Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 534510, 11 pages
http://dx.doi.org/10.1155/2014/534510

Research Article

Profit Optimization in SLA-Aware Cloud Services with a Finite Capacity Queuing Model

Yi-Ju Chiang and Yen-Chieh Ouyang

Department of Electrical Engineering, National Chung Hsing University, Taichung 40227, Taiwan

Correspondence should be addressed to Yen-Chieh Ouyang; ycouyang@nchu.edu.tw

Received 26 February 2014; Accepted 24 April 2014; Published 5 June 2014

Academic Editor: Her-Terng Yau

Cloud computing realizes a utility computing paradigm by offering shared resources through an Internet-based computing. However, how a system control can enhance profit and simultaneously satisfy the service level agreements (SLAs) has become one of the major interests for cloud providers. In this paper, a cloud server farm provided with finite capacity is modeled as an $M/M/R/K$ queuing system. Revenue losses are estimated according to the system controls and impatient customer behaviors. Three important issues are solved in this paper. First, a profit function is developed in which both the system blocking loss and the user abandonment loss are evaluated in total revenue. A tradeoff between meeting system performances and reducing operating costs is conducted. Second, the effects of system capacity control and utilization on various performances of waiting time, loss probability, and final arrival rate are demonstrated. Finally, the proposed optimal profit control (OPC) policy allows a cloud provider to make the optimal decision in the number of servers and system capacity, so as to maximize profit. As compared to a system without applying the OPC policy, enhancing providers' profit and improving system performances can be obtained.

1. Introduction

Cloud computing is a popular service paradigm recently in an information technology (IT) industry that offers infrastructure, platform, software, and so forth as services. Consumers can eliminate the burden of expensive hardware/software spending and complex infrastructure management. Service resources (e.g., networks, servers, storage, applications, and services) are allowed to be dynamically rented with minimal management effort based on consumer's needs rather than traditional "own-and-use" patterns. For example, Amazon EC2, Google's AppEngine, Microsoft's Azure, and so forth are some typical cloud service providers.

Before starting a service, there has been a formal contract known as service level agreement (SLA) that needs to be agreed and signed by consumers and cloud providers in advance. Quality of service (QoS) is a part of a SLA negotiation to specify consumer's expectation in terms of constraints [1]. Mean waiting time, expected queuing length, blocking probability, mean time to failure (MTTF), and so forth are all important performance indicators. In case of violating the SLA contract, a penalty or compensation is required in accordance with seriousness. Hence, it is crucial to do an accurate performance analysis due to the fact that a performance guarantee plays a crucial role in a service-oriented system.

Since no system is accompanied by infinite waiting buffer space, an analytic queuing model with finite capacity is suitable for the study of various server configuration settings [2, 3]. There are two common situations in a system that will result in revenue losses. First, "customer abandonment" can be found from communication systems to network services in daily life, no exception in cloud computing services. Abandonment means that an arrival task will leave a system without obtaining service due to long queuing length or latency. Second, reducing waiting buffer in a system is one of the simplest ways to mitigate system congestion and shorten queuing length. However, the overflowing tasks will be rejected from the system [4], which will directly result in revenue losses for a cloud provider.

Maintaining performance is usually the prime objective in system administration [5]. Besides, the relationship

between system control (e.g., server quantity and waiting buffer) and user abandonment (e.g., reneging) on effective arrival rate and performances must all be taken into account. In this paper, we discuss the problem of profit optimization in an abandonment system provided with finite capacity. Our contribution in this paper is threefold as follows.

(i) Blocking loss and abandonment loss that are rarely analyzed in previous works are considered in our cloud model to give a more practical analysis in revenue estimation. A profit function is developed by taking resources provisioning cost, power consumption cost, system congestion cost, and expected revenue into account.

(ii) The first presented optimal profit control (OPC) policy combined with a heuristic algorithm allows a cloud provider to effectively address constrained optimization problems. Besides, the effects of system capacities and utilizations on system performances and loss probabilities are demonstrated.

(iii) Simulations show that the optimal decision making in server quantity and system capacity to maximize profit within a loss probability guarantee can be obtained by applying the proposed OPC policy. Enhancing providers' profit and improving system performances can be verified as compared to a system without applying our policy.

The rest of the paper is organized as follows. Section 2 gives a brief overview of existing researches related to queuing models and profit optimization in cloud systems. A cloud service model with system capacity and server provisioning controls is presented in Section 3. The effects of system capacity and utilization on various performances are also demonstrated. In Section 4, a profit function is developed based on the expected revenue analysis and costs estimation. An algorithm (Algorithm 1) is presented to effectively solve constrained optimization problems. Section 5 shows the comparison of experimental results with respect to the optimal system control within a loss probability constraint. Finally, some conclusions are discussed in Section 6.

2. Related Work

2.1. Queuing Models with Finite Capacity. Researches in queuing models with finite capacity have a long history; here we refer to [6, 7]. In [6], the authors addressed a rate control problem associated with a single server. The controller could choose a buffer size for the queuing system and dynamically control the service rate depending on the current state of the system. An infinite horizon cost minimization problem was considered. An explicit optimal strategy for the limiting diffusion control problem was obtained. This solution was then used to construct an asymptotically optimal control policy.

An analytic cost model for M/G/1/N queuing systems was presented in [7]. The costs of customer loss versus customer delays by varying buffer size and processor speed were considered. In their analytic study, the authors explored the interplay of queue size, customer loss, and mean service time for various service time distributions. However, the possibility of adding multiple servers and some form of processor sharing were ignored in their work. Although cloud computing has attracted many research attentions [8–10] based on queuing models for a performance analysis, few literatures took buffer capacity restriction into consideration.

In [8], the authors described a novel approximate analytical model for performance evaluation of cloud server farms and solved it to obtain accurate estimation of complete probability distribution, request response time, and other important performance indicators. They also pointed out that accommodating heterogeneous services in a cloud center might impose longer waiting time for its clients as compared to its homogeneous equivalent with the same system utilization. However, they only focused on performance analysis; no cost analysis or profit evaluation was discussed in their research.

In [9], the authors presented algorithms for scheduling service requests and prioritizing their data accesses in a cloud service with the main objective of maximizing profit. The processor sharing (PS) was used for developing a pricing model for clouds, and then the data service was modeled as M/M/1/FCFS. They assumed that consumer's service requests might pay the service provider to reduce the rate of incoming requests in order to maintain a satisfactory response time. In [10], a cloud center was modeled as an M/M/m/m queuing system to conduct a preliminary study of the fault recovery impact on a cloud service performance.

When a user submitted a service request to the cloud, the request would first arrive at the cloud management system (CMS) which maintained a request queue. If the queue was not full, the request would enter the queue; otherwise it would be dropped and the service fails. Cloud service performance was quantified by service response time, whose probability density function was derived. This was similar to our queuing model but their work only focused on fault tolerance analysis. No system capacity control, cost analysis, or profit optimization was discussed in their work.

2.2. Profit Optimization in Cloud Systems. In [11], the authors had presented a characterization of cloud federation aimed at enhancing providers' profit and they characterized these decisions as a function of several parameters. They studied the effect of these decisions on the provider's profit and evaluated the most appropriate provider's configuration depending on the environment conditions. Evaluated parameters included the provider's incoming workload and the cost of maintaining the provider's resources operating. Results demonstrated that local resources were preferred over outsourced resources though the latter could enhance the provider's profit when the workload could not be supported locally.

In [12], the authors proposed policies that helped in the decision-making process to increase resources utilization and profit. Since each provider had restricted amount of capacity, increasing in load might overload a provider's data center and result in QoS violation or users' request rejection. This work attempted to incorporate the outsourcing

Input Data:
(1) Arrival rate λ.
(2) Potential abandonment index and expected revenue $[d(s,t),N(s,t)]$.
(3) Cost matrix $[C_H, C_W, C_U, C_R]$.
(4) A given execution rate μ and a baseline rate μ_B.
(5) Loss probability threshold T.
(6) The upper bound parameters (u and b) for server quantity and system capacity, respectively.
Output Data:
R^* and K^* and $F(R^*, K^*)$
Step 1. For $i = 1$; $i = u$; i++
Set $R_i \leftarrow a$ current server quantity;
Step 2. For $j = 0$; $j = b$; j++
Set $K_j \leftarrow a$ current system capacity;
Step 3. Calculate ρ_s, P_K, L_q, λ_d, λ^* and loss probability using (5)–(14) and (15), respectively.
Step 4. If loss probability $< T$
Then, record the current joint value of (R_i, K_j) and identify it as an approved test parameter;
Else
Return to Step 1 and begin to test next index of parameters;
End
Step 5. When all test parameters have been done, $\{R_{i+a}, K_{j+a}\}, \ldots, \{R_u, K_b\} \leftarrow$ current approved parameters;
Bring all revenue and cost parameters into the developed profit function and test all current approved parameters
Step 6. If a joint value of (R_{i+a}, K_{j+a}) obtains the maximum profit value in all tests,
Then
Output (R_{i+a}, K_{j+a}) and $F(R_{i+a}, K_{j+a})$;
Else
Return to Step 5 and begin to test next index of the approved parameters.
End

Algorithm 1: OPC algorithm.

problem with option of terminating spot VMs within a data center. Their objective was to maximize a provider's profit by accommodating as many on-demand requests as possible.

In [13], the authors treated a multiserver system as an M/M/m queuing model, such that the problem of optimal multiserver configuration for profit maximization in a cloud computing environment could be formulated and solved analytically. Their pricing model took the amount of a service, the workload of an application environment, the configuration of a multiserver system, the service level agreement, and so forth into consideration. They also considered two server speed and power consumption models, namely, the idle-speed model and the constant-speed model.

In [14], the response time based on different allocation of resources was modeled and used for different servers. The total profit in their system was the total price gained from the clients subtracted by the cost of operating the active servers in their system. The problem was formulized based on Generalized Processor Sharing and an elaborate multistage heuristic algorithm was used to solve the problem. However, previous studies did not take customer abandonment behaviors, system capacity control, loss probability estimation, and so forth into consideration. To the best of our knowledge, profit and system control analysis in a cloud computing system with abandonment events has not been investigated.

3. A Cloud Multiserver Model

3.1. M/M/R/K Queuing System. We consider a cloud server farm with finite waiting buffer and model it as an M/M/R/K queuing system [15]. The mathematical expressions are derived in detail as follows. There are R identical servers in operation and at most K tasks are allowed in the system including those in services and in queue. Task demands arrive from an infinite source with mean arrival rate λ of Poisson distribution. Service times are assumed to be independently

and identically distributed and have an exponential distribution with parameter μ. The service discipline is first-come-first-served (FCFS). In the cloud server farm, let the states n represent the number of tasks currently in the system. The values of the arrival rate and service rate are taken to be

$$\lambda_n = \begin{cases} \lambda, & 0 \le n \le K-1, \\ 0, & n \ge K, \end{cases} \qquad \mu_n = \begin{cases} n\mu, & 1 \le n \le R-1, \\ R\mu, & R \le n \le K. \end{cases} \tag{1}$$

We denote the notation P_n by the probability of n task currently served in the system ($n = 0, 1, 2, \ldots, \infty$) and P_o implies the probability that there is no task in system. For a steady-state case, the state probability functions P_n can be obtained from the birth-and-death formula. According to the value n (number of task services) that may happen, two segments are defined by the vector: [Segment 1, Segment 2] = $[1 \le n \le R-1, R \le n \le K]$. With the expressions in (1), the initial state probability functions P_n can be derived in terms of two segments as follows.

Segment 1: $1 \le n \le R-1$

$$P_n = \frac{\lambda_0 \cdot \lambda_1 \cdot \lambda_2 \cdots \lambda_{n-1}}{\mu_1 \cdot \mu_2 \cdot \mu_3 \cdots \mu_n}, \qquad P_o = \frac{\lambda^n}{n!\mu^n} P_o. \tag{2}$$

Segment 2: $R \le n \le K$

$$P_n = \frac{\lambda^n}{R!R^{n-R}\mu^n} P_o. \tag{3}$$

Equations (2) and (3) are the closed forms of the state probability functions P_n in which the number of tasks in service system may happen. To obtain P_o, (2) and (3) are brought into the normalizing equation: $\sum_{n=0}^{K} P_n = 1$,

$$\sum_{n=0}^{K} P_n = P_o \cdot \left(1 + \sum_{n=1}^{R-1} \frac{\lambda^n}{\mu^n n!} + \sum_{n=R}^{K} \frac{\lambda^n}{\mu^n R! R^{n-R}} \right). \tag{4}$$

Then, the steady-state probability of zero service P_o can be obtained as follows:

$$P_o = \left[\sum_{n=0}^{R-1} \frac{\lambda^n}{\mu^n n!} + \sum_{n=R}^{K} \frac{\lambda^n}{\mu^n R! R^{n-R}} \right]^{-1}. \tag{5}$$

3.2. A Designed Cloud Control Model. The proposed cloud service model is provided with finite capacity (denoted by K), as shown in Figure 1. The system blocking rate, effective arrival rate, user abandonment rate, and final arrival rate will be analyzed according to the system capacity and server provisioning control. Users are allowed to send task demands into the service system if there has been some waiting space left (as long as the current number of tasks in buffer are less than $K-R$). Otherwise, they would be rejected by prerejecting mechanism (PRM) and lost.

PRM is used to control and block excess tasks in advance before they are sent into the system. It is reasonable to assume that if customers are rejected before they send task demands, a service provider has no responsibility to give any compensation. Therefore, rejection penalties can be avoided [16] in this proposed model with PRM. A cloud system controller must distinguish carefully between the original arrival rate and the effective arrival rate, denoted by λ_α; the relevant notations are described in Notation Section.

The fraction of rejecting arrival tasks in a steady state are referred to as the blocking probability, denoted by P_K. This can be obtained by giving $n = K$ in (3) as follows:

$$P_K = \frac{\lambda^K}{R! \cdot R^{K-R}\mu^K}, \qquad P_o = \frac{\rho^K}{R! \cdot R^{K-R}} P_o. \tag{6}$$

The effective arrival rate can be obtained as

$$\lambda_\alpha = \lambda \left(1 - P_K\right). \tag{7}$$

Customers that are rejected still can come back later after a random time and they will be treated as a new arrival in subsequent periods.

3.3. Abandonment Rate. When there are no idle servers available, an accepted task will be forwarded to a queue and wait until all tasks in front have completed their required services. Hence, using (2) and (3), the expected queuing length L_q can be obtained as

$$L_q = \sum_{n=R}^{K} (n - R) P_n. \tag{8}$$

To find the predicted waiting time W_q in queue, we apply the well-known Little's law [15], which is a widely used formula in queuing theory. It states that the average number of items in queue is equal to the average arrival rate multiplied by the expected waiting time. Historically, it can be written as

$$L_q = \lambda W_q. \tag{9}$$

We get the expected waiting time in queue prior to service as

$$W_q = \frac{L_q}{\lambda \left(1 - P_K\right)}. \tag{10}$$

"Balking" and "reneging" are used to describe customers' abandonment behavior in queuing systems. Balking means that tasks leave a system without getting service due to long queuing length. Unlike balking case, customers always renege after facing a long waiting time in a queue. In our service system, the waiting-time information is sent to each arrival for the purpose of enhancing service quality [17, 18]. After being notified, some impatient customers may feel that waiting time is too long to endure and they will abandon the system without obtaining service. There are various reneging rules presented by previous researchers [19]. Waiting time is usually the main factor to affect customer's decisions. In addition, reneging events also depend on some potential factors. Therefore, a cloud provider will record the abandonment rate and update the latest data per planning

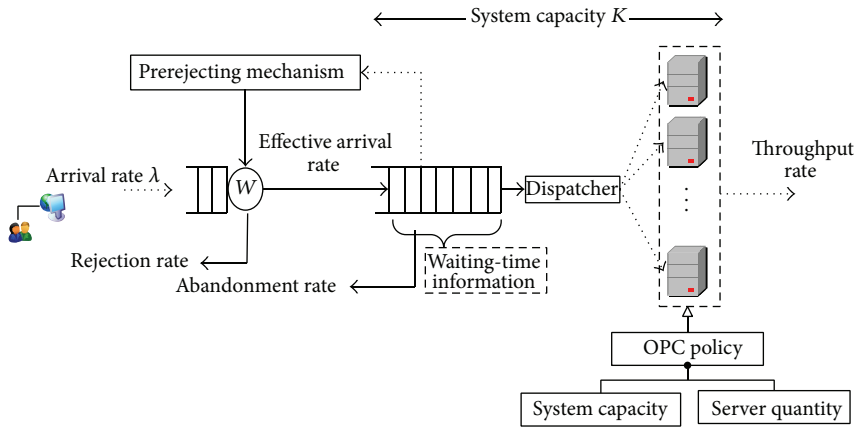

Figure 1: A designed cloud service model with the system capacity and server provisioning controls.

period in this proposed model to analyze the severity of abandonment rate. The notation $\overline{\lambda}_d$ is defined as the average abandonment rate for a certain service pattern s (while $s \in S$) at period t (while $t \in T$). Then, taking $\overline{\lambda}_d$ divided by the predicted waiting time and effective arrival rate, the potential abandonment index can be obtained as

$$d(s,t) = \frac{\overline{\lambda}_d}{W_q \times \lambda_\alpha}. \tag{11}$$

The abandonment probability can be calculated as the expected waiting time W_q multiplying by $d(s,t)$:

$$P_d = W_q \times d(s,t) = \frac{\lambda_d}{\lambda_\alpha}. \tag{12}$$

3.4. Loss Probability. Users decide to accept the cloud service with probability $1 - P_d$, or abandon this cloud service with probability P_d. Then, the expected abandonment rate λ_d can be obtained as

$$\lambda_d = \lambda_\alpha \times P_d = \lambda(1 - P_K) \times W_q \times d(s,t). \tag{13}$$

According to historical records, we obtain the mean final arrival rate λ^* that really wants to receive service as

$$\begin{aligned} \lambda^* &= \lambda_\alpha - \lambda_d = \lambda_\alpha - \lambda_\alpha P_d \\ &= \lambda_\alpha(1 - P_d) = \lambda(1 - P_K)(1 - P_d). \end{aligned} \tag{14}$$

Finally, the loss probability can be obtained as

$$\begin{aligned} \text{Loss probability} &= \frac{\lambda - \lambda^*}{\lambda} = \frac{\lambda[1 - (1 - P_K)(1 - P_d)]}{\lambda} \\ &= P_K + P_d - P_K \cdot P_d. \end{aligned} \tag{15}$$

To gain more insight into the designed system behavior and compare system performances among different capacities, several experiments are demonstrated. It is assumed that $R = 20$, $\mu = 45$, and abandonment index = 0.01, while the system capacity K is made a variable from $K = 1.5R$ to $K = 1.8R$ (ranging from 30, 32, and 34 to 36) in four steps and different arrival rates are considered. It is known that a system provided with more waiting buffer can reduce system blocking probability; however, it will result in long waiting time, as shown in Figure 2(a).

Final arrival rate distribution under various system capacities and arrival rates is shown in Figure 2(b). It is noted that final arrival rate reduces as system capacity and arrival rate increase. This is due to the fact that an arriving task is not very likely to obtain service immediately when a system is congested; instead, it has to wait in queue. In other words, the situation of final arrival rate decrease is caused by providing excessive waiting buffer with fixed servers which will directly increase the user abandonment rate. Figure 3 demonstrates the final arrival rate when the number of servers is increased to 24 as compared to Figure 2(b). It is noted that the final arrival rates increase as arrival rate increases; however, it starts to decrease as arrival rate further increases.

Results show that although final arrival rate can be increased by providing more servers, it will result in higher server provisioning cost. Therefore, comprehensive evaluations for system losses and operational cost are necessary. In the following, we compare the system loss probability in an abandonment system with a nonabandonment system. The effects of various system utilizations and system capacities on the loss probability are studied. It is assumed that system utilizations = 0.65, 0.7 0.75, and 0.8 and a cloud server farm is configured with 64 servers. The system capacity is made variable from $R + 1$ to $2R$ (ranging from 65 to 128) and $\lambda = 2500$/min. Loss probability in a nonabandonment system is shown in Figure 4(a). It is noticeable that keeping system in a higher utilization will lead to a higher loss probability, while the loss probability decreases rapidly as the system capacity further increases.

In an abandonment system, a loss probability directly depends on the system blocking loss and user abandonment loss. Figure 4(b) demonstrates the loss probability when an abandonment index is assumed value of 0.01. Results show that the loss probabilities become stable rather than decreasing rapidly (see Figure 4(a)) as the system capacity

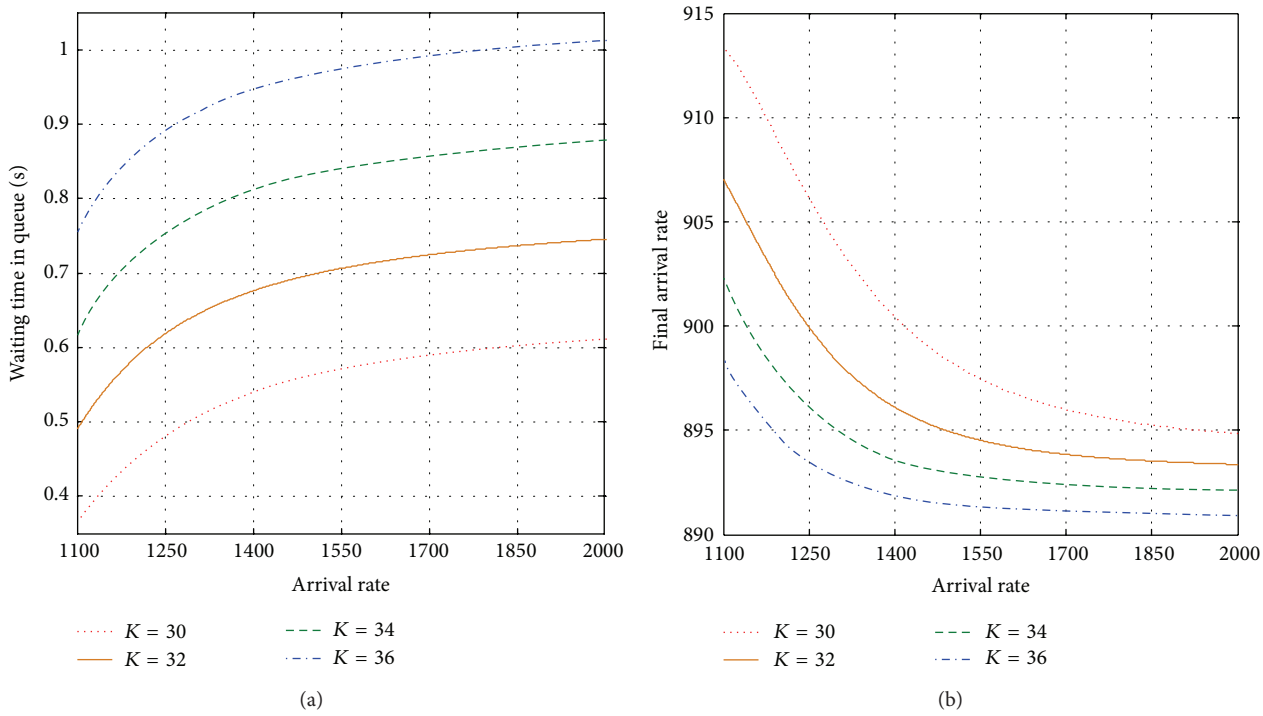

Figure 2: Waiting time and final arrival rate under various system capacities.

further increases. It seems more real as compared to a nonabandonment system since a loss probability in a service system is less likely to approach zero. Although reducing the system capacity can effectively reduce waiting time (see Figure 2(a)), it causes a higher loss probability when a system is under a higher utilization. Therefore, conducting a tradeoff analysis in a cloud system is essential to achieve a successful control for different performance levels.

4. Revenue and Cost Analysis

4.1. Revenue Function. Profit is one of the most important indicators of how well a business development is, no exception in a cloud computing industry. A value of expected revenue per task demand is denoted by $N(s,t)$ for a service pattern s ($S = 1,2,3,\ldots,s$) at period t ($T = 1,2,3,\ldots,t$). It is known that expecting more profit comes from revenue expansion, cost reduction, or both simultaneously. The total revenue is the original expected revenue minus the blocking loss and user abandonment loss, which is equivalent to the value of final arrival rate multiplying $N(s,t)$. Then, an expected total revenue $R(R,K)$ per unit time under R servers with capacity K can be written as

$$\begin{aligned} R(R,K) &= \lambda_{\alpha}\left(1-P_r\right)N(s,t) \\ &= \lambda\left(1-P_K\right)\left(1-P_r\right)N(s,t). \end{aligned} \tag{16}$$

4.2. Power Consumption Cost. Not only server provisioning but also operating power consumption is major cost burdens in a cloud system. Typically, the power consumed by a CPU is approximately proportional to a CPU frequency f and to the square of a CPU voltage V, defined by $P = a \cdot C \cdot V^2 \cdot f$, where "$C$" is the capacitance being switched per clock cycle. Since most gates do not operate at every clock cycle, they are often accompanied by an activity factor "a" [20–23]. If a processor execution rate is not given in accordance with

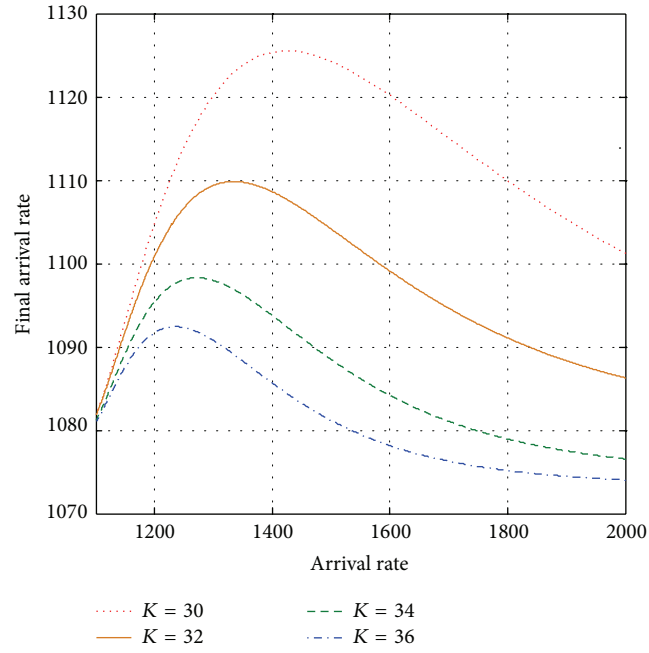

Figure 3: Final arrival rate under various system capacities when $R = 24$.

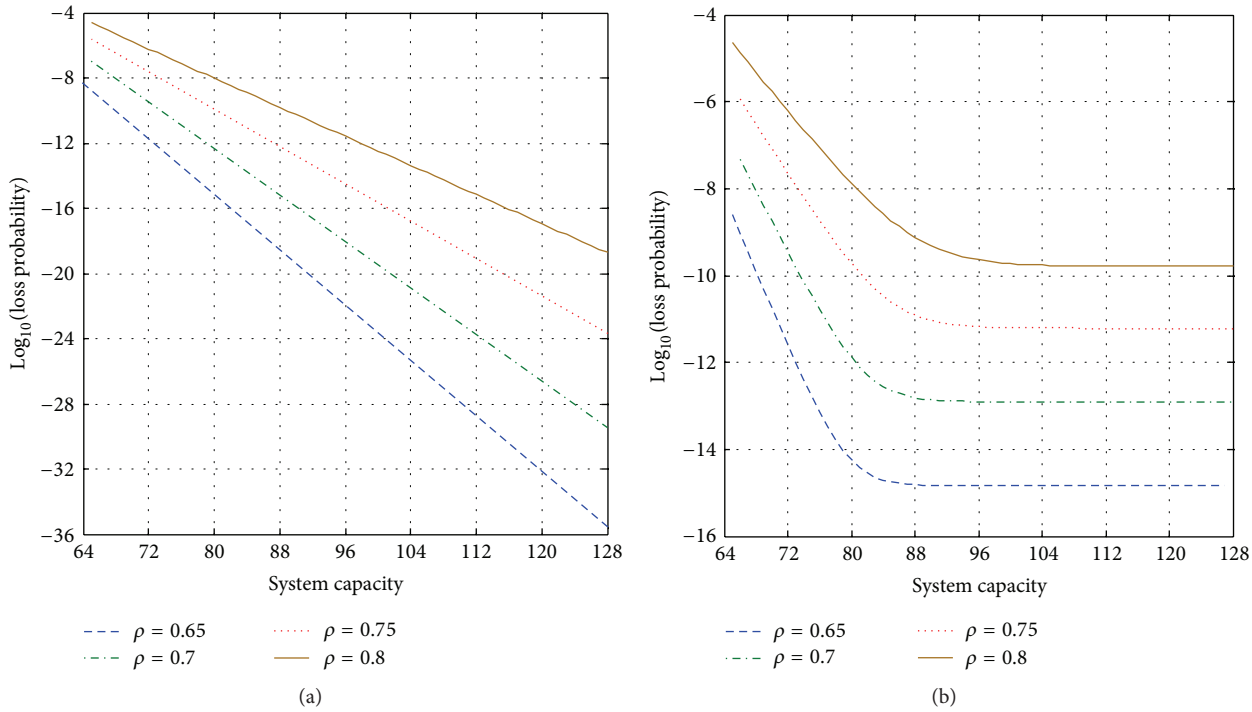

FIGURE 4: Loss probabilities (a) in a nonabandonment system and (b) in an abandonment system under various ρ and K values.

TABLE 1: Cost and system parameters.

Parameter	Description
C_R	Server provisioning cost per server per unit time
ε	Power consumption cost per Watt per unit time
μ_b	A baseline service rate
μ	A given execution rate ($\mu > \mu_b$)
μ'	An increased execution rate ($\mu - \mu_b$)
C_H	Cost incurred by holding tasks in buffer per unit time
C_W	Cost incurred by tasks waiting in buffer per unit time
C_U	Cost incurred by providing per waiting buffer space per unit time

a baseline rate μ_b, the operating cost of accelerating rate μ' needs to be calculated. It is known that the voltage and the clock frequency are related to $V \propto f^x$ $(x > 0)$ for an ideal case. Therefore, both $\mu' = zf$ and $V = yf^x$ $(x, y > 0)$ are set to facilitate the discussion of execution rate power consumption [24]. The power consumption P can be written as $P = a{\cdot}C{\cdot}V^2{\cdot}f = a{\cdot}C{\cdot}y^2{\cdot}f^{2x+1} = a{\cdot}C{\cdot}y^2{\cdot}(\mu'/z)^{2x+1} = \varphi\mu'^{\upsilon}$, where $\varphi = a{\cdot}C{\cdot}y^2/z^{2x+1}$ and $\upsilon = 2x+1$. The related notations are listed in Table 1.

Notation B is denoted by the power consumption of baseline execution rate and notation ε is denoted by the incurred cost per Watt per unit time, as presented in Table 1. Let μ denote the given execution rate $(\mu > \mu_b)$; then, the expected power consumption cost $V(R, \mu)$ per unit time under R servers with the given execution rate μ can be obtained as

$$\begin{aligned} V(R,\mu) &= \rho^{*-1}\varepsilon R\left(\varphi(\mu-\mu_b)^{\upsilon} + B\right) \\ &= \rho^{*-1}\varepsilon R\left(\varphi\mu'^{\upsilon} + B\right). \end{aligned} \tag{17}$$

Besides, it is known that a system utilization, denoted by ρ^*, is also an important factor to affect resources provisioning cost [25, 26].

4.3. System Congestion Cost. Service applications for an on-demand service pattern are typically time sensitive. Therefore, cloud providers have the responsibility to make a compensation for differentiated levels of service. As presented in Table 1, cost items that are considered in a system congestion function include C_W (incurred by tasks waiting in a queue), C_H (incurred by holding tasks in a queue), and C_U (incurred by providing per waiting buffer space). Hence, a system congestion cost, denoted by $S(R, K)$ under R servers with capacity K, can be obtained as

$$S(R,K) = C_W W^* + C_H L^* + C_U(K-R). \tag{18}$$

4.4. Profit Function. After constructing the revenue and cost functions, an expected profit function per unit time can be developed. Our objective is to determine an optimal joint value of R^* and K^* under a given execution rate μ in a cloud

server farm, so as to maximize profit. The profit maximization (PM) problem can be presented mathematically as

$$\begin{aligned} &\text{Maximize} \quad \text{PM} \\ &\text{Subject to} \quad 0 \le \mu_b < \mu \\ &\qquad\qquad \text{Loss probability} < T, \end{aligned}$$

$$\begin{aligned} \text{where PM} &= F(R,K) \\ &= R(R,K) - \rho^{*-1} R C_R - \rho^{*-1} V(R,\mu) - S(R,K) \\ &= \lambda (1 - P_K)(1 - P_d) N(s,t) - \rho^{*-1} R C_R \\ &\quad - \rho^{*-1} \varepsilon R \left(\varphi(\mu - \mu_b)^{\upsilon} + B\right) - C_W W^* \\ &\quad - C_H L^* - C_U \alpha. \end{aligned} \tag{19}$$

It is known that a loss probability is one of the major concerns for customers since no one wants to be rejected or leave because of facing long waiting time. In this work, a SLA is specified by the following relationship: loss probability $\le T$, where T is the maximum threshold value. It is extremely difficult to obtain the analytical results for the optimal value (R^*, K^*) due to the fact that this profit function is nonlinear and highly complex. Instead, the optimal profit control (OPC) algorithm is presented to find the optimal solution. For the OPC policy, satisfying the SLA constraint has the highest priority in determining the optimal solution.

5. Numerical Validation

Experiments are conducted to validate that (i) the optimal resources provisioning can be obtained by implementing the proposed heuristic algorithms and show that the OPC policy is practical and (ii) more profit gaining and significant performance improvement can be achieved by applying the OPC policy as compared to a general method, which is given only by considering a performance guarantee. Simulations are demonstrated by considering the following system parameters and cost parameters: $\varepsilon = 0.1$, $\upsilon = 2$, $\varphi = 2$, $B = 8$, $\mu_b = 40$/sec, $\mu = 45$/sec, $C_H = 60$, $C_W = 60$, $C_U = 30$, $C_R = 800$, $\lambda = 4000$/min, and $d(s,t) = 0.01$; and all computational programs are coded by MATLAB. The OPC heuristic algorithm is applied to search the optimal joint solution of the number of servers and system capacity within the loss probability guarantee of $T = 0.01$ in a SLA constraint.

Profit distribution under various number of servers and system capacities is shown in Figure 5. As can be seen, profit increases as the number of servers and system capacity increase in the beginning. However, as the server quantity or system capacity further exceeds a certain value, it will cause no more increasing in profit and begin to decrease gradually. Figure 6 shows the loss probability distribution under various number of servers and system capacities. As can be seen, the loss probability decreases obviously as the number of servers increases. The effect of server quantity on reducing loss probability is more obvious than system capacity. It is also noted that only when the number of servers is larger than 102, a system can satisfy the $T = 0.01$ constraint. Within the $T = 0.01$ constraint, the maximum profit of 1094322 and loss probability of 0.56% can be obtained at the optimal solution $(R^*, K^*) = (103, 119)$.

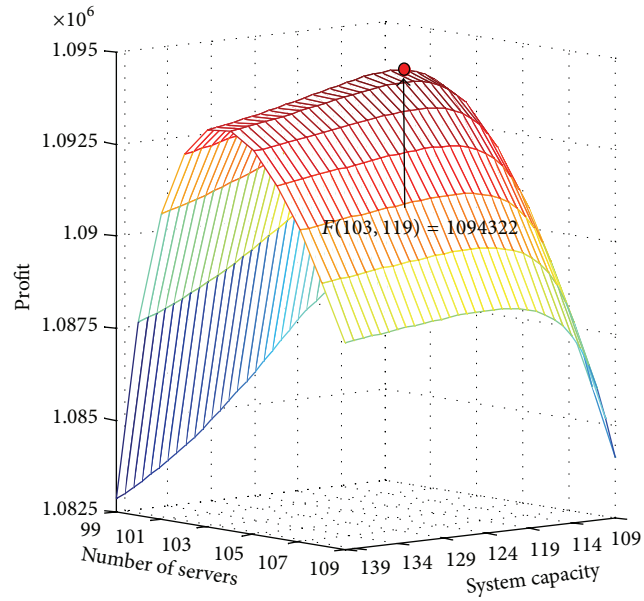

Figure 5: Profit distribution under various system capacities and the number of servers.

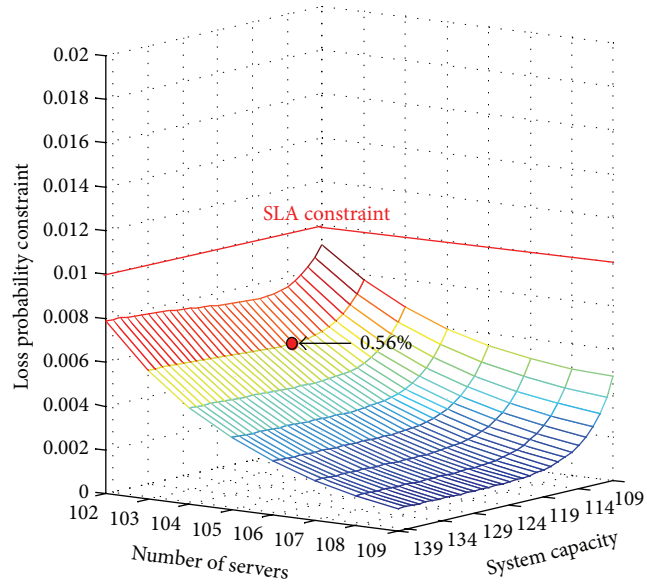

Figure 6: Loss probability constraint.

Next, the proposed OPC policy is evaluated on the basis of comparisons with a general method. It implies that a cloud resources provisioning is controlled based on a performance guarantee/threshold (in most cloud system management algorithms/methods [27–29]). For brevity, here it is referred to as a non-OPC policy since no system loss evaluation and

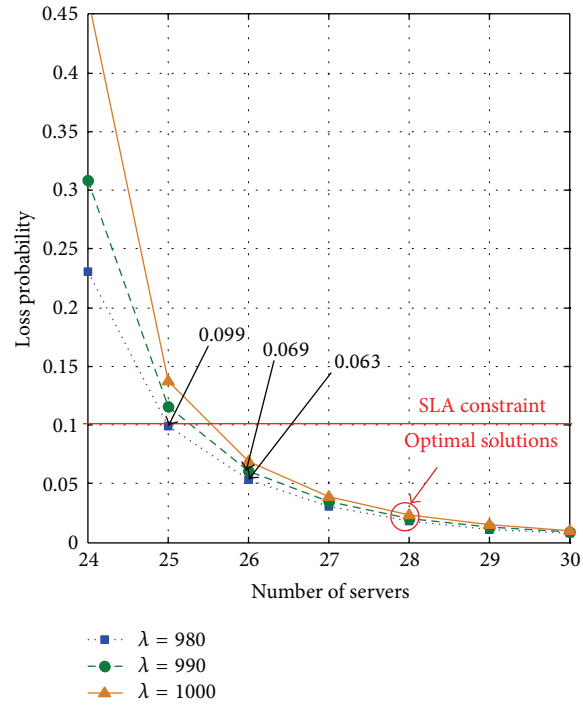

Figure 7: Loss probability comparisons between the OPC and non-OPC policies.

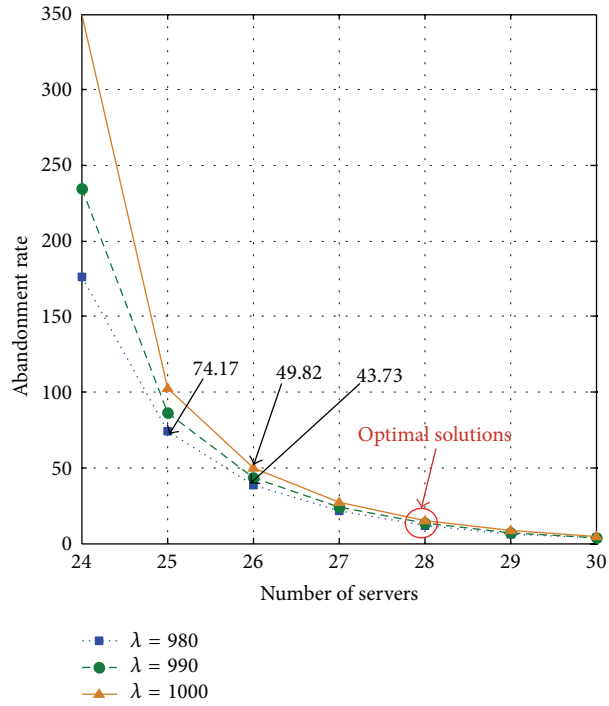

Figure 8: Abandonment rate comparisons between the OPC and non-OPC policies.

profit optimality analysis are considered. Here, simulations differ from previous investigations since we try to verify that the OPC policy is also applicable if a system is provided with a fixed capacity. A system with different arrival rates of 980, 990, and 1000 is considered and others use the same parameters as previous simulations.

A system capacity is fixed by 36 and both policies need to comply with the same constraint of $T = 0.1$. Loss probability and abandonment rate comparisons are shown in Figures 7 and 8, respectively. The solutions determined by a non-OPC policy are 0.099, 0.069, and 0.063 to meet $T = 0.1$ constraint, respectively. It is noted that the number

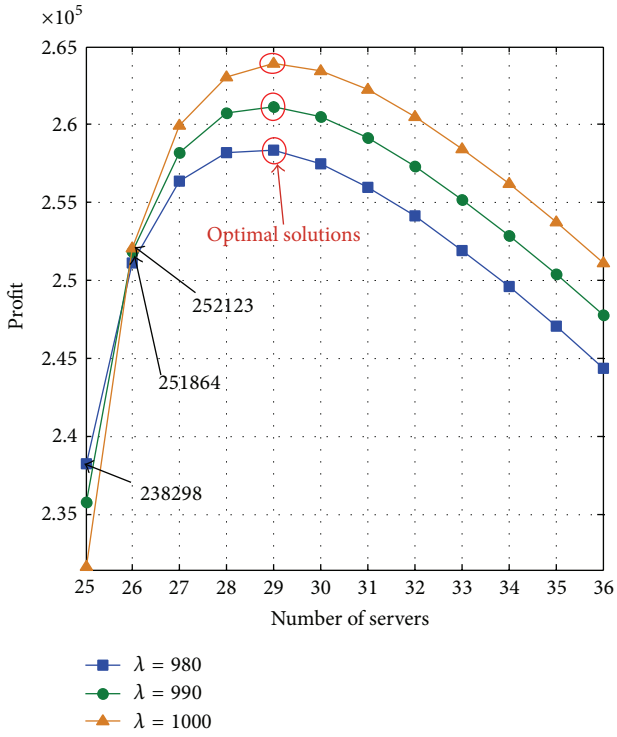

Figure 9: Comparisons of profit between the OPC and non-OPC policies.

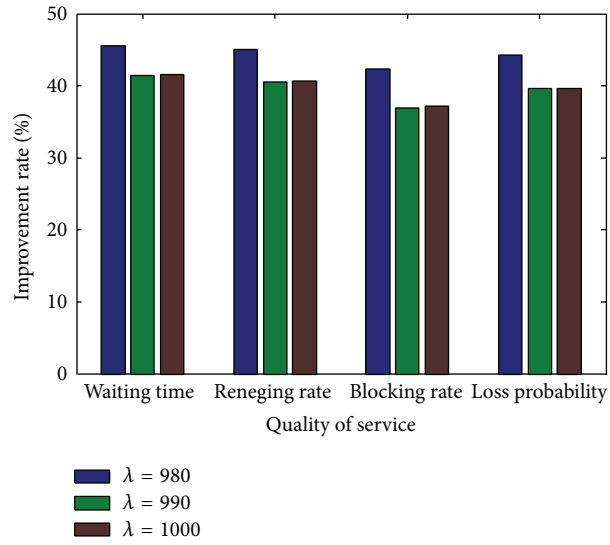

Figure 10: QoS improvement rates by applying the OPC policy.

of servers determined by a non-OPC policy is 25, 26, and 26, respectively, which is fewer than the OPC policy since the general solution is determined only based on its performance guarantee for the purpose of reducing server provisioning cost and power consumption cost. Although the number of servers controlled by the OPC policy is larger than a non-OPC policy, it can obtain lower loss probability and alleviate user abandonment rate.

The abandonment rates are 7.34, 8.29, and 9.35, respectively, for the OPC policy and 74.17, 49.82, and 43.73, respectively, for the non-OPC policy. Besides, more profit also can be achieved, as shown in Figure 9. The profit values are 258383, 261179, and 263905, respectively, for the OPC policy and 238298, 251846, and 252123, respectively, for a non-OPC policy. Finally, the QoS improvement rates are measured which calculate the relative value of improvements to an original value instead of an absolute value; the results are shown in Figure 10. As can be seen, applying the OPC policy not only can obtain more profit, but also has the potential to greatly improve various performances, including waiting time, abandonment rate, blocking rate, and loss probability.

6. Conclusions

Developing a successful cloud service system depends on accurate performance evaluations and effective system controls. Enhancing profit and simultaneously satisfying the SLA constraint have become one of the major interests for a cloud provider. In this paper, the relationship between system controls and user abandonment on the loss probability is estimated in the designed model. The effects of various system capacities and utilizations on the waiting time, final arrival rate, and system loss probability are studied. Our main goal is to maximize profit under a loss probability guarantee. Revenue is evaluated by taking system blocking loss, abandonment loss, and final arrival rate into consideration. The first proposed OPC policy combined with a heuristic algorithm allows cloud providers to effectively conduct the server quantity and the system capacity controls. It also contributes to addressing the tradeoff problem between maintaining system performances and enhancing profit. Simulation results show that the effectiveness of the OPC policy can be validated. The benefits of enhancing providers' profit and improving performances can be achieved as compared to a general method.

Notations

P_K: Blocking probability while system capacity is K
λ_α: Efficient arrival rate where tasks originally look forward to receiving service
$d(s,t)$: Potential abandonment index depended on historical records at period t (while $t \in T$) for a service type s (while $s \in S$)
P_d: Potential abandonment probability which would be expressed as a function of $d(s,t)$ and W_q
λ_d: Abandonment rate which would be expressed as a function of $d(s,t)$, W_q, and λ_α
λ^*: Expected final arrival rate
L^*: Mean final queuing length
W^*: Mean final waiting time.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proceedings of the Grid Computing Environments Workshop (GCE '08)*, pp. 1–10, November 2008.

[2] A. Chydzinski and B. Adamczyk, "Transient and stationary losses in a finite-buffer queue with batch arrivals," *Mathematical Problems in Engineering*, vol. 2012, Article ID 326830, 17 pages, 2012.

[3] B. Dragović, N.-K. Park, N. Zrnić, and R. Meštrović, "Mathematical models of multiserver queuing system for dynamic performance evaluation in port," *Mathematical Problems in Engineering*, vol. 2012, Article ID 710834, 19 pages, 2012.

[4] P. Fitzpatrick, C. S. Lee, and B. Warfield, "Teletraffic performance of mobile radio networks with hierarchical cells and overflow," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 8, pp. 1549–1557, 1997.

[5] F. E. Sandnes, "Secure distributed configuration management with randomised scheduling of system-administration tasks," *IEICE Transactions on Information and Systems*, vol. 86, no. 9, pp. 1601–1610, 2003.

[6] A. P. Ghosh and A. P. Weerasinghe, "Optimal buffer size and dynamic rate control for a queueing system with impatient customers in heavy traffic," *Stochastic Processes and Their Applications*, vol. 120, no. 11, pp. 2103–2141, 2010.

[7] D. Doran, L. Lipsky, and S. Thompson, "Cost-based optimization of buffer size in M/G/1/N systems under different service-time distributions," in *Proceedings of the 9th IEEE International Symposium on Network Computing and Applications (NCA '10)*, pp. 28–35, July 2010.

[8] H. Khazaei, J. Misic, and V. B. Misic, "Performance analysis of cloud computing centers using M/G/m/m+r queuing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp. 936–943, 2012.

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

[10] B. Yang, F. Tan, Y. Dai, and S. Guo, "Performance evaluation of cloud service considering fault recovery," in *Proceedings of the 1st International Conference on Cloud Computing (CloudCom '09)*, pp. 571–576, 2009.

[11] Í. Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," in *Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD '10)*, pp. 123–130, July 2010.

[12] A. N. Toosi, R. N. Calheiros, R. K. Thulasiram, and R. Buyya, "Resource provisioning policies to increase IaaS provider's profit in a federated cloud environment," in *Proceedings of the 13th IEEE International Conference on High Performance Computing and Communications (HPCC '11)*, pp. 279–287, September 2011.

[13] J. Cao, K. Hwang, K. Li, and A. Y. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1087–1096, 2013.

[14] H. Goudarzi and M. Pedram, "Maximizing profit in cloud computing system via resource allocation," in *Proceedings of the 31st International Conference on Distributed Computing Systems Workshops (ICDCSW '11)*, pp. 1–6, June 2011.

[15] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, Wiley Series in Probability and Statistics, John Wiley & Sons, Hoboken, NJ, USA, 4th edition, 2008.

[16] Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall, and L. Stougie, "Multiprocessor scheduling with rejection," *SIAM Journal on Discrete Mathematics*, vol. 13, no. 1, pp. 64–78, 2000.

[17] P. Afèche and H. Mendelson, "Pricing and priority auctions in queueing systems with a generalized delay cost structure," *Management Science*, vol. 50, no. 7, pp. 869–882, 2004.

[18] P. Guo and P. Zipkin, "Analysis and comparison of queues with different levels of delay information," *Management Science*, vol. 53, no. 6, pp. 962–970, 2007.

[19] J. Ou and B. M. Rao, "Benefits of providing amenities to impatient waiting customers," *Computers & Operations Research*, vol. 30, no. 14, pp. 2211–2225, 2003.

[20] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, 1992.

[21] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *Proceedings of the 41st Design Automation Conference*, pp. 868–873, June 2004.

[22] CPU Power Dissipation, http://en.wikipedia.org/wiki/CPU_power_dissipation.

[23] Central Processing Unit, http://en.wikipedia.org/wiki/Central_processing_unit.

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

[25] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker, "Green cloud computing: balancing energy in processing, storage, and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2010.

[26] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, 2012.

[27] L. Mao, Y. Yang, H. Xu, and Y. Chen, "Service selection algorithm based on constraint for cloud workflow system," *Journal of Software*, vol. 8, no. 5, pp. 1124–1131, 2013.

[28] G. Le, K. Xu, and J. Song, "Dynamic resource provisioning and scheduling with deadline constraint in elastic cloud," in *Proceedings of the International Conference on Service Science (ICSS '13)*, pp. 113–117, April 2013.

[29] B. Li, A. M. Song, and J. Song, "A distributed QoS-constraint task scheduling scheme in cloud computing environment: model and algorithm," *Advances in Information Sciences and Service Sciences*, vol. 4, no. 5, pp. 283–291, 2012.