain range which is determined by the SLA. The SLA is widely used by many types of businesses, and it adopts a price compensation mechanism to guarantee service quality and customer satisfaction. For example, China Post gives a service time commitment for domestic express mails. It promises that if a domestic express mail does not arrive within a deadline, the mailing charge will be refunded. The SLA is also adopted by many real world cloud service providers such as Rackspace [39], Joyent [40], Microsoft Azure [41], and so on. Taking Joyent as an example, the customers order Smart Machines, Smart Appliances, and/or Virtual Machines from Joyent, and if the availability of a customer's services is less than 100%, Joyent will credit the customer 5% of the monthly fee for each 30 minutes of downtime up to 100% of the customer's monthly fee for the affected server. The only difference is that its performance metric is availability and ours is waiting time.

In this paper, the service level is reflected by the waiting time of requests. Hence, we define D as the maximum waiting time here that the service requests can tolerate, in other words, D is their deadline. The service charge of each task is related to the amount of a service and the service-level agreement. We define the service charge function for a service request with execution requirement r and waiting time W in Eq. (2),

$$R(r, W) = \begin{cases} ar, & 0 \le W \le D; \\ 0, & W > D. \end{cases} \quad (2)$$

where a is a constant, which indicates the price per one billion instructions (Unit: cents per one billion instructions). When a service request starts its execution before waiting a fixed time D (Unit: second), a service provider considers that the service request is processed with high quality-of-service and charges a customer ar. If the waiting time of a service request exceeds deadline D, a service provider must serve it for free. Similar revenue models have been used in many existing research such as [2, 11, 42].

According to Theorem 1, it is easy to know that the probability that the waiting time of a service request exceeds its deadline D is

$$P(W \ge D) = 1 - F_W(D) = \frac{\pi_m}{1-\rho} e^{-m\mu(1-\rho)D}. \quad (3)$$

3.4 Cost Modeling

The cost of a service provider consists of two major parts, i.e., the rental cost of physical resources and the utility cost of energy consumption. Many existing research such as [11, 43, 44] only consider the power consumption cost. As a major difference between their models and ours, the resource rental cost is considered in this paper as well, since it is a major part which affects the profit of service providers. A similar cost model is adopted in [2]. The resources can be rented in two ways, long-term renting and short-term renting, and the rental price of long-term renting is much cheaper than that of short-term renting. This is reasonable and common in the real life. In this paper, we assume that the long-term rental price of one server for unit of time is β (Unit: cents per second) and the short-term rental price of one server for unit of time is γ (Unit: cents per second), where $\beta < \gamma$.

The cost of energy consumption is determined by the electricity price and the amount of energy consumption. In this paper, we adopt the following dynamic power model, which is adopted in the literature such as [2, 7, 45, 46]:

$$P_d = N_{sw} C_L V^2 f, \tag{4}$$

where N_{sw} is the average gate switching factor at each clock cycle, C_L is the loading capacitance, V is the supply voltage, and f is the clock frequency [45]. In the ideal case, the relationship between the clock frequency f and the supply voltage V is $V \propto f^{\phi}$ for some constant $\phi > 0$ [46]. The server execution speed s is linearly proportional to the clock frequency f, namely, $s \propto f$. Hence, the power consumption is $P_d \propto N_{sw} C_L s^{2\phi+1}$. For ease of discussion, we assume that $P_d = b N_{sw} C_L s^{2\phi+1} = \xi s^{\alpha}$ where $\xi = b N_{sw} C_L$ and $\alpha = 2\phi + 1$. In this paper, we set $N_{sw} C_L = 7.0$, $b = 1.3456$ and $\phi = 0.5$. Hence, $\alpha = 2.0$ and $\xi = 9.4192$. The value of power consumption calculated by $P_d = \xi s^{\alpha}$ is close to the value of the Intel Pentium M processor [47]. It is reasonable that a server still consumes some amount of static power [8], denoted as P^* (Unit: Watt), when it is idle. For a busy server, the average amount of energy consumption per unit of time is $P = \xi s^{\alpha} + P^*$ (Unit: Watt). Assume that the price of energy is δ (Unit: cents per Watt).

4 A Quality-Guaranteed Scheme

The traditional single resource renting scheme cannot guarantee the quality of all requests but wastes a great amount of resources due to the uncertainty of system workload. To overcome the weakness, we propose a double renting scheme as follows, which not only can guarantee the quality of service completely but also can reduce the resource waste greatly.

4.1 The Proposed Scheme

In this section, we first propose the Double-Quality-Guaranteed (DQG) resource renting scheme which combines long-term renting with short-term renting. The main computing capacity is provided by the long-term rented servers due to their low price. The short-term rented servers provide the extra capacity in peak period. The detail of the scheme is shown in Algorithm 1.

The proposed DQG scheme adopts the traditional FCFS queueing discipline. For each service request entering the system, the system records its waiting time. The requests are assigned and executed on the long-term rented servers in the order of arrival times. Once the waiting time of a request reaches D, a temporary server is rented from infrastructure

Algorithm 1 Double-Quality-Guaranteed (DQG) Scheme

A multiserver system with m servers is running and waiting for the events as follows
A queue Q is initialized as empty
Event – A service request arrives
Search if any server is available
if true **then**
  Assign the service request to one available server
else
  Put it at the end of queue Q and record its waiting time
end if
End Event
Event – A server becomes idle
Search if the queue Q is empty
if true **then**
  Wait for a new service request
else
  Take the first service request from queue Q and assign it to the idle server
end if
End Event
Event – The deadline of a request is achieved
Rent a temporary server to execute the request and release the temporary server when the request is completed
End Event

providers to process the request. We consider the novel service model as an $M/M/m+D$ queuing model [48, 49, 50]. The $M/M/m+D$ model is a special $M/M/m$ queuing model with impatient customers. In an $M/M/m+D$ model, the requests are impatient and they have a maximal tolerable waiting time. If the waiting time exceeds the tolerable waiting time, they lose patience and leave the system. In our scheme, the impatient requests do not leave the system but are assigned to temporary rented servers.

Since the requests with waiting time D are all assigned to temporary servers, it is apparent that all service requests can guarantee their deadline and are charged based on the workload according to the SLA. Hence, the revenue of the service provider increases. However, the cost increases as well due to the temporarily rented servers. Moreover, the amount of cost spent in renting temporary servers is determined by the computing capacity of the long-term rented multiserver system. Since the revenue has been maximized using our scheme, minimizing the cost is the key issue for profit maximization. Next, the tradeoff between the long-term rental cost and the short-term rental cost is considered, and an optimal problem is formulated in the following to get the optimal long-term configuration such that the profit is maximized.

4.2 The Profit Optimization Problem

Assume that a cloud service platform consists of m long-term rented servers. It is known that part of requests need temporary servers to serve, so that their quality can be guaranteed. Denoted by $p_{ext}(D)$ the steady-state probability that a request is assigned to a temporary server, or put differently, $p_{ext}(D)$ is the long-run fraction of requests whose waiting times exceed the deadline D. $p_{ext}(D)$ is different from $F_W(D)$. In calculating $F_W(D)$, all service requests, whether exceed the deadline, will be waiting in the queue. However, in calculating $p_{ext}(D)$, the requests whose waiting times are equal to the deadline will be assigned to the temporary servers, which will reduce the waiting time of the following requests. In general, $p_{ext}(D)$ is much less than $F_W(D)$. Refer to [50], we can known that $p_{ext}(D)$ is:

$$p_{ext}(D) = \frac{(1-\rho)(1-F_W(D))}{1-\rho(1-F_W(D))}. \tag{5}$$

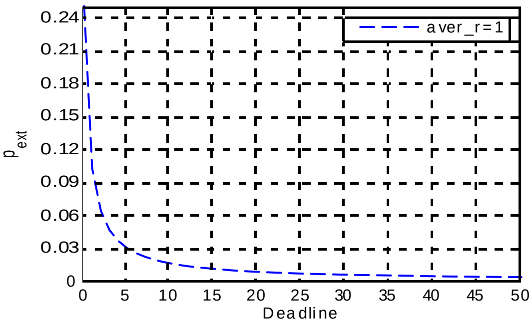


Fig. 4: The probability of waiting time exceeding D.

That is to say, there are about $\lambda p_{ext}(D)$ service requests in one unit of time which need short-term rented servers. Fig. 4 gives the probability versus different deadline where $\lambda = 5.99$, $\bar{r} = 1$, $m = 6$ and $s = 1$. Hence, the cost on short-term rented servers in one unit of time is calculated as:

$$C_{short} = \lambda p_{ext}(D) \frac{\bar{r}}{s}(\gamma + \delta P), \tag{6}$$

where $\frac{\bar{r}}{s}$ is the average execution time of each request.

Among the requests entering the service system, about $p_{ext}(D)$ percentage requests are not executed by the m long-term rented servers. Hence, the system utilization of the m servers is $\rho(1-p_{ext}(D))$. Since the power for speed s is ξs^α, the average amount of energy consumed by a long-term rented server in one unit of time is $P_{long} = \rho(1-p_{ext}(D))\xi s^\alpha + P^*$. Hence, the cost of the long-term rented servers in one unit of time is calculated as:

$$C_{long} = m(\beta + \delta P_{long}). \tag{7}$$

The following theorem gives the expected charge to a service request.

Theorem 4.1. The expected charge to a service request is $a\bar{r}$.

Proof 4.1. Because the waiting time W of each request is less than or equal to D, the expected charge to a service request with execution requirement r is ar according to the SLA. Since r is a random variable, ar is also random variable. It is known that r is an exponential random variable with mean $\bar{r}$, so its probability distribution function is $f_r(z) = \frac{1}{\bar{r}}e^{-z/\bar{r}}$. The expected charge to a service request is

$$\begin{aligned}
\int_0^\infty f_r(z)R(r,z)dz &= \int_0^\infty \frac{1}{\bar{r}}e^{-z/\bar{r}}azdz \\
&= \frac{a}{\bar{r}}\int_0^\infty e^{-z/\bar{r}}zdz = -a\int_0^\infty zde^{-z/\bar{r}} \\
&= -a\left[ze^{-z/\bar{r}}\Big|_0^\infty - \int_0^\infty e^{-z/\bar{r}}dz\right] \\
&= -a\left[ze^{-z/\bar{r}}\Big|_0^\infty + \bar{r}e^{-z/\bar{r}}\Big|_0^\infty\right] \\
&= a\bar{r}.
\end{aligned} \tag{8}$$

The theorem is proven.

The profit of a service provider in one unit of time is obtained as

$$Profit = Revenue - C_{long} - C_{short}, \tag{9}$$

where $Revenue = \lambda a\bar{r}$,

$$C_{long} = m(\beta + \delta(\rho(1-p_{ext}(D))\xi s^\alpha + P^*)),$$

and

$$C_{short} = \lambda p_{ext}(D)\frac{\bar{r}}{s}(\gamma + \delta(\xi s^\alpha + P^*)).$$

We aim to choose the optimal number of fixed servers m and the optimal execution speed s to maximize the profit:

$$\begin{aligned}
Profit(m,s) &= \lambda a\bar{r} - \lambda p_{ext}(D)\frac{\bar{r}}{s}(\gamma + \delta(\xi s^\alpha + P^*)) \\
&\quad - m(\beta + \delta(\rho(1-p_{ext}(D))\xi s^\alpha + P^*)).
\end{aligned} \tag{10}$$

Fig. 5 gives the graph of function $Profit(m,s)$ where $\lambda = 5.99$, $\bar{r} = 1$, $D = 5$, $a = 15$, $P^* = 3$, $\alpha = 2.0$, $\xi = 9.4192$, $\beta = 1.5$, $\gamma = 3$, and $\delta = 0.3$.

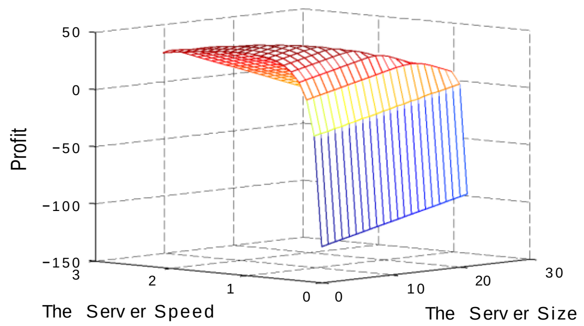


Fig. 5: The function $Profit(m,s)$.

From the figure, we can see that the profit of a service provider is varying with different server size and different execution speed. Therefore, we have the problem of selecting the optimal server size and/or server speed so that the profit is maximized. In the following section, the solutions to this problem are proposed.

5 Optimal Solution

In this section, we first develop an analytical method to solve our optimization problem. Using the analytical method, the ideal optimal solutions are obtained. Because the server size and the server speed are limited and discrete, we give an algorithmic method to get the actual solutions based on the ideal optimal ones.

5.1 An Analytical Method for Ideal Solutions

We firstly solve our optimization problem analytically, assuming that m and s are continuous variables. To this end, a closed-form expression of $p_{ext}(D)$ is needed. In this paper, we use the same closed-form expression as [2], which is $\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} \approx e^{m\rho}$. This expression is very accurate when m is not too small and ρ is not too large [2]. Since Stirling's approximation of $m!$ is $\sqrt{2\pi m}(\frac{m}{e})^m$, one closed-form expression of π_m is

$$\pi_m \approx \frac{1-\rho}{\sqrt{2\pi m}(1-\rho)(e\rho)^m+1},$$

and

$$p_{ext}(D) \approx \frac{(1-\rho)e^{-m\mu(1-\rho)D}}{1+\sqrt{2\pi m(1-\rho)}\left(\frac{e^{\rho}}{e\rho}\right)^m - \rho e^{-m\mu(1-\rho)D}}.$$

For convenience, we rewrite $p_{ext}(D) \approx \frac{(1-\rho)K_1}{K_2-\rho K_1}$, where $K_1 = e^{-m\mu(1-\rho)D}$, and $K_2 = 1 + \sqrt{2\pi m(1-\rho)}\Phi$, where $\Phi = (e^{\rho}/e\rho)^m$.

In the following, we solve our optimization problems based on above closed-form expression of $p_{ext}(D)$.

Given λ, $\bar{r}$, a, P^*, α, β, γ, δ, ξ, D, and s, our objective is to find m such that *Profit* is maximized. To maximize *Profit*, m must be found such that

$$\frac{\partial Profit}{\partial m} = -\frac{\partial C_{long}}{\partial m} - \frac{\partial C_{short}}{\partial m} = 0,$$

where

$$\frac{\partial C_{long}}{\partial m} = \beta + \delta P^* - \delta\lambda\bar{r}\xi s^{\alpha-1}\frac{\partial p_{ext}(D)}{\partial m},$$

and

$$\frac{\partial C_{short}}{\partial m} = \lambda(\gamma + \delta P^*)\frac{\bar{r}}{s}\frac{\partial p_{ext}(D)}{\partial m} + \lambda\bar{r}\delta\xi s^{\alpha-1}\frac{\partial p_{ext}(D)}{\partial m}.$$

Since

$$\ln\Phi = m\ln(e^{\rho}/e\rho) = m(\rho - \ln\rho - 1),$$

and

$$\frac{\partial\rho}{\partial m} = -\frac{\lambda\bar{r}}{m^2 s} = -\frac{\rho}{m},$$

we have

$$\frac{1}{\Phi}\frac{\partial\Phi}{\partial m} = (\rho - \ln\rho - 1) + m\left(1 - \frac{1}{\rho}\right)\frac{\partial\rho}{\partial m} = -\ln\rho,$$

and

$$\frac{\partial\Phi}{\partial m} = -\Phi\ln\rho.$$

Then, we get

$$\frac{\partial K_1}{\partial m} = -\mu D K_1,$$

and

$$\frac{\partial K_2}{\partial m} = \sqrt{2\pi m}\,\Phi\left(\frac{1+\rho}{2m} - \ln\rho(1-\rho)\right).$$

$$\begin{aligned}\frac{\partial p_{ext}(D)}{\partial m} = \frac{1}{(K_2-\rho K_1)^2}\Big[&\frac{\rho}{m}K_1(K_2-K_1)\\ &+ (\rho-1)\mu D K_1 K_2 - \frac{(1+\rho)K_1}{2m}(K_2-1)\\ &+ (1-\rho)K_1(\ln\rho)(K_2-1)\Big].\end{aligned}$$

We cannot get a closed-form solution to m, but we can get the numerical solution to m. Since $\partial Profit/\partial m$ is not an increasing or decreasing function of m, we need to find the decreasing region of m, and then use the standard bisection method. If there are more than one maximal values, they are compared and the maximum is selected. When using the bisection method to find the extreme point, the iteration accuracy is set as a unified value 10^{-10}.

In Fig. 6, we demonstrate the net profit in one unit of time as a function of m and λ where $s = 1$, $\bar{r} = 1$, and the other parameters are same as with those in Fig. 5. We notice that there is an optimal choice of m such that the net profit is maximized. Using the analytical method, the optimal value of m such that $\partial Profit/\partial m = 0$ is 4.8582, 5.8587, 6.8590, 7.8592 for λ = 4.99, 5.99, 6.99, 7.99, respectively. When the number of servers m is less than the optimal value, the service provider needs to rent more temporary servers to execute the requests whose waiting times are equal to the deadline; hence, the extra cost increases, even surpassing the gained revenue. As m increases, the waiting times are significantly reduced, but the cost on fixed servers increases greatly, which also surpasses the gained revenue too. Hence, there is an optimal choice of m which maximizes the profit.

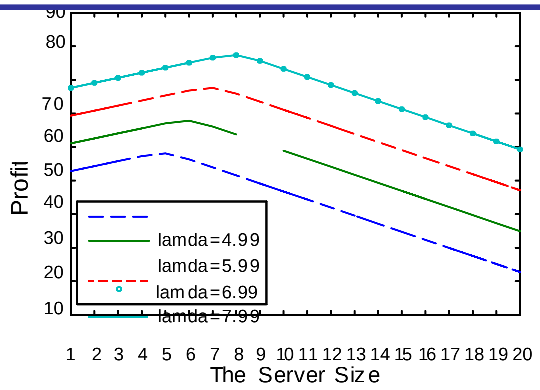
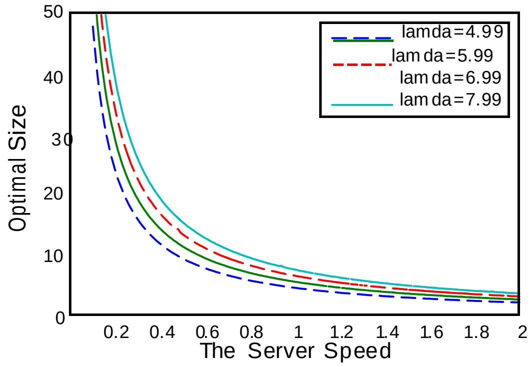
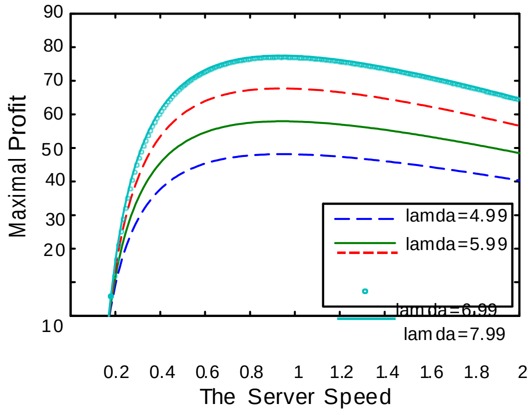


Fig. 6: Net profit versus m and λ.



(a) Optimal size versus s and λ.



(b) Maximal profit versus s and λ.

Fig. 7: Optimal size and maximal profit vs. s and λ.

In Fig. 7, we demonstrate the optimal size and maximal profit in one unit of time as a function of s and λ. It means, for each combination of s and λ, we find the optimal number of servers and the maximal profit. The parameters are same as those in Fig. 6. From the figures we can see that a higher speed leads to a less number of servers needed for each λ, and different λ values have different optimal combinations of speed and size. In addition, the greater the λ is, the more the maximal profit can be obtained.

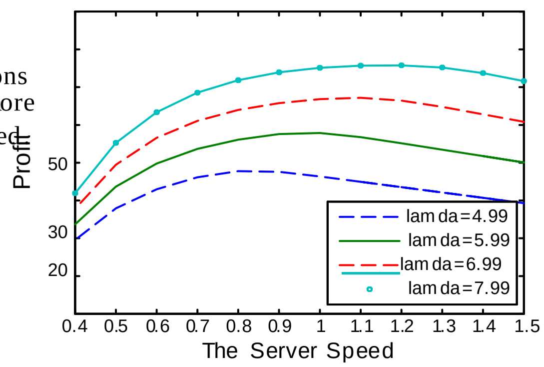


Fig. 8: Net profit versus s and λ.

5.1.2 *Optimal Speed*

Given λ, $\bar{r}$, a, P^*, α, β, γ, δ, ξ, D, and m, our objective is to find s such that *Profit* is maximized. To maximize *Profit*, s must be found such that

$$\frac{}{\partial s} = -\frac{}{\partial s} - \frac{}{\partial s} = 0,$$

where

$$\frac{\partial C_{long}}{\partial s} = \delta\xi\lambda\bar{r}s^{\alpha-2}\left[(\alpha-1)(1-p_{ext}(D)) - s\frac{\partial p_{ext}(D)}{\partial s}\right],$$

and

$$\frac{\partial C_{short}}{\partial s} = \frac{\bar{r}\lambda(\gamma+\delta P^*)}{s^2}\left(s\frac{\partial p_{ext}(D)}{\partial s} - p_{ext}(D)\right) + \lambda r\delta\xi s^{\alpha-2}\left[s\frac{\partial p_{ext}(D)}{\partial s} + (\alpha-1)p_{ext}(D)\right],$$

Since

$$\frac{\partial\rho}{\partial s} = -\frac{\lambda\bar{r}}{ms^2} = -\frac{\rho}{s},$$

and

$$\frac{1}{\Phi}\frac{\partial\Phi}{\partial s} = m\left(1-\frac{1}{\rho}\right)\frac{\partial\rho}{\partial s},$$

we have

$$\frac{\partial\Phi}{\partial s} = \frac{m}{s}(1-\rho)\Phi.$$

Now, we get

$$\frac{\partial K_1}{\partial s} = -DK_1\frac{m}{\bar{r}},$$

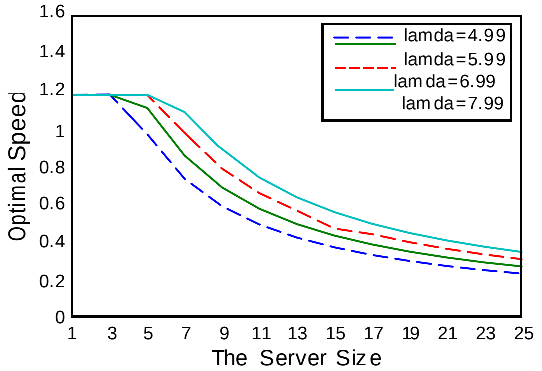
and

$$\frac{\partial K_2}{\partial s} = \sqrt{2\pi m}(\rho + m(1-\rho)^2)\frac{\Phi}{s}.$$

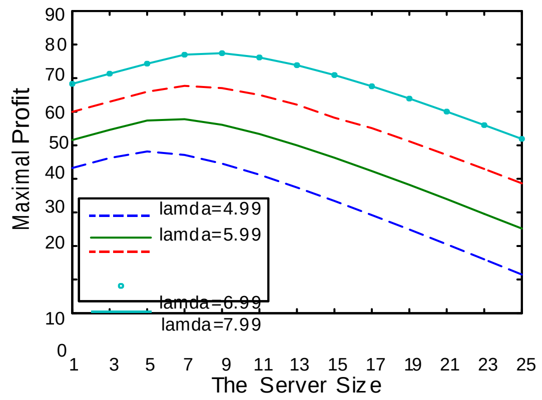
$$\frac{\partial p_{ext}(D)}{\partial s} = \frac{1}{(K_2-\rho K_1)^2}\Big[\frac{\rho}{s}(K - K) + (\rho-1)K_1\frac{\sqrt{2\pi m}(\rho + m(1-\rho)^2)\Phi}{s} + (\rho-1)DK_1K_2\frac{m}{\bar{r}}\Big].$$

Similarly, we cannot get the closed-form expression of s, so we can use the same method to find the numerical solution of s. In Fig. 8, we demonstrate the net profit in one unit of time as a function of s and λ, where $m = 6$. The rest parameters are the same as that in Figs. 6 and 7. We notice that there is an optimal choice of s such that the net profit is maximized. Using the analytical method, the optimal value of s such that respectively. When the servers run at a slower speed than the optimal speed, the waiting times of service requests will be long and exceed the deadline. So, the revenue is small and the profit is not optimal. When s increases, the energy consumption as well as the electricity cost increases. Hence, the increased revenue is much less than the increased cost. As a result, the profit is reduced. Therefore, there is an optimal choice of s such that the net profit is maximized.

In Fig. 9, we demonstrate the optimal speed and maximal profit in one unit of time as a function of m and λ. The parameters are same as that in Figs. 6–8. From the figures we can see that if the number of fixed servers is great, the servers must run at a lower speed, which can lead to an optimal profit. In addition, the optimal speed of servers is not faster than 1.2, that is because the increased electricity cost surpasses the increased cost that rents extra servers. The figure also shows us that different λ values have different optimal combinations of speed and size.



(a) Optimal speed versus m and λ.



(b) Maximal profit versus m and λ.

Fig. 9: Optimal speed and maximal profit versus m and λ.

5.1.3 *Optimal Size and Speed*

Given $\lambda, \bar{r}, a, P^*, \alpha, \beta, \gamma, \delta, \xi, D$, our third problem is to find m and s such that *Profit* is maximized. Hence, we need to find m and s such that $\partial Profit/\partial m = 0$ and $\partial Profit/\partial s = 0$, where $\partial Profit/\partial m$ and $\partial Profit/\partial s$ have been derived in the

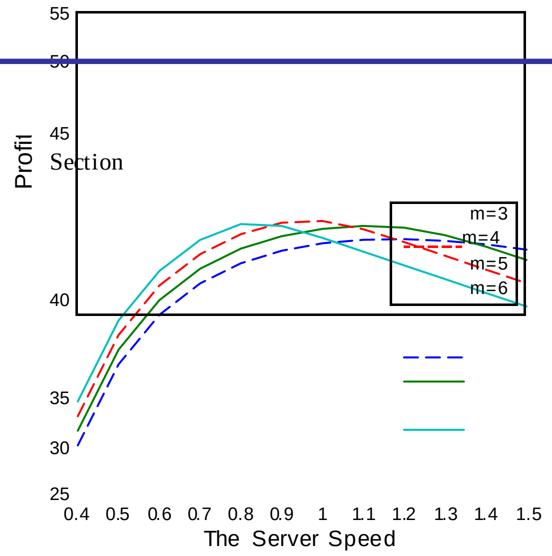


Fig. 10: Net profit versus m and s.

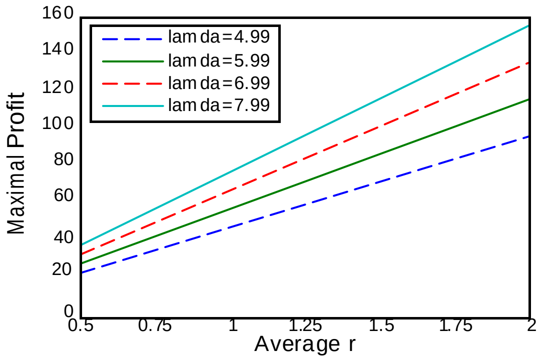


Fig. 11: Maximal profit versus λ and $\bar{r}$.

last two sections. The two equations are solved by using the same method as [2]. In Fig. 10, we demonstrate the net profit in one unit of time as a function of m and s. Here λ is 5.99, and $\bar{r} = 1$. The optimal value is $m = 6.2418$ and $s = 0.9386$, which result in the maximal profit 58.0150. In Fig. 11, we demonstrate the maximal profit in one unit of time in different combinations of λ and r. The figure shows that the service providers can obtain more profit when the service requests are with greater λ and $\bar{r}$.

5.2 An Algorithmic Method for Actual Solutions

In above subsection, the optimal solutions find using the analytical method are ideal solutions. Since the number of rented servers must be integer and the server speed levels are discrete and limited in real system, we need to find the optimal solutions for the discrete scenarios. Assume that $S = \{s_i | 1 \le i \le n\}$ is a discrete set of n speed levels with increasing order. Next, different situations are discussed and the corresponding methods are given as follows.

5.2.1 Optimal Size

Assume that all servers run at a given execution speed s. Given $\lambda, r, a, P^*, \alpha, \beta, \gamma, \delta, \xi$, and D, the first problem is to find the number of long-term rented servers m such that the profit is maximized. The method is shown in Algorithm 2.

Algorithm 2 Finding the optimal size

Input: $s, \lambda, \bar{r}, a, P^*, \alpha, \beta, \gamma, \delta, \xi$, and D
Output: the optimal number Opt_size of fixed servers

```
Profit_max ← 0
find the server size m using the analytical method in Section 5.1.1
m*_l ← ⌊m⌋, m*_u ← ⌈m⌉
Profit_l ← Profit(m*_l, s), Profit_u ← Profit(m*_u, s)
if Profit_l > Profit_u then
    Profit_max ← Profit_l
    Opt_size ← m*_l
else
    Profit_max ← Profit_u
    Opt_size ← m*_u
end if
```

5.2.2 Optimal Speed

Assume that the service provider rents m servers. Given λ, $\bar{r}, a, P^*, \alpha, \beta, \gamma, \delta, \xi$, and D, the second problem is to find the optimal execution speed of all servers such that the profit is maximized. The method is shown in Algorithm 3.

Algorithm 3 Finding the optimal speed

Input: $m, \lambda, \bar{r}, a, P^*, \alpha, \beta, \gamma, \delta, \xi$, and D
Output: the optimal server speed Opt_speed

```
Profit_max ← 0
find the server speed s using the analytical method in Section 5.1.2
s*_l ← s_i, s*_u ← s_{i+1} if s_i < s ≤ s_{i+1}
Profit_l ← Profit(m, s*_l), Profit_u ← Profit(m, s*_u)
if Profit_l > Profit_u then
    Profit_max ← Profit_l
    Opt_speed ← s*_l
else
    Profit_max ← Profit_u
    Opt_speed ← s*_u
end if
```

5.2.3 Optimal Size and Speed

In this subsection, we solve the third problem, which is to find the optimal combination of m and s such that the profit is maximized. Given $\lambda, \bar{r}, a, P^*, \alpha, \beta, \gamma, \delta, \xi$, and D, the method is shown in Algorithm 4.

Algorithm 4 Finding the optimal size and speed

Input: $\lambda, \bar{r}, a, P^*, \alpha, \beta, \gamma, \delta, \xi$, and D
Output: the optimal number Opt_size of fixed servers and the optimal execution speed Opt_speed of servers

```
Profit_max ← 0
find the server size m and speed s using the analytical method in Section 5.1.3
m*_l ← ⌊m⌋, m*_u ← ⌈m⌉
find the optimal speed s*_l and s*_u using Algorithm 3 with server size m*_l and m*_u, respectively
Profit_l ← Profit(m*_l, s*_l), Profit_u ← Profit(m*_u, s*_u)
if Profit_l ≤ Profit_u then
    Profit_max ← Profit_u
    Opt_size ← m*_u , Opt_speed ← s*_u
else
    Profit_max ← Profit_l
    Opt_size ← m*_l , Opt_speed ← s*_l
end if
```

5.3 Comparison of Two Kinds of Solutions

In Tables 1, 2, and 3, the ideal optimal solutions and the actual optimal solutions are compared for three different cases. Table 1 compares the ideal optimal size and the actual optimal size under the given server speed. Table 2 compares the ideal optimal speed and the actual optimal speed under the given server size. In Table 3, two kinds of solutions are compared for different combinations of λ and r. Here, m can be any positive integer, and the available speed levels are $S = \{0.2, 0.4, \cdots, 2.0\}$. According to the comparisons we can see that the ideal maximal profit is greater than the actual maximal profit. In the tables, we also list the *relative difference (RD)* between the ideal optimal profit and the actual optimal profit, which is calculated as

$$RD = \frac{Ide_p - Act_p}{Act_p},$$

where Ide_p and Act_p are the maximal profit in ideal and actual scenarios. From the results we know that the relative difference is always small except some cases in Table 2. That is because a small difference of speed would lead to a big difference of profit when the server size is large.

6 Performance Comparison

Using our resource renting scheme, temporary servers are rented for all requests whose waiting time are equal to the deadline, which can guarantee that all requests are served with high service quality. Hence, our scheme is superior to the traditional resource renting scheme in terms of the service quality. Next, we conduct a series of calculations to compare the profit of our renting scheme and the renting scheme in [2]. In order to distinguish the proposed scheme and the compared scheme, the proposed scheme is renamed as Double-Quality-Guaranteed (DQG) renting scheme and the compared scheme is renamed as Single-Quality-Unguaranteed (SQU) renting scheme in this paper.

6.1 The Compared Scheme

Firstly, the average charge of the using the SQU renting scheme is analyzed.

Theorem 6.1. The expected charge to a service request using the SQU renting scheme is

$$a\bar{r}(1 - P_q e^{-(1-\rho)m\mu D}).$$

Proof 6.1. Recall that the probability distribution function of the waiting time W of a service request is

$$f_W(t) = (1 - P_q)u(t) + m\mu\pi_m e^{-(1-\rho)m\mu t}.$$

Since W is a random variable, so $R(r, W)$ is also a random variable. The expected charge to a service request with execution requirement r is

$$\begin{aligned} R(r) &= \overline{R(r, W)} \\ &= \int_0^\infty f_W(t)R(r,t)dt \\ &= \int_0^D \left[(1 - P_q)u(t) + m\mu\pi_m e^{-(1-\rho)m\mu t}\right] ar\,dt \\ &= (1 - P_q)ar + m\mu\pi_m ar \frac{1 - e^{-(1-\rho)m\mu D}}{(1-\rho)m\mu} \\ &= ar(1 - P_q e^{-(1-\rho)m\mu D}). \end{aligned}$$

Therefore, the expected charge to a service request is the expected value of $R(r)$:

$$\begin{aligned} \overline{R(r)} &= \int_0^\infty f_r(z)R(z)dz \\ &= \int_0^\infty \frac{1}{\bar{r}} e^{-z/\bar{r}} az(1 - P_q e^{-(1-\rho)m\mu D})dz \\ &= \frac{a}{\bar{r}}(1 - P_q e^{-(1-\rho)m\mu D}) \int_0^\infty e^{-z/\bar{r}} z\,dz \\ &= a\bar{r}(1 - P_q e^{-(1-\rho)m\mu D}). \end{aligned}$$

The theorem is proven.

By the above theorem, the profit in one unit of time using the SQU renting scheme is calculated as:

$$\lambda a\bar{r}(1 - P_q e^{-(1-\rho)m\mu D}) - m(\beta + \delta(\rho\xi s^\alpha + P^*)). \quad (11)$$

Using the SQU renting scheme, a service provider must rent more servers or scale up the server speed to maintain a high quality-guaranteed ratio. Assumed that the required quality-guaranteed ratio of a service provider is ψ and the deadline of service requests is D. By solving equation

$$F_W(D) = 1 - \frac{\pi_m}{1-\rho} e^{-m\mu(1-\rho)D} \geq \psi$$

with given m or s, we can get the corresponding s or m such that the required quality-guaranteed ratio is achieved.

6.2 Profit Comparison under Different Quality-Guaranteed Ratio

Let λ be 5.99 and the other parameters be the same as those in Section 5. In the first example, for a given number of servers, we compare the profit using the SQU renting scheme with quality-guaranteed ratio 100%, 99%, 92%, 85% and the optimal profit using our DQG renting scheme. Because the quality-guaranteed ratio 100% cannot be achieved using the SQU renting scheme, hence, we set 99.999999% $\approx$ 100%. The results are shown in Fig. 12. From the figure, we can see that the profit obtained using the proposed scheme is always greater than that using the SQU renting scheme, and the five curves reach the peak at different sizes. In addition, the profit obtained by a service provider increases when the qualtiy-guaranteed ratio increases from 85% to 99%, but decreases when the ratio is greater than 99%. That is because more service requests are charged with the increasing ratio from 85% to 99%; but once the ratio is greater than 99%, the cost to expand the server size is greater than the revenue obtained from the extra qualtiy-guaranteed requests, hence, the total profit is reduced.

In the second example, we compare the profit of the above five scenarios under the given server speed. The results are given in Fig. 13. The figure shows the trend of profit when the server speed is increasing from 0.1 to 2.9. From the figure, we can see that the curves increase firstly and reach the peak at certain speed, and then decrease along with the increasing speed on the whole. The figure verifies that our proposed scheme can obtain more profit than the SQU renting scheme. Noticed that the changing trends of the curves of the SQU renting scheme with 100%, 99%, 92%, and 85% quality-guaranteed ratio are interesting. They show an increasing trend at the beginning and then decrease during a small range of speed repeatedly. The reason is

TABLE 1: Comparison of the two methods for finding the optimal size

	Given Speed	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
Ideal Solution	Optimal Size	29.1996	14.6300	9.7599	7.3222	5.8587	4.8827	4.1854	3.6624	3.2555	2.9300
	Maximal Profit	11.5546	45.5262	54.6278	57.5070	57.8645	56.9842	55.3996	53.3498	51.0143	48.4578
Actual Solution	Optimal Size	29	15	10	7	6	5	4	4	3	3
	Maximal Profit	11.5268	45.4824	54.6014	57.3751	57.8503	56.9727	55.3259	53.0521	50.8526	48.4513
Relative Difference		0.2411%	0.0964%	0.0483%	0.2299%	0.0246%	0.0202%	0.1332%	0.5612%	0.3180%	0.01325%

TABLE 2: Comparison of the two methods for finding the optimal speed

	Given Size	5	7	9	11	13	15	17	19	21	23
Ideal Solution	Optimal Speed	1.1051	0.8528	0.6840	0.5705	0.4895	0.4288	0.3817	0.3440	0.3132	0.2875
	Maximal Profit	57.3742	57.7613	56.0783	53.3337	49.9896	46.2754	42.3167	38.1881	33.9366	29.5933
Actual Solution	Optimal Speed	1.0	0.8	0.8	0.6	0.6	0.4	0.4	0.4	0.4	0.4
	Maximal Profit	57.0479	57.3751	54.7031	53.1753	48.4939	45.4824	42.2165	37.4785	32.6795	27.8795
Relative Difference		0.5721%	0.6732%	2.5140%	0.2979%	3.0843%	1.7435%	0.2373%	1.8934%	3.8470%	6.1474%

TABLE 3: Comparison of the two methods for finding the optimal size and the optimal speed

		r	0.50	0.75	1.00	1.25	1.50	1.75	2.00
$\lambda = 4.99$	Ideal Solution	Optimal Size	2.5763	3.8680	5.1608	6.4542	7.7480	9.0420	10.3362
		Optimal Speed	0.9432	0.9422	0.9413	0.9406	0.9399	0.9394	0.9388
		Maximal Profit	24.0605	36.0947	48.1539	60.1926	72.2317	84.3121	96.3528
	Actual Solution	Optimal Size	3	4	5	6	7	9	10
		Optimal Speed	1.0	1.0	1.0	1.0	1.0	1.0	1.0
		Maximal Profit	23.8770	35.7921	48.0850	60.1452	72.0928	83.9968	96.2230
	Relative Difference		0.7695%	0.8454%	0.14355%	0.0789%	0.1927%	0.3754%	0.1349%
$\lambda = 5.99$	Ideal Solution	Optimal Size	3.1166	4.6787	6.2418	7.8056	9.3600	10.9346	12.4995
		Optimal Speed	0.9401	0.9393	0.9386	0.9380	0.9375	0.9370	0.9366
		Maximal Profit	28.9587	43.4364	57.9339	72.4121	86.9180	101.3958	115.9086
	Actual Solution	Optimal Size	3	4	6	7	9	10	12
		Optimal Speed	1.0	1.0	1.0	1.0	1.0	1.0	1.0
		Maximal Profit	28.9158	43.1208	57.8503	72.2208	86.7961	101.2557	115.7505
	Relative Difference		0.1484%	0.7317%	0.1445%	0.2649%	0.1405%	0.1384%	0.1365%

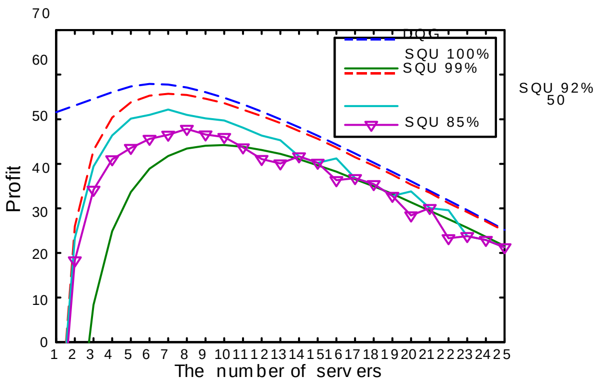


Fig. 12: Profit versus *m* and different quality-guaranteed ratios.

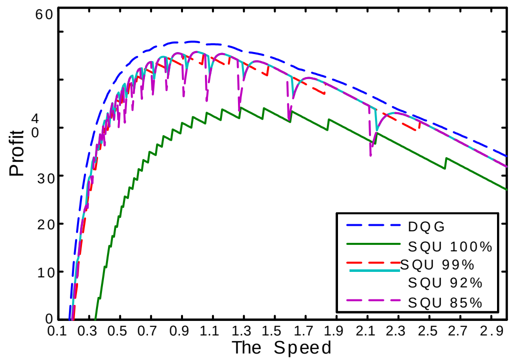
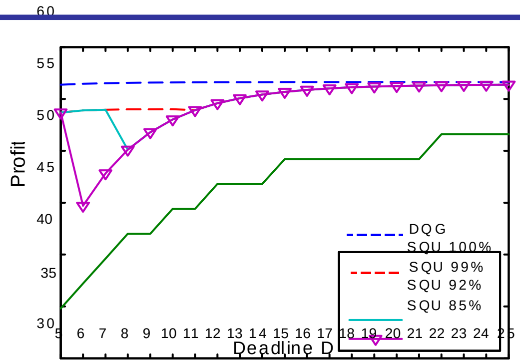


Fig. 13: Profit versus *s* and different quality-guaranteed ratios.

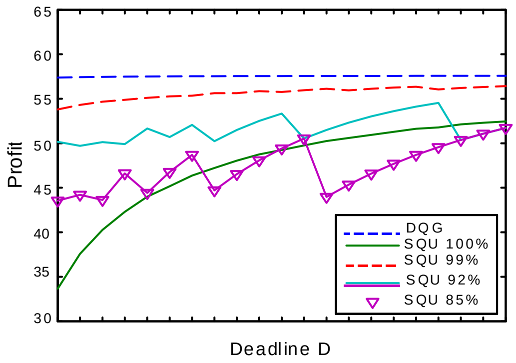
analyzed as follows. When the server speed is changing within a small speed range, in order to satisfy the required deadline-guaranteed ratio, the number of servers rented by a service provider keeps unchanged. At the beginning, the added revenue is more than the added cost, so the profit is increasing. However, when the speed becomes greater, the energy consumption increases, leading to the total increased cost surpassing the increased revenue, hence, the profit decreases.

In the third example, we explore the changing trend of the profit with different *D*, and the results are shown as Fig. 14. Fig. 14(a) gives the numerical results when the server speed is fixed at 0.7, and Fig. 14(b) shows the numerical results when the number of servers is fixed at 5. We analyze the results as follows.

From Fig. 14(a), we can see that the profit obtained using the SQU renting scheme increases slightly with the increment of *D*. That is because the service charge keeps constant but the extra cost is reduced when *D* is greater. As a consequence, the profit increases. The second phenomenon from the figure is that the curves of *SQU 92%* and *SQU 85%* have sharp drop at some points and then ascend gradually



(a) Fixed server speed $s = 0.7$.



(b) Fixed server size $m = 5$.

Fig. 14: Profit versus D and different quality-guaranteed ratios.

and smoothly. The reasons are explained as follows. When the server speed is fixed, enough servers are needed to satisfy the given quality-guaranteed ratio. By calculating, we know that the number of required servers is the same for all D values in a certain interval. For example, *[5,7]* and *[8,25]* are two intervals of D for the curve of *SQU 92%*, and the required servers are 10 and 9, respectively. For all D within the same interval, their costs are the same with each other. Whereas, their actual quality-guaranteed ratios are different which get greater with the increasing D. Hence, during the same interval, the revenue gets greater as well as the profit. However, if the deadline increases and enters a different interval, the quality-guaranteed ratio sharply drops due to the reduced servers, and the lost revenue surpasses the reduced cost, hence, the profit sharply drops as well. Moreover, we can also see that the profit of *SQU 100%* is much less than the other scenarios. That is because when the quality-guaranteed ratio is great enough, adding a small revenue leads to a much high cost.

From Fig. 14(b), we can see that the curves of *SQU 92%* and *SQU 85%* descend and ascend repeatedly. The reasons are same as that of Fig. 14(a). The deadlines within the same interval share the same minimal speed, hence, the cost keeps constant. At the same time, the revenue increases due to the increasing quality-guaranteed ratio. As a consequence, the profit increases. At each break point, the minimal speed satisfying the required quality-guaranteed ratio gets smaller, which leads to a sharp drop of the actual quality-guaranteed ratio. Hence, the revenue as well as the profit drops.

6.3 Comparison of Optimal Profit

In order to further verify the superiority of our proposed scheme in terms of profit, we conduct the following com-

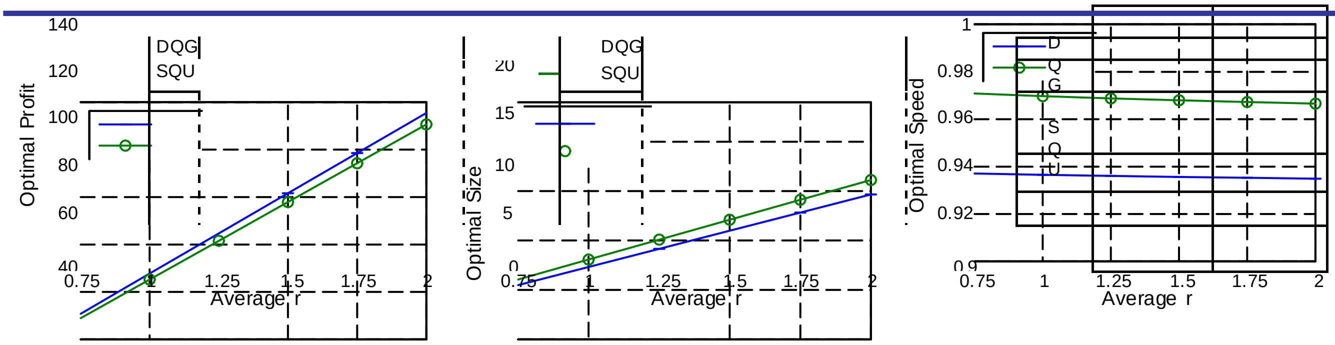
renting scheme and that of the SQU renting scheme in [2]. In this group of comparisons, λ is set as 6.99, D is 5, $\bar{r}$ is varying from 0.75 to 2.00 in step of 0.25, and the other parameters are the same as Section 5. In Fig. 15, the optimal profit and the corresponding configuration of two renting schemes are presented. From Fig. 15(a) we can see that the optimal profit obtained using our scheme is always greater than that using the SQU renting scheme. According to the calculation, our scheme can obtain 4.17 percent more profit on the average than the SQU renting scheme. This shows that our scheme outperforms the SQU renting scheme in terms of both of quality of service and profit. Figs. 15(b) and 15(c)

figures show that using our renting scheme the capacity provided by the long-term rented servers is much less than the capacity using the SQU renting scheme. That is because a lot of requests are assigned to the temporary servers using our scheme, and less servers and slower server speed are configured to reduce the waste of resources in idle period. In conclusion, our scheme can not only guarantee the service quality of all requests, but also achieve more profit than the compared one.

7 Conclusions

In order to guarantee the quality of service requests and maximize the profit of service providers, this paper has proposed a novel Double-Quality-Guaranteed (DQG) renting scheme for service providers. This scheme combines short-term renting with long-term renting, which can reduce the resource waste greatly and adapt to the dynamical demand of computing capacity. An $M/M/m+D$ queueing model is build for our multiserver system with varying system size. And then, an optimal configuration problem of profit maximization is formulated in which many factors are taken into considerations, such as the market demand, the workload of requests, the server-level agreement, the rental cost of servers, the cost of energy consumption, and so forth. The optimal solutions are solved for two different situations, which are the ideal optimal solutions and the actual optimal solutions. In addition, a series of calculations are conducted to compare the profit obtained by the DQG renting scheme with the Single-Quality-Unguaranteed (SQU) renting scheme. The results show that our scheme outperforms the SQU scheme in terms of both of service quality and profit.

In this paper, we only consider the profit maximization problem in a homogeneous cloud environment, because the analysis of a heterogenous environment is much more complicated than that of a homogenous environment. However, we will extend our study to a heterogenous environment in the future.



(a) Comparison of Profit. (b) Comparison of Server Size. (c) Comparison of Server Speed.

Fig. 15: Comparison between our scheme with that in [2].

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their valuable comments and suggestions. The research was partially funded by the Key Program of National Natural Science Foundation of China (Grant Nos. 61133005, 61432005), the National Natural Science Foundation of China (Grant Nos. 61370095, 61472124, 61173013, 61202109, and 61472126).

REFERENCES

[1] K. Hwang, J. Dongarra, and G. C. Fox, *Distributed and Cloud Computing*. Elsevier/Morgan Kaufmann, 2012.

[2] J. Cao, K. Hwang, K. Li, and A. Y. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1087–1096, 2013.

[3] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences*, vol. 28, 2009.

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comp. Sy.*, vol. 25, no. 6, pp. 599–616, 2009.

[5] P. Mell and T. Grance, "The NIST definition of cloud computing. national institute of standards and technology," *Information Technology Laboratory*, vol. 15, p. 2009, 2009.

[6] J. Chen, C. Wang, B. B. Zhou, L. Sun, Y. C. Lee, and A. Y. Zomaya, "Tradeoffs between profit and customer satisfaction for service provisioning in the cloud," in *Proc. 20th Int'l Symp. High Performance Distributed Computing*. ACM, 2011, pp. 229–238.

[7] J. Mei, K. Li, J. Hu, S. Yin, and E. H.-M. Sha, "Energy-aware preemptive scheduling algorithm for sporadic tasks on dvs platform," *MICROPROCESS MICROSY.*, vol. 37, no. 1, pp. 99–112, 2013.

[8] P. de Langen and B. Juurlink, "Leakage-aware multiprocessor scheduling," *J. Signal Process. Sys.*, vol. 57, no. 1, pp. 73–88, 2009.

[9] G. P. Cachon and P. Feldman, "Dynamic versus static pricing in the presence of strategic consumers," Tech. Rep., 2010.

[10] Y. C. Lee, C. Wang, A. Y. Zomaya, and B. B. Zhou, "Profit-driven scheduling for cloud services with data access awareness," *J. Parallel Distr. Com.*, vol. 72, no. 4, pp. 591–602, 2012.

[11] M. Ghamkhari and H. Mohsenian-Rad, "Energy and performance management of green data centers: a profit maximization approach," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 1017–1025, 2013.

[12] A. Odlyzko, "Should flat-rate internet pricing continue," *IT Professional*, vol. 2, no. 5, pp. 48–51, 2000.

[13] G. Kesidis, A. Das, and G. de Veciana, "On flat-rate and usage-based pricing for tiered commodity internet services," in *42nd Annual Conf. Information Sciences and Systems*. IEEE, 2008, pp. 304–308.

[14] S. Shakkottai, R. Srikant, A. Ozdaglar, and D. Acemoglu, "The price of simplicity," *IEEE J. Selected Areas in Communications*, vol. 26, no. 7, pp. 1269–1276, 2008.

[15] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Trans. Cloud Computing*, vol. 1, no. 2, pp. 158–171, July 2013.

[16] M. Macías and J. Guitart, "A genetic model for pricing in cloud computing markets," in *Proc. 2011 ACM Symp. Applied Computing*, 2011, pp. 113–118.

[17] "Amazon EC2," http://aws.amazon.com, 2014.

[18] "Amazon EC2 spot instances," http://aws.amazon.com/ec2/spot-instances, 2014.

[19] R. L. Hall and C. J. Hitch, "Price theory and business behaviour," *Oxford economic papers*, no. 2, pp. 12–45, 1939.

[20] D. Kahneman, J. L. Knetsch, and R. Thaler, "Fairness as a constraint on profit seeking: Entitlements in the market," *The American economic review*, pp. 728–741, 1986.

[21] D. E. Irwin, L. E. Grit, and J. S. Chase, "Balancing risk and reward in a market-based task service," in *13th IEEE Int'l Symp. High performance Distributed Computing*, 2004, pp. 160–169.

[22] J. Heo, D. Henriksson, X. Liu, and T. Abdelzaher, "Integrating adaptive components: An emerging challenge in performance-adaptive systems and a server farm case-study," in *RTSS 2007*, Dec 2007, pp. 227–238.

[23] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Dynamic cluster reconfiguration for power and performance," in *Compilers and operating systems for low power*. Springer, 2003, pp. 75–93.

[24] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2. ACM, 2007, pp. 13–23.

[25] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5. ACM, 2001, pp. 103–116.

[26] M. Mazzucco and D. Dyachuk, "Optimizing cloud providers revenues via energy efficient server allocation," *Sustainable Computing: Informatics and Systems*, vol. 2, no. 1, pp. 1–12, 2012.

[27] Y.-J. Chiang and Y.-C. Ouyang, "Profit optimization in sla-aware cloud services with a finite capacity queuing model," *Math. Probl. Eng.*, 2014.

[28] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth,