Optimal Multiserver Configuration for Profit Maximization in Cloud Computing

Junwei Cao, *Senior Member, IEEE* Kai Hwang, *Fellow, IEEE* Keqin Li, *Senior Member, IEEE*
Albert Y. Zomaya, *Fellow, IEEE*

Abstract—As cloud computing becomes more and more popular, understanding the economics of cloud computing becomes critically important. To maximize the profit, a service provider should understand both service charges and business costs, and how they are determined by the characteristics of the applications and the configuration of a multiserver system. The problem of optimal multiserver configuration for profit maximization in a cloud computing environment is studied. Our pricing model takes such factors into considerations as the amount of a service, the workload of an application environment, the configuration of a multiserver system, the service level agreement, the satisfaction of a consumer, the quality of a service, the penalty of a low quality service, the cost of renting, the cost of energy consumption, and a service provider's margin and profit. Our approach is to treat a multiserver system as an M/M/m queueing model, such that our optimization problem can be formulated and solved analytically. Two server speed and power consumption models are considered, namely, the idle-speed model and the constant-speed model. The probability density function of the waiting time of a newly arrived service request is derived. The expected service charge to a service request is calculated. The expected net business gain in one unit of time is obtained. Numerical calculations of the optimal server size and the optimal server speed are demonstrated.

Index Terms—Cloud computing, multiserver system, pricing model, profit, queueing model, response time, server configuration, service charge, service level agreement, waiting time.

1 Introduction

CLOUD computing is quickly becoming an effective and efficient way of computing resources and computing services consolidation [9]. By centralized management of resources and services, cloud computing delivers hosted services over the Internet, such that accesses to shared hardware, software, databases, information, and all resources are provided to consumers on-demand. Cloud computing is able to provide the most cost-effective and energy-efficient way of computing resources management and computing services provision. Cloud computing turns information technology into ordinary commodities and utilities by using the pay-per-use pricing model [3], [5], [16]. However, cloud computing will never be free [8], and understanding the economics of cloud computing becomes critically important.

One attractive cloud computing environment is a three-tier structure [13], which consists of infrastructure vendors, service providers, and consumers. The three parties are also called cluster nodes, cluster managers, and consumers in cluster computing systems [19], and resource providers, service providers, and clients in grid computing systems [17]. An infrastructure vendor maintains basic hardware and software facilities. A service provider rents resources from the infrastructure vendors, builds appropriate multiserver systems, and provides various services to users. A consumer submits a service request to a service provider, 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 quality of the service. A service provider can build different multiserver systems for different applications domains, such that service requests of different nature are sent to different multiserver systems. Each multiserver system contains multiple servers, and such a multiserver system can be devoted to serve one type of service requests and applications. The configuration of a multiserver system is characterized by two basic features, i.e., the size of the multiserver system (the number of servers) and the speed of the multiserver system (execution speed of the servers).

Like all business, the pricing model of a service provider in cloud computing is based on two components, namely, the income and the cost. For a

- *J. Cao is with the Research Institute of Information Technology, Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China.
E-mail: jcao@tsinghua.edu.cn*
- *K. Hwang is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA.
E-mail: kaihwang@usc.edu*
- *K. Li is with the Department of Computer Science, State University of New York, New Paltz, New York 12561, USA.
E-mail: lik@newpaltz.edu*

This is the author for correspondence.
- *A. Y. Zomaya is with the School of Information Technologies, University of Sydney, Sydney, NSW 2006, Australia.
E-mail: albert.zomaya@sydney.edu.au*

service provider, the income (i.e., the revenue) is the service charge to users, and the cost is the renting cost plus the utility cost paid to infrastructure vendors. A pricing model in cloud computing includes many considerations, such as the amount of a service (the requirement of a service), the workload of an application environment, the configuration (the size and the speed) of a multiserver system, the service level agreement, the satisfaction of a consumer (the expected service time), the quality of a service (the task waiting time and the task response time), the penalty of a low quality service, the cost of renting, the cost of energy consumption, and a service provider's margin and profit. The profit (i.e., the net business gain) is the income minus the cost. To maximize the profit, a service provider should understand both service charges and business costs, and in particular, how they are determined by the characteristics of the applications and the configuration of a multiserver system.

The service charge to a service request is determined by two factors, i.e., the expected length of the service and the actual length of the service. The expected length of a service (i.e., the expected service time) is the execution time of an application on a standard server with a baseline or reference speed. Once the baseline speed is set, the expected length of a service is determined by a service request itself, i.e., the service requirement (amount of service) measured by the number of instructions to be executed. The longer (shorter, respectively) the expected length of a service is, the more (less, respectively) the service charge is. The actual length of a service (i.e., the actual service time) is the actual execution time of an application. The actual length of a service depends on the size of a multiserver system, the speed of the servers (which may be faster or slower than the baseline speed), and the workload of the multiserver system. Notice that the actual service time is a random variable, which is determined by the task waiting time once a multiserver system is established.

There are many different service performance metrics in service level agreements [2]. Our performance metric in this paper is the task response time (or the turn around time), i.e., the time taken to complete a task, which includes task waiting time and task execution time. The service level agreement is the promised time to complete a service, which is a constant times the expected length of a service. If the actual length of a service is (or, a service request is completed) within the service level agreement, the service will be fully charged. However, if the actual length of a service exceeds the service level agreement, the service charge will be reduced. The longer (shorter, respectively) the actual length of a service is, the more (less, respectively) the reduction of the service charge is. In other words, there is penalty for a service provider to break a service level agreement. If the actual service time exceeds certain limit (which is service request dependent), a service will be entirely free with no charge. Notice that the service charge of a service request is a random variable, and we are interested in its expectation.

The cost of a service provider includes two components, i.e., the renting cost and the utility cost. The renting cost is proportional to the size of a multiserver system, i.e., the number of servers. The utility cost is essentially the cost of energy consumption and is determined by both the size and the speed of a multiserver system. The faster (slower, respectively) the speed is, the more (less, respectively) the utility cost is. To calculate the cost of energy consumption, we need to establish certain server speed and power consumption models.

To increase the revenue of business, a service provider can construct and configure a multiserver system with many servers of high speed. Since the actual service time (i.e., the task response time) contains task waiting time and task execution time, more servers reduce the waiting time and faster servers reduce both waiting time and execution time. Hence, a powerful multiserver system reduces the penalty of breaking a service level agreement and increases the revenue. However, more servers (i.e., a larger multiserver system) increase the cost of facility renting from the infrastructure vendors and the cost of base power consumption. Furthermore, faster servers increase the cost of energy consumption. Such increased cost may counterweight the gain from penalty reduction. Therefore, for an application environment with specific workload which includes the task arrival rate and the average task execution requirement, a service provider needs to decide an optimal multiserver configuration (i.e, the size and the speed of a multiserver system), such that the expected profit is maximized.

In this paper, we study the problem of optimal multiserver configuration for profit maximization in a cloud computing environment. Our approach is to treat a multiserver system as an M/M/m queueing model, such that our optimization problem can be formulated and solved analytically. We consider two server speed and power consumption models, namely, the idle-speed model and the constant-speed model. Our main contributions are as follows. We derive the probability density function of the waiting time of a newly arrived service request. This result is significant in its own right and is the base of our discussion. We calculate the expected service charge to a service request. Based on these results, we get the expected net business gain in one unit of time, and obtain the optimal server size and the optimal server speed numerically. To the best of our knowledge, there has been no similar investigation in the literature.

One related research is user-centric and market-based and utility-driven resource management and task scheduling, which have been considered for cluster computing systems [7], [18], [19] and grid computing systems [4], [11], [17]. To compete and bid for shared computing resources through the use of economic mechanisms such as auctions, a user can specify the value (utility, yield) of a task, i.e., the reward (price, profit) of completing the task. A utility function, which measures the value and importance of a task as well as a user's tolerance to delay and sensitivity to quality of service, supports market-based bidding, negotiation, and admission control. By taking an economic approach to providing service-oriented and utility computing, a service provider allocates resources and schedules tasks in such a way that the total profit earned is maximized. Instead of traditional system-centric performance optimization such as minimizing the average task response time, the main concern in such computational economy is user-centric performance optimization, i.e., maximizing the total utility delivered to the users (i.e., the total user-perceived value).

The rest of the paper is organized as follows. In Section 2, we describe our queueing model for multiserver systems. In Section 3, we present our server speed and power consumption models. In Section 4, we derive the probability density function of the waiting time of a newly arrived service request. In Section 5, we define a service charge function and calculate the expected service charge to a service request. In Section 6, we obtain the expected net business gain in one unit of time. In Section 7, we show how to obtain the optimal server size and the optimal server speed numerically. In Section 8, we demonstrate simulation data to validate our analytical results and to find more effective queueing disciplines. We conclude the paper in Section 9.

2 A Multiserver Model

Throughout the paper, we use $\boldsymbol{P}[e]$ to denote the probability of an event e. For a random variable x, we use $f_x(t)$ to represent the probability density function (pdf) of x, and $F_x(t)$ to represent the cumulative distribution function (cdf) of x, and $\bar{x}$ to represent the expectation of x.

A cloud computing service provider serves users' service requests by using a multiserver system, which is constructed and maintained by an infrastructure vendor and rented by the service provider. The architecture detail of the multiserver system can be quite flexible. Examples are blade servers and blade centers where each server is a server blade [14], clusters of traditional servers where each server is an ordinary processor [7], [18], [19], and multicore server processors where each server is a single core [15]. We will simply call these blades/processors/cores as *servers*. Users (i.e., customers of a service provider) submit service requests (i.e., applications and tasks) to a service provider, and the service provider serves the requests (i.e., run the applications and perform the tasks) on a multiserver system.

Assume that a multiserver system S has m identical servers. In this paper, a multiserver system is treated as an M/M/m queueing system which is elaborated as follows. 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 S maintains a queue with infinite capacity for waiting tasks when all the m servers are busy. The first-come-first-served (FCFS) queueing discipline is adopted. The task execution requirements (measured by the number of instructions to be executed) are i.i.d. exponential random variables r with mean $\bar{r}$. The m servers (i.e., blades/processors/cores) of S have identical execution speed s (measured by the number of instructions that can be executed in one unit of time). Hence, the task execution times on the servers of S are i.i.d. exponential random variables $x = r/s$ with mean $\bar{x} = \bar{r}/s$.

Let $\mu = 1/\bar{x} = s/\bar{r}$ be the average service rate, i.e., the average number of service requests that can be finished by a server of S in one unit of time. The server utilization is

$$\rho = \frac{\lambda}{m\mu} = \frac{\lambda\bar{x}}{m} = \frac{\lambda}{m} \cdot \frac{\bar{r}}{s},$$

which is the average percentage of time that a server of S is busy. Let p_k denote the probability that there are k service requests (waiting or being processed) in the M/M/m queueing system for S. Then, we have ([12], p. 102)

$$p_k = \begin{cases} p_0 \dfrac{(m\rho)^k}{k!}, & k \le m; \\[2ex] p_0 \dfrac{m^m \rho^k}{m!}, & k \ge m; \end{cases}$$

where

$$p_0 = \left(\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!} \cdot \frac{1}{1-\rho} \right)^{-1}.$$

The probability of queueing (i.e., the probability that a newly submitted service request must wait because all servers are busy) is

$$P_q = \sum_{k=m}^{\infty} p_k = \frac{p_m}{1-\rho} = p_0 \frac{(m\rho)^m}{m!} \cdot \frac{1}{1-\rho}.$$

The average number of service requests (in waiting or in execution) in S is

$$\overline{N} = \sum_{k=0}^{\infty} k p_k = m\rho + \frac{\rho}{1-\rho} P_q.$$

Applying Little's result, we get the average task response time as

$$\overline{T} = \frac{\overline{N}}{\lambda} = \bar{x}\left(1 + \frac{P_q}{m(1-\rho)}\right) = \bar{x}\left(1 + \frac{p_m}{m(1-\rho)^2}\right).$$

The average waiting time of a service request is

$$\overline{W} = \overline{T} - \bar{x} = \frac{p_m}{m(1-\rho)^2}\bar{x}.$$

The waiting time is the source of customer dissatisfaction. A service provider should keep the waiting time to a low level by providing enough servers and/or increasing server speed, and be willing to pay back to a customer in case the waiting time exceeds certain limit.

3 Power Consumption Models

Power dissipation and circuit delay in digital CMOS circuits can be accurately modeled by simple equations, even for complex microprocessor circuits. CMOS circuits have dynamic, static, and short-circuit power dissipation; however, the dominant component in a well designed circuit is dynamic power consumption p (i.e., the switching component of power), which is approximately $P = aCV^2f$, where a is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency [6]. In the ideal case, the supply voltage and the clock frequency are related in such a way that $V \propto f^{\phi}$ for some constant $\phi > 0$ [20]. The processor execution speed s is usually linearly proportional to the clock frequency, namely, $s \propto f$. For ease of discussion, we will assume that $V = bf^{\phi}$ and $s = cf$, where b and c are some constants. Hence, we know that power consumption is $P = aCV^2f = ab^2Cf^{2\phi+1} = (ab^2C/c^{2\phi+1})s^{2\phi+1} = \xi s^{\alpha}$, where $\xi = ab^2C/c^{2\phi+1}$ and $\alpha = 2\phi + 1$. For instance, by setting $b = 1.16$, $aC = 7.0$, $c = 1.0$, $\phi = 0.5$, $\alpha = 2\phi+1 = 2.0$, and $\xi = ab^2C/c^{\alpha} = 9.4192$, the value of P calculated by the equation $P = aCV^2f = \xi s^{\alpha}$ is reasonably close to that in [10] for the Intel Pentium M processor.

We will consider two types of server speed and power consumption models. In the *idle-speed model*, a server runs at zero speed when there is no task to perform. Since the power for speed s is ξs^{α}, the average amount of energy consumed by a server in one unit of time is

$$\rho\xi s^{\alpha} = \frac{\lambda}{m}\bar{r}\xi s^{\alpha-1},$$

where we notice that the speed of a server is zero when it is idle. The average amount of energy consumed by an m-server system S in one unit of time, i.e., the power supply to the multiserver system S, is

$$P = m\rho\xi s^{\alpha} = \lambda\bar{r}\xi s^{\alpha-1},$$

where $m\rho = \lambda\bar{x}$ is the average number of busy servers in S. Since a server still consumes some amount of power P^* even when it is idle (assume that an idle server consumes certain base power P^*, which includes static power dissipation, short circuit power dissipation, and other leakage and wasted power [1]), we will include P^* in P, i.e.,

$$P = m(\rho\xi s^{\alpha} + P^*) = \lambda\bar{r}\xi s^{\alpha-1} + mP^*.$$

Notice that when $P^* = 0$, the above P is independent of m.

In the *constant-speed model*, all servers run at the speed s even if there is no task to perform. Again, we use P to represent the power allocated to multiserver system S. Since the power for speed s is ξs^{α}, the power allocated to multiserver system S is $P = m(\xi s^{\alpha} + P^*)$.

4 Waiting Time Distribution

Let W denote the waiting time of a new service request that arrives to a multiserver system. In this section, we find the pdf $f_W(t)$ of W. To this end, we consider W in different situations, depending on the number of tasks in the queueing system when a new service request arrives. Let W_k denote the waiting time of a new task that arrives to an M/M/m queueing system under the condition that there are k tasks in the queueing system when the task arrives.

We define a *unit impulse function* $u_z(t)$ as follows:

$$u_z(t) = \begin{cases} z, & 0 \le t \le \dfrac{1}{z}; \\ 0, & t > \dfrac{1}{z}. \end{cases}$$

The function $u_z(t)$ has the following property,

$$\int_0^{\infty} u_z(t)dt = 1,$$

namely, $u_z(t)$ can be treated as a pdf of a random variable with expectation

$$\int_0^{\infty} tu_z(t)dt = z\int_0^{1/z} tdt = \frac{1}{2z}.$$

Let $z \to \infty$ and define

$$u(t) = \lim_{z\to\infty} u_z(t).$$

It is clear that any random variable whose pdf is $u(t)$ has expectation 0.

The following theorem gives the pdf of the waiting time of a newly arrived service request.

Theorem 1: The pdf of the waiting time W of a newly arrived service request is

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

where $P_q = p_m/(1-\rho)$ and $p_m = p_0(m\rho)^m/m!$.

Sketch of the Proof. Let W_k be the waiting time of a new service request if there are k tasks in the queueing system when the service request arrives. We find the pdf of W_k for all $k \ge 0$. Then, we have

$$f_W(t) = \sum_{k=0}^{\infty} p_k f_{W_k}(t).$$

Actually, W_k can be found for two cases, i.e., when $k < m$ and when $k \geq m$. A complete proof of the theorem is given in the appendix. ∎

5 Service Charge

If all the servers have a fixed speed s, the execution time of a service request with execution requirement r is known as $x = r/s$. The response time to the service request is

$$T = W + x = W + \frac{r}{s}.$$

(Note: The above equation implies that the pdf of T is simply the convolution of the pdf's of W and x. However, we will not pursue this direction to avoid unnecessary mathematical complication. In fact, we will only use the pdf of W.) The response time T is related to the service charge to a customer of a service provider in cloud computing.

To study the expected service charge to a customer, we need a complete specification of a service charge based on the amount of a service, the service level agreement, the satisfaction of a consumer, the quality of a service, the penalty of a low quality service, and a service provider's margin and profit.

Let s_0 be the baseline speed of a server. We define the *service charge function* for a service request with execution requirement r and response time T to be

$$C(r,T) = \begin{cases} ar, & \text{if } 0 \leq T \leq \dfrac{c}{s_0} r; \\[2ex] ar - d\left(T - \dfrac{c}{s_0} r\right), & \\ \qquad \text{if } \dfrac{c}{s_0} r < T \leq \left(\dfrac{a}{d} + \dfrac{c}{s_0}\right) r; & \\[2ex] 0, & \text{if } T > \left(\dfrac{a}{d} + \dfrac{c}{s_0}\right) r. \end{cases}$$

The above function is defined with the following rationals.

- If the response time T to process a service request is no longer than $(c/s_0)r = c(r/s_0)$ (i.e., a constant c times the task execution time with speed s_0), where the constant c is a parameter indicating the service level agreement, and the constant s_0 is a parameter indicating the expectation and satisfaction of a consumer, then a service provider considers that the service request is processed successfully with high quality of service and charges a customer ar, which is linearly proportional to the task execution requirement r (i.e., the amount of service), where a is the service charge per unit amount of service (i.e., a service provider's margin and profit).
- If the response time T to process a service request is longer than $(c/s_0)r$ but no longer than $(a/d + c/s_0)r$, then a service provider considers that the service request is processed with low quality of service and the charge to a customer should decrease linearly as T increases. The parameter d indicates the degree of penalty of breaking the service level agreement.
- If the response time T to process a service request is longer than $(a/d + c/s_0)r$, then a service provider considers that the service request has been waiting too long, so there is no charge and the service is free.

Notice that the task response time T is compared with the task execution time on a server with speed s_0 (i.e., the baseline or reference speed). The actual speed s of a server can be decided by a service provider, which can be either lower or higher than s_0, depending on the workload (i.e., λ and $\bar{r}$) and system parameters (e.g., m, α, and P^*) and the service charge function (i.e., a, c, and d), such that the net business gain to be defined below is maximized.

To build our discussion upon our earlier result on task waiting time, we notice that the service charge function can be rewritten equivalently in terms of r and W as

$$C(r,W) = \begin{cases} ar, & \text{if } 0 \leq W \leq \left(\dfrac{c}{s_0} - \dfrac{1}{s}\right) r; \\[2ex] \left(a + \dfrac{cd}{s_0} - \dfrac{d}{s}\right) r - dW, & \\ \qquad \text{if } \left(\dfrac{c}{s_0} - \dfrac{1}{s}\right) r < W \leq \left(\dfrac{a}{d} + \dfrac{c}{s_0} - \dfrac{1}{s}\right) r; & \\[2ex] 0, & \text{if } W > \left(\dfrac{a}{d} + \dfrac{c}{s_0} - \dfrac{1}{s}\right) r. \end{cases}$$

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

Theorem 2: The expected charge to a service request is

$$C = a\bar{r}\left(1 - \frac{P_q}{((ms - \lambda\bar{r})(c/s_0 - 1/s) + 1)} \times \frac{1}{((ms - \lambda\bar{r})(a/d + c/s_0 - 1/s) + 1)}\right),$$

where $P_q = p_m/(1-\rho)$ and $p_m = p_0(m\rho)^m/m!$.

Sketch of the Proof. The proof is actually a detailed calculation of C, which contains two steps. In the first step, we calculate $C(r)$, i.e., the expected charge to a service request with execution requirement r, based on the pdf of W obtained from Theorem 1. In the second step, we calculate C based on the pdf of r. A complete proof of the theorem is given in the appendix. ∎

In Figure 1, we consider the expected charge to a service request with execution requirement r, i.e.,

$$C(r) = ar - \frac{dP_q}{(1-\rho)m\mu}\left(e^{-(1-\rho)m\mu(c/s_0 - 1/s)r} - e^{-(1-\rho)m\mu(a/d + c/s_0 - 1/s)r}\right).$$

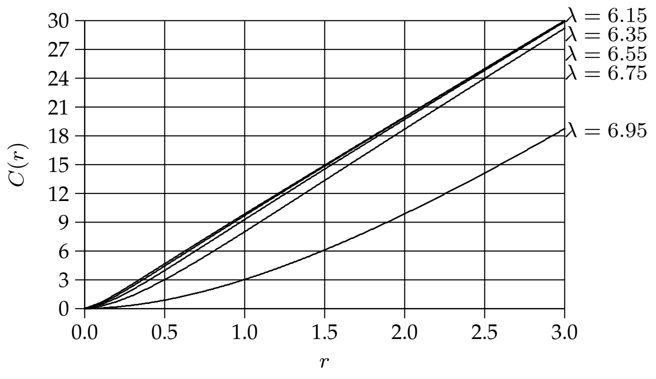

Fig. 1. Service charge $C(r)$ vs. r and λ.

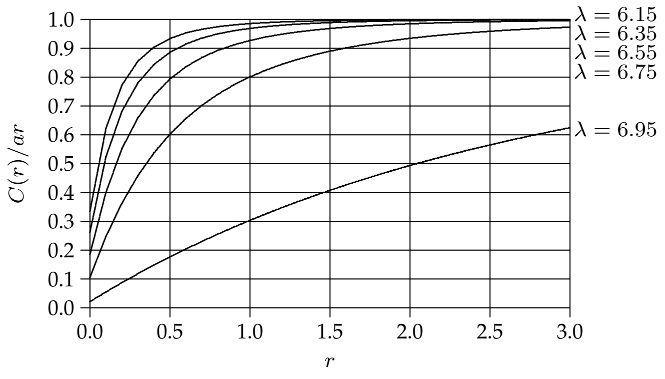

Fig. 2. Normalized service charge $C(r)/ar$ vs. r and λ.

We assume that $\bar{r} = 1$ billion instructions, $m = 7$ servers, $s_0 = 1$ billion instructions per second, $s = 1$ billion instructions per second, $a = 10$ cents per one billion instructions (Note: The monetary unit "cent" in this paper may not be identical but should be linearly proportional to the real cent in US dollars.), $c = 3$, and $d = 1$ cents per second. For $\lambda = 6.15, 6.35, 6.55, 6.75, 6.95$ service requests per second, we show $C(r)$ for $0 \le r \le 3$. It can be seen that the service charge is a decreasing function of λ, since the waiting time and lateness penalty increase as λ increases. It can also be seen that the service charge is an increasing function of r, i.e., large service requests generate more revenue than small service requests.

In Figure 2, we further display $C(r)/ar$ using the same parameters in Figure 1. Since ar is the ideal (maximum) charge to a service request with execution requirement r, $C(r)/ar$ is considered as the *normalized service charge*. For $\lambda = 6.15, 6.35, 6.55, 6.75, 6.95$ service requests per second, we show $C(r)/ar$ for $0 \le r \le 3$. It can be seen that the normalized service charge is a decreasing function of λ, since the waiting time and lateness penalty increase as λ increases. It can also be seen that the normalized service charge is an increasing function of r, i.e., the percentage of lost service charge due to waiting time decreases as service requirement r increases. In other words, it is more likely to make profit from large service requests and it is more likely to give free services to small service requests. It can be verified that as r approaches 0, the normalized service charge is

$$\lim_{r \to 0} \frac{C(r)}{ar} = 1 - P_q,$$

where P_q increases (and $1 - P_q$ decreases) as λ increases. It can also be verified that as r approaches infinity, the normalized service charge is

$$\lim_{r \to \infty} \frac{C(r)}{ar} = 1,$$

for all λ.

6 Net Business Gain

Since the number of service requests processed in one unit of time is λ in a stable M/M/m queueing system, the expected service charge in one unit of time is λC, which is actually the expected *revenue* of a service provider. Assume that the rental cost of one server for unit of time is β. Also, assume that the cost of energy is γ per Watt. The *cost* of a service provider is the sum of the cost of infrastructure renting and the cost of energy consumption, i.e., $\beta m + \gamma P$. Then, the expected *net business gain* (i.e., the net profit) of a service provider in one unit of time is

$$G = \lambda C - (\beta m + \gamma P),$$

which is defined as the revenue minus the cost. The above equation is

$$G = \lambda C - (\beta m + \gamma(\lambda \bar{r} \xi s^{\alpha-1} + mP^*)),$$

for the idle-speed model, and

$$G = \lambda C - (\beta m + \gamma m(\xi s^{\alpha} + P^*)),$$

for the constant-speed model.

In Figures 3 and 4, we demonstrate the revenue λC and the net business gain G in one unit of time as a function of λ for the two power consumption models respectively, using the same parameters in Figures 1 and 2. Furthermore, we assume that $P^* = 2$ Watts, $\alpha = 2.0$, $\xi = 9.4192$, $\beta = 1.5$ cents per second, and $\gamma = 0.1$ cents per Watt. For $0 \le \lambda \le 7$, we show λC and G. The cost of infrastructure renting is $\beta m = 14$ cents per second, and the cost of energy consumption is $0.5\lambda + 7$ cents per second for the idle-speed model and 10.5 cents per second for the constant-speed model. We observe that both λC and G increase with λ almost linearly and drop sharply after certain point. In other words, more service requests bring more revenue and net business gain; however, after the number of service requests per unit of time reaches certain point, the excessive waiting time causes increased lateness penalty, so that there is no revenue and negative business gain.

There are two situations that cause negative business gain. In the first case, there is no enough business

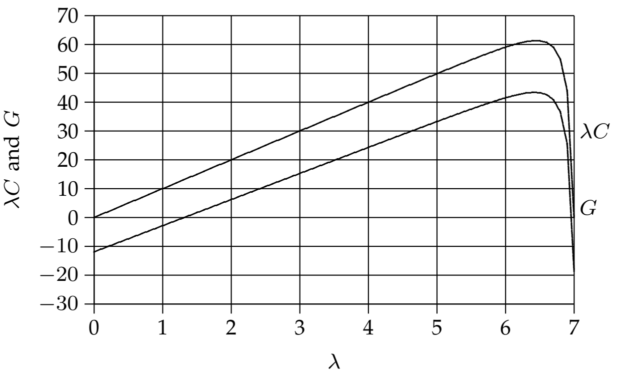

Fig. 3. Revenue and net business gain vs. λ (idle-speed model).

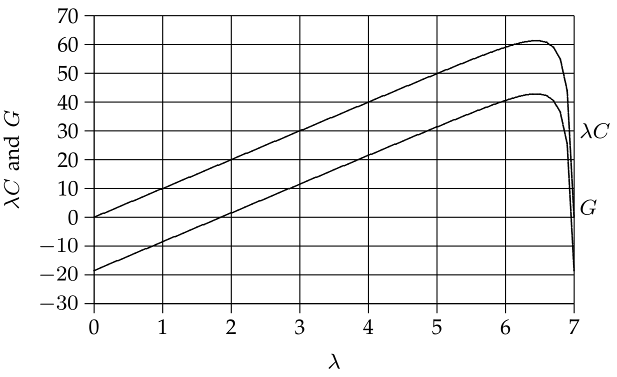

Fig. 4. Revenue and net business gain vs. λ (constant-speed model).

(i.e., service requests). In this case, a service provider should consider reducing the number of servers m and/or server speed s, so that the cost of infrastructure renting and the cost of energy consumption can be reduced. In the second case, there is too much business (i.e., service requests). In this case, a service provider should consider increasing the number of servers and/or server speed, so that the waiting time can be reduced and the revenue can be increased. However, increasing the number of servers and/or server speed also increases the cost of infrastructure renting and the cost of energy consumption. Therefore, we have the problem of selecting the optimal server size and/or server speed so that the profit is maximized.

7 Profit Maximization

To formulate and solve our optimization problems analytically, we need a closed-form expression of C. To this end, let us use the following closed-form approximation,

$$\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} \approx e^{m\rho},$$

which is very accurate when m is not too small and ρ is not too large [15]. We also need Stirling's approximation of $m!$, i.e.,

$$m! \approx \sqrt{2\pi m}\left(\frac{m}{e}\right)^m.$$

Therefore, we get the following closed-form approximation of p_0,

$$p_0 \approx \left(e^{m\rho} + \frac{(e\rho)^m}{\sqrt{2\pi m}} \cdot \frac{1}{1-\rho}\right)^{-1},$$

and the following closed-form approximation of p_m,

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

namely,

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

and the following closed-form approximation of P_q,

$$P_q \approx \frac{1}{\sqrt{2\pi m}(1-\rho)(e^\rho/e\rho)^m + 1}.$$

By using the above closed-form expression of P_q, we get a closed-form approximation of the expected service charge to a service request as

$$C \approx a\bar{r}\Bigg(1 - \frac{1}{(\sqrt{2\pi m}(1-\rho)(e^\rho/e\rho)^m + 1)} \times \frac{1}{((ms-\lambda\bar{r})(c/s_0 - 1/s) + 1)} \times \frac{1}{((ms-\lambda\bar{r})(a/d + c/s_0 - 1/s) + 1)}\Bigg).$$

For convenience, we rewrite C as

$$C = a\bar{r}\left(1 - \frac{1}{D_1 D_2 D_3}\right),$$

where

$$\begin{aligned} D_1 &= \sqrt{2\pi m}(1-\rho)(e^\rho/e\rho)^m + 1, \\ D_2 &= (ms-\lambda\bar{r})(c/s_0 - 1/s) + 1, \\ D_3 &= (ms-\lambda\bar{r})(a/d + c/s_0 - 1/s) + 1. \end{aligned}$$

Our discussion in this section is based on the above closed-form expression of C.

7.1 Optimal Size

Given λ, $\bar{r}$, s, P^*, α, β, γ, a, c, and d, our first problem is to find m such that G is maximized. To maximize G, we need to find m such that

$$\frac{\partial G}{\partial m} = \lambda\frac{\partial C}{\partial m} - (\beta + \gamma P^*) = 0,$$

for the idle-speed model, and

$$\frac{\partial G}{\partial m} = \lambda\frac{\partial C}{\partial m} - (\beta + \gamma(\xi s^\alpha + P^*)) = 0,$$

for the constant-speed model, where

$$\frac{\partial C}{\partial m} = \frac{a\bar{r}}{(D_1D_2D_3)^2} \times \left(D_2D_3\frac{\partial D_1}{\partial m} + D_1D_3\frac{\partial D_2}{\partial m} + D_1D_2\frac{\partial D_3}{\partial m} \right).$$

To continue the calculation, we rewrite D_1 as

$$D_1 = \sqrt{2\pi m}(1-\rho)R + 1,$$

where

$$R = (e^\rho/e\rho)^m.$$

Notice that

$$\ln R = m\ln(e^\rho/e\rho) = m(\rho - \ln\rho - 1).$$

Since

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

we get

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

and

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

Now, we have

$$\begin{aligned}
\frac{\partial D_1}{\partial m} &= \sqrt{2\pi}\bigg(\frac{1}{2\sqrt{m}}(1-\rho)R + \sqrt{m}\left(-\frac{\partial\rho}{\partial m}\right)R \\
&\qquad + \sqrt{m}(1-\rho)\frac{\partial R}{\partial m}\bigg) \\
&= \sqrt{2\pi}\bigg(\frac{1}{2\sqrt{m}}(1-\rho)R + \sqrt{m}\frac{\rho}{m}R \\
&\qquad - \sqrt{m}(1-\rho)R\ln\rho\bigg) \\
&= \sqrt{2\pi}\bigg(\frac{1}{2\sqrt{m}}(1-\rho)R + \frac{1}{\sqrt{m}}\rho R \\
&\qquad - \sqrt{m}(1-\rho)(\ln\rho)R\bigg) \\
&= \sqrt{2\pi}\left(\frac{1}{2\sqrt{m}}(1+\rho)R - \sqrt{m}(1-\rho)(\ln\rho)R \right).
\end{aligned}$$

Furthermore, we have

$$\frac{\partial D_2}{\partial m} = cs/s_0 - 1,$$

and

$$\frac{\partial D_3}{\partial m} = as/d + cs/s_0 - 1.$$

Although there is no closed-form solution to m, we notice that $\partial G/\partial m$ is a decreasing function of m. Therefore, m can be found numerically by using the standard bisection method.

In Figures 5 and 6, we demonstrate the net business gain G in one unit of time as a function of m and λ for the two power consumption models respectively, using the same parameters in Figures 1–4. For $\lambda = 2.9, 3.9, 4.9, 5.9, 6.9$, we display G for m large enough such that $\rho < 1$. We notice that there is an optimal choice of m such that G is maximized. Using our analytical results, we can find m such that $\partial G/\partial m = 0$. The optimal value of m is 3.67479, 4.79218, 5.89396, 6.98457, 8.06655, respectively, for the idle-speed model, and 3.54842, 4.64834, 5.73478, 6.81160, 7.88104, respectively, for the constant-speed model.

Such server size optimization has clear physical interpretation. When m is small such that ρ is close to 1, the waiting times of service requests are excessively long, and the service charges and the net business gain are low. As m increases, the waiting times are significantly reduced, and the service charges and the net business gain are increased. However, as m further increases, there will be no more increase in the expected services charge which has an upper bound $a\bar{r}$; on the other hand, the cost of a service provider (i.e., the rental cost and base power consumption) increases, so that the net business gain is actually reduced. Hence, there is an optimal choice of m which maximizes the profit.

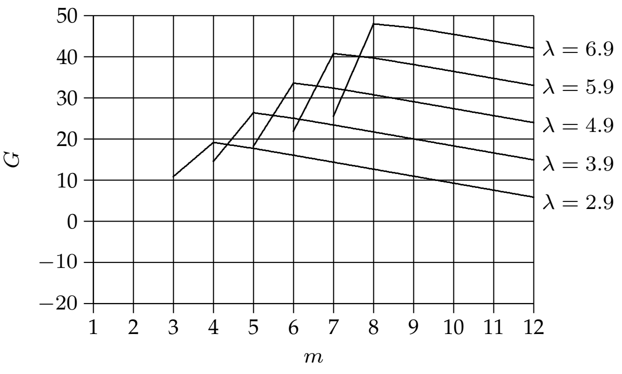

Fig. 5. Net business gain G vs. m and λ (idle-speed model).

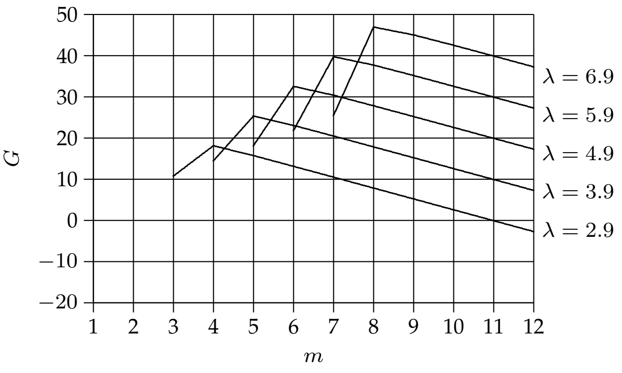

Fig. 6. Net business gain G vs. m and λ (constant-speed model).

7.2 Optimal Speed

Given λ, $\bar{r}$, m, P^*, α, β, γ, a, c, and d, our second problem is to find s such that G is maximized. To maximize G, we need to find s such that

$$\frac{\partial G}{\partial s} = \lambda \frac{\partial C}{\partial s} - \gamma\lambda\bar{r}\xi(\alpha-1)s^{\alpha-2} = 0,$$

for the idle-speed model, and

$$\frac{\partial G}{\partial s} = \lambda \frac{\partial C}{\partial s} - \gamma m\xi\alpha s^{\alpha-1} = 0,$$

for the constant-speed model, where

$$\frac{\partial C}{\partial s} = \frac{a\bar{r}}{(D_1D_2D_3)^2} \times \left(D_2D_3\frac{\partial D_1}{\partial s} + D_1D_3\frac{\partial D_2}{\partial s} + D_1D_2\frac{\partial D_3}{\partial s} \right).$$

Similar to the calculation in the last subsection, we have

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

and

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

and

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

Now, we have

$$\begin{aligned}\frac{\partial D_1}{\partial s} &= \sqrt{2\pi m}\left(\left(-\frac{\partial \rho}{\partial s}\right)R + (1-\rho)\frac{\partial R}{\partial s}\right)\\ &= \sqrt{2\pi m}\left(\frac{\rho}{s}R + \frac{m}{s}(1-\rho)^2R\right)\\ &= \sqrt{2\pi m}\left(\rho + m(1-\rho)^2\right)\frac{R}{s}.\end{aligned}$$

Furthermore, we have

$$\begin{aligned}\frac{\partial D_2}{\partial s} &= m\left(\frac{c}{s_0} - \frac{1}{s}\right) + (ms - \lambda\bar{r})\frac{1}{s^2}\\ &= \frac{mc}{s_0} - \frac{\lambda\bar{r}}{s^2},\end{aligned}$$

and

$$\begin{aligned}\frac{\partial D_3}{\partial s} &= m\left(\frac{a}{d} + \frac{c}{s_0} - \frac{1}{s}\right) + (ms - \lambda\bar{r})\frac{1}{s^2}\\ &= m\left(\frac{a}{d} + \frac{c}{s_0}\right) - \frac{\lambda\bar{r}}{s^2}.\end{aligned}$$

Although there is no closed-form solution to s, we notice that $\partial G/\partial s$ is a decreasing function of s. Therefore, s can be found numerically by using the standard bisection method.

In Figures 7 and 8, we demonstrate the net business gain G in one unit of time as a function of s and λ for the two power consumption models respectively, using the same parameters in Figures 1–6. For $\lambda = 2.9, 3.9, 4.9, 5.9, 6.9$, we display G for s large enough such that $\rho < 1$. We notice that there is an optimal choice of s such that G is maximized. Using our analytical results, we can find s such that $\partial G/\partial s = 0$. The optimal value of s is 0.63215, 0.76982, 0.90888, 1.04911, 1.19011, respectively, for the idle-speed model, and 0.57015, 0.71009, 0.85145, 0.99348, 1.13584, respectively, for the constant-speed model.

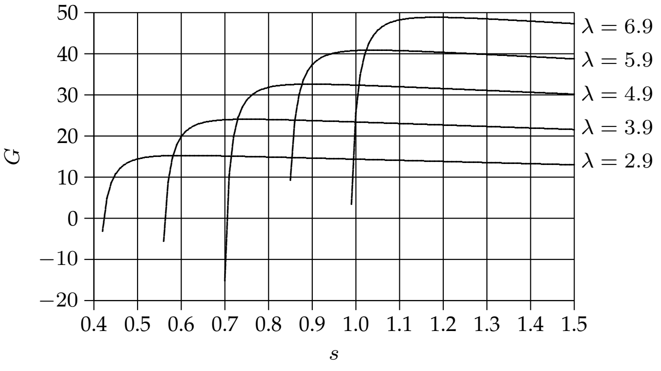

Fig. 7. Net business gain G vs. s and λ (idle-speed model).

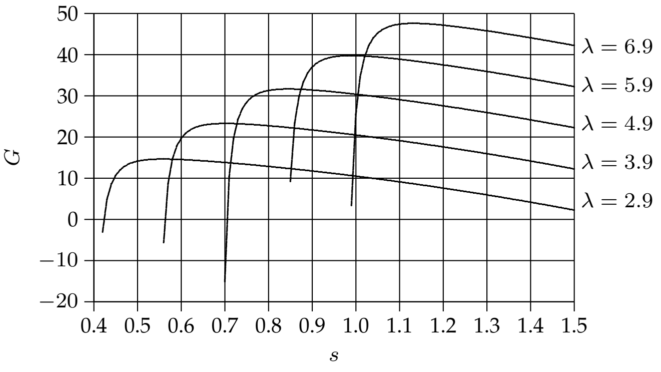

Fig. 8. Net business gain G vs. s and λ (constant-speed model).

Such server speed optimization also has clear physical interpretation. When s is small such that ρ is close to 1, the waiting times of service requests are excessively long, and the service charges and the net business gain are low. As s increases, the waiting times are significantly reduced, and the service charges and the net business gain are increased. However, as s further increases, there will be no more increase in the expected services charge which has an upper bound $a\bar{r}$; on the other hand, the cost of a service provider (i.e., the cost of energy consumption) increases, so that the net business gain is actually reduced. Hence, there is an optimal choice of s which maximizes the profit.

7.3 Optimal Size and Speed

Given λ, $\bar{r}$, P^*, α, β, γ, a, c, and d, our third problem is to find m and s such that G is maximized. To maximize G, we need to find m and s such that $\partial G/\partial m = 0$ and $\partial G/\partial s = 0$, where $\partial G/\partial m$ and $\partial G/\partial s$ have been derived in the last two subsections. The two equations can be solved by a nested bisection search procedure.

In Figures 9 and 10, we demonstrate the net business gain G in one unit of time as a function of

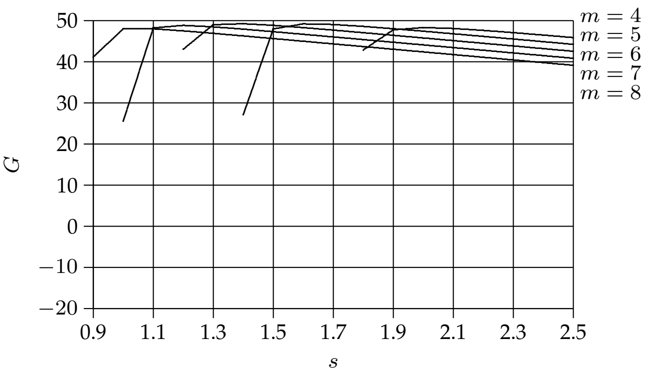

Fig. 9. Net business gain G vs. s and m (idle-speed model).

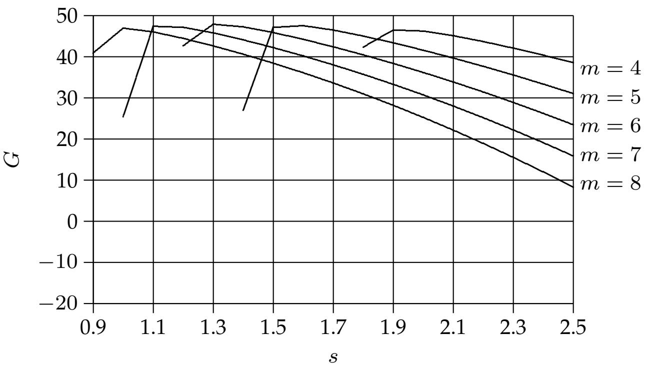

Fig. 10. Net business gain G vs. s and m (constant-speed model).

s and m for the two power consumption models respectively, using the same parameters in Figures 1–8, where $\lambda = 6.9$. For $m = 4, 5, 6, 7, 8$, we display G for s large enough such that $\rho < 1$. Using our analytical results, we can find m and s such that $\partial G/\partial m = 0$ and $\partial G/\partial s = 0$. For the idle-speed model, the theoretically optimal values are $m = 5.56827$ and $s = 1.46819$, which result in the maximum $G = 49.25361$ by using the closed-form approximation of C. Practically, m can be either 5 or 6. When $m = 5$, the optimal value of s is 1.62236, which results in the maximum $G = 49.16510$. When $m = 6$, the optimal value of s is 1.37044, which results in the maximum $G = 49.18888$. Hence, the practically optimal setting is $m = 6$ and $s = 1.37044$, and the maximum net business gain in one unit of time is $G = 49.29273$ by using the exact expression of C. For the constant-speed model, the theoretically optimal values are $m = 5.79074$ and $s = 1.35667$, which result in the maximum $G = 47.80769$ by using the closed-form approximation of C. Practically, m can be either 5 or 6. When $m = 5$, the optimal value of s is 1.55839, which results in the maximum $G = 47.63979$. When $m = 6$, the optimal value of s is 1.31213, which results in the maximum $G = 47.78640$. Hence, the practically optimal setting is $m = 6$ and $s = 1.31213$, and the maximum net business gain in one unit of time is $G = 47.91830$ by using the exact expression of C.

8 Simulation Results

Simulations have been conducted for two purposes, namely, (1) to validate our analytical results (Theorems 1 and 2); (2) and to find more effective queueing disciplines which increase the net profit of a service provider.

In Table 1, we show our simulation results by using the same parameters in Figures 1–10. For each $\lambda = 6.05, 6.15, ..., 6.95$, we trace the behavior of an M/M/m queueing system with the FCFS queueing discipline by generating a Poisson stream of service requests with arrival rate λ, recording the waiting and response times of each service request, and calculating the service charge to each service request. The average service charge of 1,000,000 service requests is reported in Table 1 for each λ. Notice that the maximum 99% confidence interval of all the data in the table is $\pm 0.5165372\%$. The analytical data in the table are obtained by Theorem 2 to calculate the expected charge to a service request. It is easily observed that our simulation results match with the analytical data very well. These results validate our theoretically predicted service charge in Theorem 2, which is based on our analytical result on waiting time distribution in Theorem 1.

Our analysis in this paper is based on the FCFS queueing discipline. A different queueing discipline may change the distribution of the waiting times, and thus, changes the average task response time and the expected service charge. Since the cost of a service provider remains the same, an increased/decreased expected service charge to a service request increases/decreases the expected net business gain of a service provider. To show the effect of queueing disciplines on the net profit of a service provider, we only need to show the effect of queueing disciplines on the expected service charge to a service request. We consider two simple queueing disciplines, namely,

- Shortest Task First (STF): Tasks (service requests) are arranged in a waiting queue in the increasing order of their task execution requirements;
- Largest Task First (LTF): Tasks (service requests) are arranged in a waiting queue in the decreasing order of their task execution requirements.

While other queueing disciplines can also be considered, these two disciplines are already very encouraging.

In Table 1, we also display our simulation results for STF and LTF by using the same parameters for FCFS. For each $\lambda = 6.05, 6.15, ..., 6.95$, we trace the behavior of an M/M/m queueing system with the STF and the LTF queueing disciplines respectively. The average service charge of 1,000,000 service requests is reported in Table 1 for each λ. We have the following observations.

Table 1: Simulation Results on the Expected Service Charge.

λ	Analytical	FCFS	STF	LTF
6.05	9.8245499	9.8185300	9.9792709	9.5841297
6.15	9.7798220	9.7762701	9.9651989	9.5239829
6.25	9.7193304	9.7151778	9.9565597	9.4591742
6.35	9.6351404	9.6384268	9.9514632	9.3984495
6.45	9.5136508	9.5015511	9.9015669	9.3375343
6.55	9.3298719	9.3432460	9.8674891	9.2532065
6.65	9.0334251	9.0317035	9.8038810	9.1751988
6.75	8.5084481	8.4933564	9.7186839	9.0780185
6.85	7.4277447	7.4383065	9.5750764	8.9842690
6.95	4.4400583	4.4744746	9.3974538	8.8743559

- STF performs consistently better than FCFS. Furthermore, while the expected service charge drops significantly for FCFS when λ is close to the saturation point and the average waiting time becomes very long, the expected service charge of STF is still close to $a\bar{r}$ when λ is large.
- LTF performs worse than FCFS when λ is not very large. However, when λ is close to the saturation point, LTF performs better than FCFS in the sense that the expected service charge of LTF does not drop significantly when λ is large.

Unfortunately, due to lack of an analytical result on waiting time distribution similar to Theorem 1 for STF and LTF, the analytical work conducted in this paper for FCFS cannot be duplicated for STF and LTF. This can be an interesting subject for further investigation.

9 Concluding Remarks

We have proposed a pricing model for cloud computing which takes many factors into considerations, such as the requirement r of a service, the workload λ of an application environment, the configuration (m and s) of a multiserver system, the service level agreement c, the satisfaction (r and s_0) of a consumer, the quality (W and T) of a service, the penalty d of a low quality service, the cost (β and m) of renting, the cost (α, γ, P^*, and P) of energy consumption, and a service provider's margin and profit a. By using an M/M/m queueing model, we formulated and solved the problem of optimal multiserver configuration for profit maximization in a cloud computing environment. Our discussion can be easily extended to other service charge functions. Our methodology can be applied to other pricing models.

Our investigation in this paper is only an initial attempt in this area. We would like to mention several further research directions.

- First, in a cloud computing environment, a multiserver system can be dynamically configured as a virtual cluster from a physical cluster, or a virtual multicore server from a physical multicore processor, or a virtual multiserver system from any elastic and dynamic resources. Our profit maximization problem can be extended to such virtual multiserver systems. To this end, a queueing model that accurately describes such a virtual multiserver system is required and needs to be developed. Such a model should be able to characterize a virtual multiserver system from a partially available physical system with deterministic or randomized availability.
- Second, our profit maximization problem can be extended to multiple heterogeneous multiserver systems of different sizes and speeds and application environments with total power consumption constraint. This is a multi-variable optimization problem, which is much more complicated than the optimization performed for a single multiserver system in this paper. Such optimization has significant and practical applications in designing energy-efficient data centers.
- Third, when a multicore server processor is spatially divided into several multicore servers, our profit maximization problem can be defined for multiple multiserver systems. When the cores have a fixed speed, the optimization problem has a total server size constraint. When the cores have variable speeds, the optimization problem has a total server size constraint as well as a power consumption constraint.
- Fourth, when a physical machine is temporally partitioned into several virtual machines, i.e., when we are facing a dynamic cloud configuration with multi-tenant utilization, our profit maximization problem might be defined for multiple multiserver systems with total server speed constraint. Again, this part of the research relies on an accurate queueing model for virtual machines which is currently not available.

We believe that the effort made in this paper should inspire significant subsequent studies in profit maximization for cloud computing.

Appendix. Proofs of the Theorems

Proof of Theorem 1. If there are $k < m$ tasks in the queueing system when a new service request arrives, the waiting time of the service request is $W_k = 0$. The pdf of W_k can be represented as

$$f_{W_k}(t) = u(t),$$

for all $0 \le k \le m-1$. Furthermore, we have

$$\overline{W}_k = \lim_{z\to\infty} \frac{1}{2z} = 0,$$

for all $0 \le k \le m-1$.

If there are $k \ge m$ tasks in the queueing system when a new service request arrives, then the service request must wait until a server is available. Notice that due to the memoryless property of an exponential distribution, the remaining execution time of a task is always the same random variable as before, i.e., the original task execution time x with pdf $f_x(t) = \mu e^{-\mu t}$, no matter how long the task has been executed. Let $x_1, x_2, ..., x_m$ be the remaining execution times of the m tasks in execution when a new service request arrives. Then, we have $f_{x_j}(t) = \mu e^{-\mu t}$, for all $1 \le j \le m$.

It is clear that $y = \min\{x_1, x_2, ..., x_m\}$ is the time until the next completion of a task. Since

$$\boldsymbol{P}[y \ge t] = \prod_{j=1}^{m} \boldsymbol{P}[x_j \ge t] = \prod_{j=1}^{m} e^{-\mu t} = e^{-m\mu t},$$

we get

$$F_y(t) = \boldsymbol{P}[y \le t] = 1 - \boldsymbol{P}[y \ge t] = 1 - e^{-m\mu t},$$

and $f_y(t) = m\mu e^{-m\mu t}$, that is, y is also an exponential random variable with mean $1/m\mu$. The time until the next completion of a task is always the same random variable y, i.e., the minimum value of m i.i.d. exponential random variables with pdf $f_y(t) = m\mu e^{-m\mu t}$.

Notice that due to multiple servers, a task does not need to wait until all tasks in front of it are completed. Actually, the waiting time W_k of a task (under the condition that there are $k \ge m$ tasks in the queueing system when the task arrives) is $W_k = y_1 + y_2 + \cdots + y_{k-m+1}$, where $y_1, y_2, ..., y_{k-m+1}$ are i.i.d. exponential random variables with the same pdf $f_y(t) = m\mu e^{-m\mu t}$. The reason is that after $k - m + 1$ completions of task executions, a task is at the front of the waiting queue and there is an available server, and the task will be scheduled to be executed. It is well known that $y_1 + y_2 + \cdots + y_k$ has an Erlang distribution whose pdf is

$$\frac{m\mu(m\mu t)^{k-1}}{(k-1)!} e^{-m\mu t}.$$

Hence, we get the pdf of W_k

$$f_{W_k}(t) = \frac{m\mu(m\mu t)^{k-m}}{(k-m)!} e^{-m\mu t},$$

for all $k \ge m$. Notice that $\bar{y} = 1/m\mu$ and

$$\overline{W}_k = (k-m+1)\bar{y} = \frac{k-m+1}{m\mu} = (k-m+1)\frac{\bar{x}}{m},$$

for all $k \ge m$.

Summarizing the above discussion, we obtain the pdf of the waiting time W of a service request as follows:

$$\begin{aligned}
& f_W(t) \\
&= \sum_{k=0}^{\infty} p_k f_{W_k}(t) \\
&= \left(\sum_{k=0}^{m-1} p_k \right) u(t) + \sum_{k=m}^{\infty} p_k \frac{m\mu(m\mu t)^{k-m}}{(k-m)!} e^{-m\mu t} \\
&= (1 - P_q)u(t) + \sum_{k=m}^{\infty} p_0 \frac{m^m \rho^k}{m!} \cdot \frac{m\mu(m\mu t)^{k-m}}{(k-m)!} e^{-m\mu t} \\
&= (1 - P_q)u(t) \\
&\quad + p_0 \frac{m^m \rho^m}{m!} m\mu e^{-m\mu t} \sum_{k=m}^{\infty} \frac{\rho^{k-m}(m\mu t)^{k-m}}{(k-m)!} \\
&= (1 - P_q)u(t) + p_0 \frac{(m\rho)^m}{m!} m\mu e^{-m\mu t} \sum_{k=0}^{\infty} \frac{(\rho m\mu t)^k}{k!} \\
&= (1 - P_q)u(t) + p_m m\mu e^{-m\mu t} e^{\rho m\mu t} \\
&= (1 - P_q)u(t) + m\mu p_m e^{-(1-\rho)m\mu t}.
\end{aligned}$$

This proves the theorem. ∎

Proof of Theorem 2. Since W is a random variable, $C(r, W)$, which is viewed as a function of W for a fixed r, is also a random variable. The expected charge to a service request with execution requirement r is (in the following, dt in parenthesis is the product of the penalty factor and the time variable)

$$\begin{aligned}
& C(r) \\
&= \overline{C(r, W)} \\
&= \int_0^{\infty} f_W(t) C(r, t) dt \\
&= \int_0^{(a/d + c/s_0 - 1/s)r} f_W(t) C(r, t) dt \\
&= \int_0^{(a/d + c/s_0 - 1/s)r} ((1 - P_q)u(t) \\
&\quad + m\mu p_m e^{-(1-\rho)m\mu t}) C(r, t) dt \\
&= \int_0^{(a/d + c/s_0 - 1/s)r} (1 - P_q)u(t) C(r, t) dt \\
&\quad + \int_0^{(a/d + c/s_0 - 1/s)r} m\mu p_m e^{-(1-\rho)m\mu t} C(r, t) dt \\
&= (1 - P_q)ar + \int_0^{(c/s_0 - 1/s)r} m\mu p_m e^{-(1-\rho)m\mu t} C(r, t) dt \\
&\quad + \int_{(c/s_0 - 1/s)r}^{(a/d + c/s_0 - 1/s)r} m\mu p_m e^{-(1-\rho)m\mu t} C(r, t) dt \\
&= (1 - P_q)ar + \int_0^{(c/s_0 - 1/s)r} m\mu p_m e^{-(1-\rho)m\mu t} ar dt \\
&\quad + \int_{(c/s_0 - 1/s)r}^{(a/d + c/s_0 - 1/s)r} m\mu p_m e^{-(1-\rho)m\mu t} \\
&\qquad \left(\left(a + \frac{cd}{s_0} - \frac{d}{s} \right) r - dt \right) dt \\
&= (1 - P_q)ar + m\mu p_m ar \int_0^{(c/s_0 - 1/s)r} e^{-(1-\rho)m\mu t} dt \\
&\quad + m\mu p_m \int_{(c/s_0 - 1/s)r}^{(a/d + c/s_0 - 1/s)r} e^{-(1-\rho)m\mu t} \\
&\qquad \left(\left(a + \frac{cd}{s_0} - \frac{d}{s} \right) r - dt \right) dt \\
&= (1 - P_q)ar + m\mu p_m ar \int_0^{(c/s_0 - 1/s)r} e^{-(1-\rho)m\mu t} dt \\
&\quad + m\mu p_m \left(a + \frac{cd}{s_0} - \frac{d}{s} \right) r \int_{(c/s_0 - 1/s)r}^{(a/d + c/s_0 - 1/s)r} e^{-(1-\rho)m\mu t} dt \\
&\quad - dm\mu p_m \int_{(c/s_0 - 1/s)r}^{(a/d + c/s_0 - 1/s)r} t e^{-(1-\rho)m\mu t} dt.
\end{aligned}$$

To continue the calculation, we notice that

$$\int e^{bt}dt = \frac{e^{bt}}{b},$$

and

$$\int te^{bt} = \frac{1}{b}\left(t - \frac{1}{b}\right)e^{bt}.$$

Hence, we have

$$\begin{aligned}\int_0^{(c/s_0-1/s)r} e^{-(1-\rho)m\mu t}dt &= -\frac{e^{-(1-\rho)m\mu t}}{(1-\rho)m\mu}\Bigg|_0^{(c/s_0-1/s)r} \\ &= \frac{1-e^{-(1-\rho)m\mu(c/s_0-1/s)r}}{(1-\rho)m\mu},\end{aligned}$$

and

$$\begin{aligned}&\int_{(c/s_0-1/s)r}^{(a/d+c/s_0-1/s)r} e^{-(1-\rho)m\mu t}dt \\ &= -\frac{e^{-(1-\rho)m\mu t}}{(1-\rho)m\mu}\Bigg|_{(c/s_0-1/s)r}^{(a/d+c/s_0-1/s)r} \\ &= \frac{e^{-(1-\rho)m\mu(c/s_0-1/s)r} - e^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r}}{(1-\rho)m\mu},\end{aligned}$$

and

$$\begin{aligned}&\int_{(c/s_0-1/s)r}^{(a/d+c/s_0-1/s)r} te^{-(1-\rho)m\mu t}dt \\ &= -\frac{1}{(1-\rho)m\mu}\left(t + \frac{1}{(1-\rho)m\mu}\right) \\ &\qquad e^{-(1-\rho)m\mu t}\Bigg|_{(c/s_0-1/s)r}^{(a/d+c/s_0-1/s)r} \\ &= \frac{1}{(1-\rho)m\mu}\Bigg(\left(\left(\frac{c}{s_0}-\frac{1}{s}\right)r + \frac{1}{(1-\rho)m\mu}\right) \\ &\qquad e^{-(1-\rho)m\mu(c/s_0-1/s)r} \\ &\quad - \left(\left(\frac{a}{d}+\frac{c}{s_0}-\frac{1}{s}\right)r + \frac{1}{(1-\rho)m\mu}\right) \\ &\qquad e^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r}\Bigg).\end{aligned}$$

Based on the above results, we get

$$\begin{aligned}&C(r) \\ &= (1-P_q)ar + m\mu p_m ar\frac{1-e^{-(1-\rho)m\mu(c/s_0-1/s)r}}{(1-\rho)m\mu} \\ &\quad + m\mu p_m\left(a + \frac{cd}{s_0} - \frac{d}{s}\right)r \\ &\quad \frac{e^{-(1-\rho)m\mu(c/s_0-1/s)r} - e^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r}}{(1-\rho)m\mu} \\ &\quad - dm\mu p_m\frac{1}{(1-\rho)m\mu} \\ &\quad \Bigg(\left(\left(\frac{c}{s_0}-\frac{1}{s}\right)r + \frac{1}{(1-\rho)m\mu}\right)e^{-(1-\rho)m\mu(c/s_0-1/s)r} \\ &\quad - \left(\left(\frac{a}{d}+\frac{c}{s_0}-\frac{1}{s}\right)r + \frac{1}{(1-\rho)m\mu}\right) \\ &\qquad e^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r}\Bigg) \\ &= (1-P_q)ar + \frac{ap_m}{1-\rho}\left(r - re^{-(1-\rho)m\mu(c/s_0-1/s)r}\right) \\ &\quad + \frac{p_m}{1-\rho}\left(a + \frac{cd}{s_0} - \frac{d}{s}\right) \\ &\quad \left(re^{-(1-\rho)m\mu(c/s_0-1/s)r} - re^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r}\right) \\ &\quad - \frac{dp_m}{1-\rho}\Bigg(\left(\frac{c}{s_0}-\frac{1}{s}\right)re^{-(1-\rho)m\mu(c/s_0-1/s)r} \\ &\qquad + \frac{1}{(1-\rho)m\mu}e^{-(1-\rho)m\mu(c/s_0-1/s)r} \\ &\qquad - \left(\frac{a}{d}+\frac{c}{s_0}-\frac{1}{s}\right)re^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r} \\ &\qquad - \frac{1}{(1-\rho)m\mu}e^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r}\Bigg) \\ &= (1-P_q)ar + aP_q\left(r - re^{-(1-\rho)m\mu(c/s_0-1/s)r}\right) \\ &\quad + P_q\left(a + \frac{cd}{s_0} - \frac{d}{s}\right) \\ &\quad \left(re^{-(1-\rho)m\mu(c/s_0-1/s)r} - re^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r}\right) \\ &\quad - dP_q\Bigg(\left(\frac{c}{s_0}-\frac{1}{s}\right)re^{-(1-\rho)m\mu(c/s_0-1/s)r} \\ &\qquad + \frac{1}{(1-\rho)m\mu}e^{-(1-\rho)m\mu(c/s_0-1/s)r} \\ &\qquad - \left(\frac{a}{d}+\frac{c}{s_0}-\frac{1}{s}\right)re^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r} \\ &\qquad - \frac{1}{(1-\rho)m\mu}e^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r}\Bigg) \\ &= ar - \frac{dP_q}{(1-\rho)m\mu} \\ &\quad \left(e^{-(1-\rho)m\mu(c/s_0-1/s)r} - e^{-(1-\rho)m\mu(a/d+c/s_0-1/s)r}\right).\end{aligned}$$

Since r is a random variable, $C(r)$, which is viewed as a function of r, is also a random variable. Let the pdf of task execution requirement r to be

$$f_r(z) = \frac{1}{\bar{r}}e^{-z/\bar{r}}.$$

The expected charge to a service request is

$$\begin{aligned}&C \\ &= \overline{C(r)} \\ &= \int_0^\infty f_r(z)C(z)dz \\ &= \int_0^\infty \frac{1}{\bar{r}}e^{-z/\bar{r}}C(z)dz \\ &= \frac{1}{\bar{r}}\int_0^\infty e^{-z/\bar{r}}\Bigg(az - \frac{dP_q}{(1-\rho)m\mu} \\ &\qquad \big(e^{-(1-\rho)m\mu(c/s_0-1/s)z} \\ &\qquad\qquad - e^{-(1-\rho)m\mu(a/d+c/s_0-1/s)z}\big)\Bigg)dz\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\bar{r}}\Bigg(a\int_0^\infty z e^{-z/\bar{r}} dz \\
&\quad - \frac{dP_q}{(1-\rho)m\mu}\Bigg(\int_0^\infty e^{-((1-\rho)m\mu(c/s_0-1/s)+1/\bar{r})z} dz \\
&\quad - \int_0^\infty e^{-((1-\rho)m\mu(a/d+c/s_0-1/s)+1/\bar{r})z} dz\Bigg)\Bigg).
\end{aligned}
$$

Since

$$
\int_0^\infty z e^{-bz} dz = -\frac{1}{b}\left(z+\frac{1}{b}\right)e^{-bz}\Bigg|_0^\infty = \frac{1}{b^2},
$$

and

$$
\int_0^\infty e^{-bz} dz = -\frac{e^{-bz}}{b}\Bigg|_0^\infty = \frac{1}{b},
$$

we get

$$
\begin{aligned}
&C \\
&= \frac{1}{\bar{r}}\Bigg(a\bar{r}^2 - \frac{dP_q}{(1-\rho)m\mu}\Bigg(\frac{1}{(1-\rho)m\mu(c/s_0-1/s)+1/\bar{r}} \\
&\quad - \frac{1}{(1-\rho)m\mu(a/d+c/s_0-1/s)+1/\bar{r}}\Bigg)\Bigg) \\
&= a\bar{r} - \frac{dP_q}{(1-\rho)m\mu}\Bigg(\frac{1}{\bar{r}(1-\rho)m\mu(c/s_0-1/s)+1} \\
&\quad - \frac{1}{\bar{r}(1-\rho)m\mu(a/d+c/s_0-1/s)+1}\Bigg) \\
&= a\bar{r} - \frac{dP_q}{(1-\rho)m\mu}\cdot\frac{\bar{r}(1-\rho)m\mu(a/d)}{(\bar{r}(1-\rho)m\mu(c/s_0-1/s)+1)} \\
&\quad \times \frac{1}{(\bar{r}(1-\rho)m\mu(a/d+c/s_0-1/s)+1)} \\
&= a\bar{r} - \frac{a\bar{r}P_q}{(\bar{r}(1-\rho)m\mu(c/s_0-1/s)+1)} \\
&\quad \times \frac{1}{(\bar{r}(1-\rho)m\mu(a/d+c/s_0-1/s)+1)} \\
&= a\bar{r}\Bigg(1 - \frac{P_q}{((ms-\lambda\bar{r})(c/s_0-1/s)+1)} \\
&\quad \times \frac{1}{((ms-\lambda\bar{r})(a/d+c/s_0-1/s)+1)}\Bigg).
\end{aligned}
$$

The theorem is proven. ∎

ACKNOWLEDGMENTS

This work is partially supported by Ministry of Science and Technology of China under National 973 Basic Research Grants No. 2011CB302805 and No. 2011CB302505, and National 863 High-tech Program Grant No. 2011AA040501.

REFERENCES

[1] http://en.wikipedia.org/wiki/CMOS

[2] http://en.wikipedia.org/wiki/Service_level_agreement

[3] M. Armbrust, *et al.*, "Above the clouds: a Berkeley view of cloud computing," Technical Report No. UCB/EECS-2009-28, February 2009.

[4] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, pp. 1507-1542, 2007.

[5] 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 Generation Computer Systems*, vol. 25, no. 6, pp. 599-616, 2009.

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

[7] B. N. Chun and D. E. Culler, "User-centric performance analysis of market-based cluster batch schedulers," *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[8] D. Durkee, "Why cloud computing will never be free," *Communications of the ACM*, vol. 53, no. 5, pp. 62-69, 2010.

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

[10] Intel, *Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor – White Paper*, March 2004.

[11] D. E. Irwin, L. E. Grit, and J. S. Chase, "Balancing risk and reward in a market-based task service," *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, pp. 160-169, 2004.

[12] L. Kleinrock, *Queueing Systems, Volume 1: Theory*, John Wiley and Sons, New York, 1975.

[13] Y. C. Lee, C. Wang, A. Y. Zomaya, and B. B. Zhou, "Profit-driven service request scheduling in clouds," *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp. 15-24, 2010.

[14] K. Li, "Optimal load distribution for multiple heterogeneous blade servers in a cloud computing environment," *Proceedings of the 25th IEEE International Parallel and Distributed Processing Symposium Workshops* (8th High-Performance Grid and Cloud Computing Workshop), pp. 943-952, Anchorage, Alaska, May 16-20, 2011.

[15] K. Li, "Optimal configuration of a multicore server processor for managing the power and performance tradeoff," *Journal of Supercomputing*, DOI: 10.1007/s11227-011-0686-1, published online 28 September 2011.

[16] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, 2009. http://csrc.nist.gov/groups/SNS/cloud-computing/

[17] F. I. Popovici and J. Wilkes, "Profitable services in an uncertain world," *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, 2005.

[18] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, and R. Buyya, "Libra: a computational economy-based job scheduling system for clusters," *Software – Practice and Experience*, vol. 34, pp. 573-590, 2004.

[19] C. S. Yeo and R. Buyya, "A taxonomy of market-based resource management systems for utility-driven cluster computing," *Software – Practice and Experience*, vol. 36, pp. 1381-1419, 2006.

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

Junwei Cao received his Ph.D. in computer science from the University of Warwick, Coventry, UK, in 2001. He received his bachelor and master degrees in control theories and engineering in 1996 and 1998, respectively, both from Tsinghua University, Beijing, China. He is currently a Professor and Vice Director, Research Institute of Information Technology, Tsinghua University, Beijing, China. He is also Director of Common Platform and Technology Division, Tsinghua National Laboratory for Information Science and Technology. Before joining Tsinghua University in 2006, he was a research scientist at MIT LIGO Laboratory and NEC Laboratories Europe for about 5 years. He has published over 130 papers and cited by international scholars for over 2,200 times. He is the book editor of *Cyberinfrastructure Technologies and Applications*, published by

Nova Science in 2009. His research is focused on advanced computing technologies and applications. Dr. Cao is a senior member of the IEEE Computer Society and a member of the ACM and CCF.

Kai Hwang is a Professor of EE/CS at the University of Southern California. He also chairs the IV-endowed visiting chair professor group at Tsinghua University in China. He received the Ph.D. from University of California, Berkeley in 1972. He has published 8 books and over 218 scientific papers in computer architecture, parallel processing, distributed systems, cloud computing, network security, and Internet applications. His popular books have been adopted worldwide and translated into 4 foreign languages. His published papers have been cited more than 9,000 times. Dr. Hwang's latest book *Distributed and Cloud Computing: from Parallel Processing to the Internet of Things* (with G. Fox and J. Dongarra) was just published by Kaufmann in 2011. Dr. Hwang was awarded an IEEE Fellow grade in 1986, received the 2004 CFC Outstanding Achievement Award, and the Founders Award for his pioneering work in parallel processing from IEEE IPDPS in 2011. He has served as a founding Editor-in-Chief of the *Journal of Parallel and Distributed Computing* for 28 years. He has delivered 34 keynote addresses on advanced computing systems and cutting-edge information technologies in major IEEE/ACM Conferences. Dr. Hwang has performed advisory, consulting and collaborative work for IBM, Intel, MIT Lincoln Lab, JPL at Caltech, ETL in Japan, ITRI in Taiwan, GMD in Germany, INRIA in France, and Chinese Academy of Sciences.

Keqin Li is a SUNY Distinguished Professor in computer science and an Intellectual Ventures endowed visiting chair professor at Tsinghua University, China. His research interests are mainly in design and analysis of algorithms, parallel and distributed computing, and computer networking. He has contributed extensively to processor allocation and resource management; design and analysis of sequential/parallel, deterministic/probabilistic, and approximation algorithms; parallel and distributed computing systems performance analysis, prediction, and evaluation; job scheduling, task dispatching, and load balancing in heterogeneous distributed systems; dynamic tree embedding and randomized load distribution in static networks; parallel computing using optical interconnections; dynamic location management in wireless communication networks; routing and wavelength assignment in optical networks; energy-efficient power management and performance optimization. Dr. Li has published over 240 research publications and has received several Best Paper Awards for his highest quality work. He is currently on the editorial board of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *Journal of Parallel and Distributed Computing*, *International Journal of Parallel, Emergent and Distributed Systems*, *International Journal of High Performance Computing and Networking*, and *Optimization Letters*.

Albert Y. Zomaya is currently the Chair Professor of High Performance Computing and Networking and Australian Research Council Professorial Fellow in the School of Information Technologies, The University of Sydney. He is also the Director of the Centre for Distributed and High Performance Computing which was established in late 2009. Professor Zomaya is the author/co-author of seven books, more than 400 papers, and the editor of nine books and 11 conference proceedings. He is the Editor-in-Chief of the *IEEE Transactions on Computers* and serves as an associate editor for 19 leading journals, such as, the *IEEE Transactions on Parallel and Distributed Systems* and *Journal of Parallel and Distributed Computing*. Professor Zomaya is the recipient of the Meritorious Service Award (in 2000) and the Golden Core Recognition (in 2006), both from the IEEE Computer Society. Also, he received the IEEE Technical Committee on Parallel Processing Outstanding Service Award and the IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing, both in 2011. Dr. Zomaya is a Chartered Engineer, a Fellow of AAAS, IEEE, and IET (UK).