A Profit Maximization Scheme with Guaranteed Quality of Service in Cloud Computing

KOLLURU PRADEEPIKA[1], KOLLI ANJALI[2], KOLLIMARLA PAVANI[3], SK. KHAJA MOHIDDIN[4]

[1,2,3,4] Computer Science and Engineering, Vasireddy Venkatadri Institute of Technology, Guntur, Andhra Pradesh, India

Abstract -- Cloud computing has become more popular in terms of providing resources and services to the costumers. In cloud service provider perspective profit is one of the most important consideration. Profit can be determined by the configuration of cloud service platform under given market demand. At present single long term renting scheme is used which has some of the drawbacks like resource wastage, no guaranteed service quality. To overcome these drawbacks, in this paper we are proposing a double resource renting scheme. A double resource renting scheme is the combination of short-term renting and long-term renting aiming at resolving the existing issues. Secondly, service system is assumed as M/M/m+D queuing model and the performance indicators are also analyzed. Here the waiting time (D) is not fixed like the existing system. Thirdly, a profit maximization problem is formulated for the double renting scheme and solved to obtain optimized configuration of cloud platform. Finally, a series of calculations are conducted to compare the profit of our proposed scheme with that of existing scheme. Obtained results specify that our proposed scheme not only guarantee the quality of all requests but also generated profit more than the existing system.

Index terms: Cloud computing, guaranteed service quality, profit maximization, Queuing model, waiting time.

I. INTRODUCTION

Cloud computing has become more and more popular to consolidate computing resources and services [1]. Cloud computing centralizes management of resources and services, and delivers host services over the internet. The hardware, software, databases, information and all resources are concentrated and provided to the consumers on demand. Usually there are three tiers in the cloud computing environment: Infrastructure provider, Service provider and Customers. Customers request services from service provider and pays for the amount and quality of service they have used [2]. In this paper, we aim at multi server configuration of a service provider such that their profit is maximized.

Usually profit of service provider depends upon two parts, which are the cost and revenue. Cost for a service provider is the renting cost provided to the infrastructure provider plus the electricity cost caused by the energy consumption. Revenue is the service charge to the customers. Infrastructure provider provides certain number of servers on rent to service provider, service provider builds different multi server systems for different application domains. Each multi server system is to execute a special type of service requests and applications. Hence, the renting cost is proportional to number of servers in the multiserver system. The power consumption of a multiserver is linearly proportional to number of servers and the server utilization, and to the square of execution speed [3,4]. The revenue of a service provider depends upon two factors: amount of service and quality of service. In short, the profit of a service provider is mainly determined by the configuration of its service platform.

Usually service providers adopt single renting scheme to configure a cloud platform. Therefore the servers in a service system are long term rented. Due to long term renting and shortage of servers, some of the incoming requests cannot be processed immediately. So a queue is maintained and the incoming requests are first inserted in the queue until they can be handled by the available servers. The waiting time of service requests cannot be too long. The waiting time of each incoming request should be limited within a range, which is determined by the service-level-agreement. The charge of a service depends upon the quality of service like if the quality of service is guaranteed, the service is fully charged. Otherwise the service provider serves the request for free as the penalty of low quality. To scale up the revenue a service provider should rent more number of servers from infrastructure providers or increase the execution speed such that more requests are processed with high service quality. This may sometimes lead to increase of renting cost or electricity cost. Such increased cost affects the profit of service providers. In this paper, we propose a novel renting scheme for service providers, which not only satisfy quality of service requirements, but also can obtain more profit. Our contributions in this paper are:

- A novel double renting scheme is proposed for service providers. It combines long term renting with short term renting.
- Multiserver system adopted in our paper is modeled as M/M/m+D queuing model.
- The performance indicators are analyzed such as the average service charge, ratio of requests that need short term servers and so on.
- 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.

II. RELATED WORK

In this section, we review recent works related to 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 [5]. 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. 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. Usage-based pricing reveals that one can use resources more efficiently.

Dynamic pricing emerges as an attractive alternative to better cope with unpredictable customer demand[6]. Amazon EC2[7,8] 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”. 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.

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 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 [9,10,11,12], 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.

Chiang and Ouyang 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. In, 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 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 . 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 produce more profit than the optimal configuration.

III. 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.

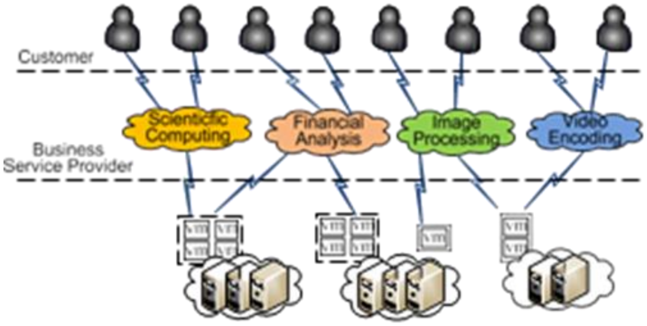


Fig. 1: The three-tier cloud structure

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.

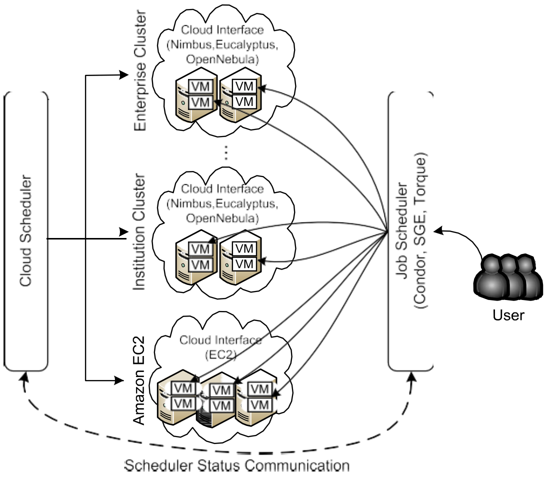

Cloud Scheduler
Enterprise Cluster
Cloud Interface (Nimbus,Eucalyptus, OpenNebula)
VM
Institution Cluster
Cloud Interface (Nimbus,Eucalyptus, OpenNebula)
Amazon EC2
Cloud Interface (EC2)
Job Scheduler (Condor, SGE, Torque)
User
Scheduler Status Communication

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, Open Nebula, 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 queuing 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 they based upon a policy. 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.

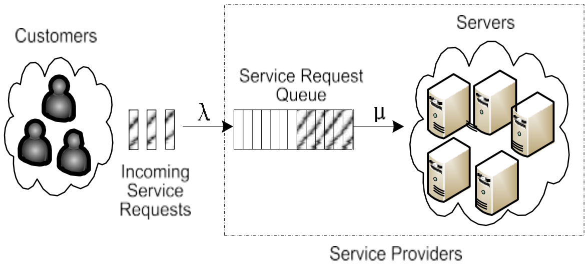


Fig. 3: The multiserver system model, where service requests are first placed in a queue before they are processed by any servers.

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

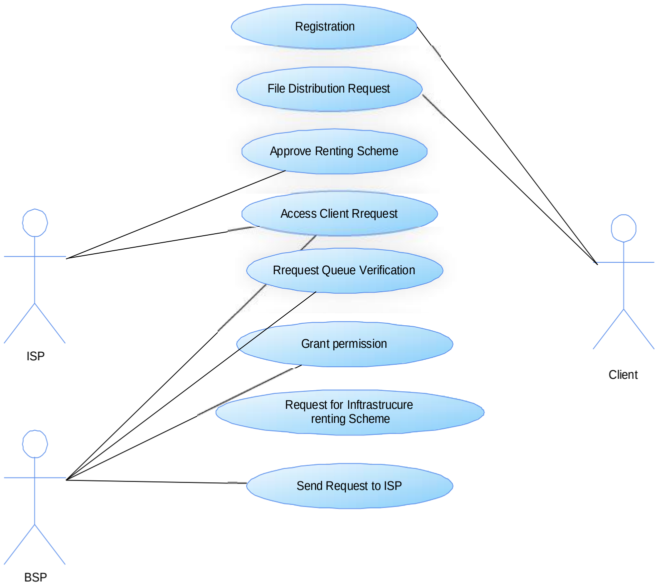


Fig 4: Usecase diagram of proposed system

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 inter arrival 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 calculate as $\mu = 1/\bar{x} = s/\bar{r}$, and the system utilization is defined as $\rho = \lambda/m\mu = \lambda/m \_ \bar{r}/s$. 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.

3.3 REVENUE MODELLING

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 inthe queue until any server is available. However, to satisfythe 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, Joyent , Microsoft Azure , and so on. Taking Joyent as an example, the customer's 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.

3.4 COST MODELLING

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 researches 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. 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.

IV. 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 Proposed System

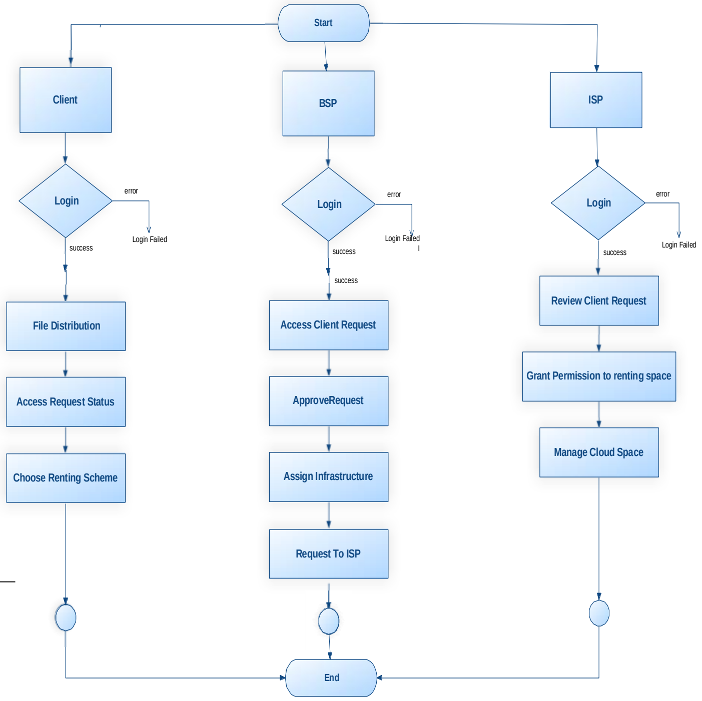
In this section, we first propose the Double -Quality-Guaranteed (DQG) resource renting scheme which com- bines long-term renting with short-term renting. The main computing capacity is provided by the long-term rented servers due to their low price. The short-term rented servers provide the extra capacity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

In this algorithm deadline D is assumed as random not fixed. It is different for different requests.

Algorithm 1 Double-Quality-Guaranteed (DQG) Scheme

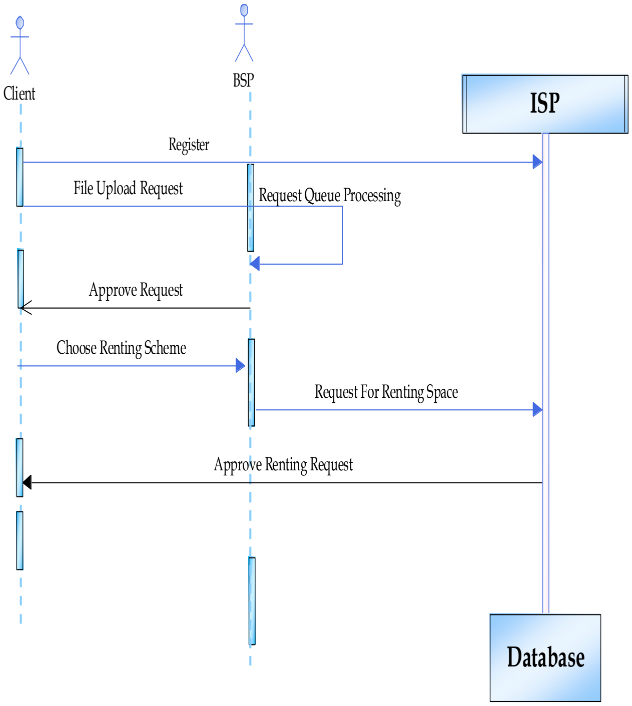
```
A multiserver system with m servers is running and waiting for the events as follows
A queue Q is initialized as empty
Event – A service request arrives
Search if any server is available
If true then
    Assign the service request to one available server
Else
    Put it at the end of the queue Q 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 the queue Q and assign it to the idle server
End if
End Event
Event – if the specific deadline(D) of a specific request is achieved
Rent a temporary server to execute the request and release the temporary server when the request is completed
End Event
```

4.2 Working of Proposed System in DFD



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. The customer rent the two types of renting scheme viz long term and short term renting. 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.

4.3 Working of Proposed System Using Sequence Diagram



V. OPTIMAL SOLUTION

5.1 OPTIMAL SIZE

```
Algorithm 2  Finding the optimal size Input:  s, λ, r, a, P ∗, α, β, γ, δ, ξ, and D
Output: The optimal number Opt size of fixed servers
Profit_max ← 0
find the server size m using the analytical method
m∗l ← ⌊m⌋, m∗u ← ⌈m⌉
Profitl ← Profit(m∗l, s), Profitu ←   Profit(m∗u,s)
if Profitl > Profitu then Profit_max ← Profitl
    Opt_size ← m∗l0
else
    Profit_max ← Profitu
    Opt_size ← m∗u
end if
```

5.2 OPTIMAL SPEED

```
Algorithm 3 Finding the optimal speed
Input: m, λ, r, a, P∗, α, β, γ, δ, ξ, and D
Output: the optimal server speed Opt speed
Profi_ max ← 0
find the server speed s using the analytical method
s∗l ← si, s∗u ← si+1 if si < s ≤ si+1
Profitl ← Profit(m, s∗l), Profitu ← Profit(m, s∗u)
if Profitl > Profitu then
    Profi_ max ← Profitl
    Opt_ speed ← s∗l
else
    Profi¬_ max ← Profitu
    Op¬t¬_ speed ← s∗u
end if
```

VI. PERFORMANCE COMPARISION

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 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 Profit Comparison under Different Quality-Guaranteed Ratio

Let λ be 5.99 and the other parameters be the same. 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% _100%. 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 quality-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 quality-guaranteed requests, hence, the total profit is reduced.

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. 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.

6.2 Comparison of Optimal Profit

In order to further verify the superiority of our proposed scheme in terms of profit, we conduct the following comparison between the optimal profit achieved by our DQG renting scheme and that of the SQU renting scheme. In this group of comparisons, λ is set as 6.99,D is 5, r is varying from 0.75 to 2.00 in step of 0.25, and the other parameters are the same . The optimal profit of corresponding configuration of two renting schemes are presented. 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.

VII. 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 queuing model is built for our multiserver system with varying system size. And then, an optimal configuration problem of profit maximization is formulated .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.

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

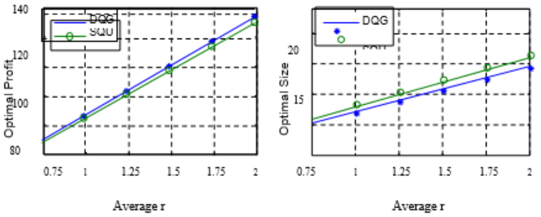
	Given Speed	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
Ideal Solution	Optimal Size	29.1996	14.6300	9.7599	7.3222	5.8587	4.8827	4.1854	3.6624	3.2555	2.9300
	Maximal Profit	11.5546	45.5262	54.6278	57.5070	57.8645	56.9842	55.3996	53.3498	51.0143	48.4578
Actual Solution	Optimal Size	29	15	10	7	6	5	4	4	3	3
	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 indicating 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%



(a)Comparison of Profit.

(b) Comparison of Server size

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their valuable comments and suggestions

REFERENCES

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

[2] 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.

[3] J. Mei, K. Li, J. Hu, S. Yin, and E. H.-M. Sha, “Energyaware preemptive scheduling algorithm for sporadic tasks on dvs platform,”

MICROPROCESS MICROSY., vol. 37, no. 1, pp. 99–112, 2013.

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

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

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

[7] “Amazon EC2,” http://aws.amazon.com, 2014.

[8] “Amazon EC2 spot instances,” http://aws.amazon.com/ ec2/spot-instances, 2014.

[9] J. Heo, D. Henriksson, X. Liu, and T. Abdelzaher, “Integrating adaptive components: An emerging challenge in performance-adaptive systems and a server farm casestudy,” in RTSS 2007, Dec 2007, pp. 227–238.

[10] 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.

[11] 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.

[12] 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.