Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 980945, 15 pages
http://dx.doi.org/10.1155/2015/980945

Research Article

Performance Analysis of Heterogeneous Data Centers in Cloud Computing Using a Complex Queuing Model

Wei-Hua Bai,[1,2] Jian-Qing Xi,[1] Jia-Xian Zhu,[2] and Shao-Wei Huang[2]

[1]*School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China*
[2]*School of Computer Science, Zhaoqing University, Zhaoqing 526061, China*

Correspondence should be addressed to Jian-Qing Xi; jianqingxi@163.com

Received 3 March 2015; Revised 28 May 2015; Accepted 1 June 2015

Academic Editor: Javier Plaza

Performance evaluation of modern cloud data centers has attracted considerable research attention among both cloud providers and cloud customers. In this paper, we investigate the heterogeneity of modern data centers and the service process used in these heterogeneous data centers. Using queuing theory, we construct a complex queuing model composed of two concatenated queuing systems and present this as an analytical model for evaluating the performance of heterogeneous data centers. Based on this complex queuing model, we analyze the mean response time, the mean waiting time, and other important performance indicators. We also conduct simulation experiments to confirm the validity of the complex queuing model. We further conduct numerical experiments to demonstrate that the traffic intensity (or utilization) of each execution server, as well as the configuration of server clusters, in a heterogeneous data center will impact the performance of the system. Our results indicate that our analytical model is effective in accurately estimating the performance of the heterogeneous data center.

1. Introduction

Cloud computing is a popular paradigm for providing services to users via three fundamental models: Infrastructure as a service (IaaS), Platform as a service (PaaS), and Software as a service (SaaS) [1, 2]. Cloud computing providers offer computing resources (e.g., servers, storage, networks, development platforms, and applications) to users either elastically or dynamically, according to user-demand and form of payment (e.g., pay-as-you-go) [3]. A modern data center is considered a heterogeneous environment because it contains many generations of servers with hardware configurations that may differ, especially in terms of the speed and capacity of the processors. Generally, these servers have been added to the data center gradually and provisioned to replace the existing (or "legacy") machines already in the data center's infrastructure [4, 5]. The heterogeneity of this mix of machine platforms affects the performance of data centers and the execution of cloud computing applications.

The ability to ensure the desired Quality of Service (QoS), which is a standard element of the Service Level Agreement (SLA) established between consumers and cloud providers, is one the most important business considerations for a cloud computing provider [6, 7]. A typical QoS outlines a set of critical performance indicators: mean response time, mean task queuing length, mean waiting time, mean task number of throughput, task capacity of the system, blocking probability, and probability of immediate service of the system. All of these indicators can be analyzed and described using queuing theory [8].

A task completed by the cloud computing center follows several steps: (1) a customer's request is transformed into a cloud computing task and sent to the task queue (the first level queue) maintained by the main scheduler server; (2) the scheduler allocates computing resources for the task and dispatches it to a node (execution server) to execute; (3) the node takes the task from the second level queue and allocates space in a CPU core (the main computing resource) to complete it; (4) the system outputs the result, and the task leaves the system. In this process, blocking will occur if a task leaves the system without having been executed (or the task may have been abandoned by the system) because of capacity constraint at the first level queue. A queuing model for a cloud computing heterogeneous data center can

analyze the relationship among three factors: the arrival rate of tasks, the core utilization, and the probability of a node (execution server) being dispatched. Cloud providers can utilize this information to analyze their systems' management and optimize their allocation of computing resources [9]. As we show in Section 2, much existing research has discussed the problem of how to analyze data center performance. However, these studies have not considered the following important issues:

(i) A cloud center can include a large number of servers of different generations with different CPU speeds and processor capacities. Existing research rarely considers cloud center heterogeneity; rather, studies generally assume that every node operates at the same speed.

(ii) A cloud system has at least two levels of scheduling processes (service processes): the main scheduler, which allocates computing resources, and the second scheduler (a node/execution server), which executes tasks. The arrival rate of a task to the main scheduler versus to a node is different, and this must be taken into account in the queuing analysis.

(iii) A node server with more than one speed and processor capacity will have different probabilities of being dispatched, depending on the situation. Each node has its own task queue to maintain, which is tracked by a separate queuing model that is included in the overall system queuing model.

(iv) A state-of-the-art resource allocation scheme based on the utilization threshold $[\delta_l, \delta_h]$ controls nodes that are to be added to a cloud system or removed from it [10].

To fill these gaps, in this paper, we employ a complex queuing model to investigate the performance of a cloud center and the relationship among various performance indicators in a heterogeneous data center. We aim to make the following contributions with this paper:

(1) We model the heterogeneous data center as a complex queuing system to characterize the service process. Two concatenated queuing systems compose the complex queuing model. The first is the finite capacity scheduler queuing system, which is used to study the main scheduling server. The second is the execution queuing system, which has multiple servers for each multicore server processor.

(2) Based on this complex queuing model, and using queuing theory, we analyze the real output rate of the main scheduling server—that is to say, the real arrival rate of the execution queuing systems, the mean response time of the system, the blocking probability, and other performance indicators—and use this analysis to evaluate the performance of, and the relationships among, these indicators in a cloud center.

(3) We conduct extensive discrete event simulations to evaluate and validate the results of using the complex queuing model to analyze the performance of heterogeneous data centers in cloud computing.

The remainder of this paper is organized as follows. Section 2 reviews and discusses the related scholarship on queuing models for cloud centers. Section 3 discusses our analytical model in detail. All of the performance metrics obtained by using this complex queuing model are presented in Section 4. In Section 5, we show the simulation results and compare them with our analytical results to evaluate and validate the applicability of the complex queuing model. Finally, conclusions and future work are presented in Section 6.

2. Related Work

Although much scholarship has been conducted on the performance analysis of a cloud center based on queuing models [11–16], factors including the loosely coupled architectures of cloud computing and the heterogeneity and dynamic infrastructure of a cloud center have not been considered.

In [11], the authors used an $M/G/m/m+r$ queuing system to evaluate a cloud computing center and obtained a complete probability distribution of response time, number of tasks in the system, and other important performance metrics. They also considered the different mean waiting times between heterogeneous services and homogeneous services under the same conditions in the system. Based on queuing theory and open Jackson's networks, a combination of $M/M/1$ and $M/M/m$ was presented to model the cloud platform for QoS requirements in [12]. In [13], the authors presented an $M/M/m$ queuing system and proposed a synthesis optimization of model, function, and strategy to optimize the performance of services in the cloud center. In [14, 15], the authors introduced a queuing model consisting of three concatenated queuing systems (the schedule queue, $M/M/1$, the computation queue, $M/M/m$, and the transmission queue, $M/M/1$) to characterize the service process in a multimedia cloud and to investigate the resource optimization problems of multimedia cloud computing. In [16], a cloud center was modeled as a $GI^X/M/S/N$ queuing model to discuss cloud service performance related to fault recovery.

In many existing methodologies [17–20], queuing theory has been used to analyze or optimize factors such as power allocation, load distribution, and profit control.

In [17], the authors presented an energy proportional model—which treats a server as an $M/G/1$ queuing system—as a strategy to improve performance efficiency. In [18], an $M/M/m$ queuing model, which considers factors such as the requirement γ of a service and the configuration of a multiserver system, is used to optimize multiserver configuration for profit maximization in cloud computing. In [19], the authors presented an $M/M/r/k$ queuing system to model a cloud server farm provided with finite capacity for profit optimization. Another study presents a queuing model for a group of heterogeneous multicore servers with different sizes and speeds to optimize power allocation and load distribution in a cloud computing environment [20]. In [17], the authors used the $M/M/c/c$ queuing system to model a cloud center with different priority classes to analyze

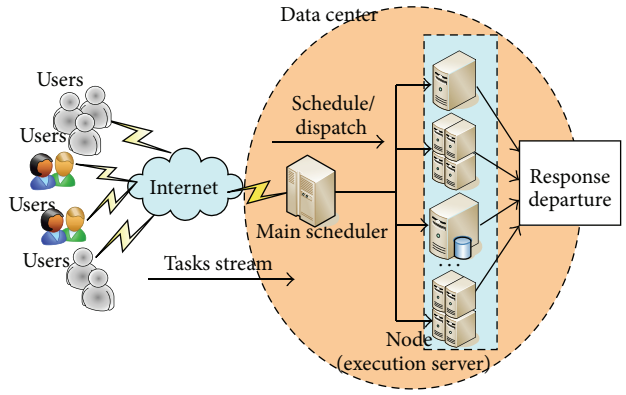

FIGURE 1: Cloud center architecture and the service model.

the general problem of resource deployment within cloud computing.

However, none of the above literature considers that the main scheduler and the execution servers have their own task queuing, that an execution server with different speeds will have different processing times, and that each execution server has a different probability of being dispatched given different arrival rates of tasks. As a result, a queuing model used to model a heterogeneous data center for performance analysis should characterize the service process within the cloud center as well as the infrastructural architecture of a data center.

3. Queuing Model for Performance Analysis

In this section, we present a complex queuing model with two concatenated queuing systems to characterize the service process and the heterogeneity of the infrastructural architecture in order to analyze the performance of a heterogeneous data center.

3.1. Cloud Center Architecture and the Service Model. In the cloud computing environment, a cloud computing provider builds a data center as the infrastructure to provide service for customers [21]. Figure 1 illustrates the architecture of a heterogeneous data center and its service model. The heterogeneous data center consists of large-scale servers of different generations. It includes the master server, which works as a main scheduler to allocate computing resources for a task or to dispatch a node to execute a task, and large-scale computing nodes, which are multicore servers with different speeds that work as execution servers.

All customers submit requests to the data center for services through the Internet, which has been deployed by the users or supplied by the cloud providers. When the tasks (user requests) arrive at the data center, the main scheduler allocates resources for the task or dispatches nodes; the task is then distributed to the execution servers to be executed. The main scheduler has a waiting buffer with a finite capacity, in which it saves waiting tasks that cannot be scheduled immediately. The buffer's finite capacity means that some requests cannot enter the waiting buffer and will instead leave the system immediately after arriving. Each execution server receives all requests from the main scheduler and maintains its own task queue.

After a task has been completed, it leaves the system and the result is sent back to the customer.

3.2. Other Queuing Models for Reference

3.2.1. Multiple-Server Queue with the Same Service Rate. A queuing model containing multiple servers with the same service rate is shown in Figure 2(a). Using this queuing model, the authors treated a cloud computing data center as an $M/G/m/m+r$ queuing system in [11], as an $M/M/r/k$ queuing system in [19], and as an $M/M/m$ queuing system in [12, 13, 18]. Because each server has the same service rate, this queuing model treats each server in the data center the same and assigns it the same dispatching probability $p = 1/n$.

This queuing model simplifies analysis and makes it easier to deduce an equation for the important performance metrics (e.g., the mean response time, mean number of tasks in the system, and mean waiting time). However, the process presented in this model does not apply to the service process in a cloud data center. It ignores the scheduling activity of the main server and the different speeds of execution servers of different generations.

3.2.2. Multiple-Server Queue with Different Service Rates. A queuing model including multiple servers with different service rates is shown in Figure 2(b). We can use this queuing system to model a heterogeneous data center that includes multicore servers with different speeds. Each execution server in the data center has a different service rate μ_i and dispatching probability p_i. We assume that the conditions are $N = 2$ and task arrival rate λ. The state-transition-probability diagram for $N = 2$ is shown in Figure 3.

When a task arrives and there are two servers that are idle, the scheduler will select one of them to execute the task. The selection probability of the number 1 server is p_1 and that of the number 2 server is p_2 ($p_2 = 1 - p_1$). The state of S_0 indicates that there is no task in the queuing system; S_{10} indicates that the task has been distributed to the number 1 server, and the number 2 server is idle; S_{01} indicates that the task has been distributed to the number 2 server, and the number 1 server is idle; S_2 indicates that there are two tasks in the system, and each has been distributed to one of the two servers. $S_3, S_4, \ldots, S_n$ indicate that there are $3, 4, \ldots, n$ tasks in the system, and two of them are in the two execution servers. Let μ_1 and μ_2 denote the service rates of the number 1 and number 2 servers, respectively. The system's service rate is $\mu = \sum_{i=1}^{2} \mu_i$. Figure 3 shows how this model calculates probability when $\rho = \lambda/\mu$ and $\alpha = \mu_2/\mu_1$ [22]:

$$\pi_{01} = \frac{\rho}{1+2\rho} \cdot \frac{1+\alpha}{\alpha} (\rho + p_2) \pi_0, \quad (E1)$$

$$\pi_{10} = \frac{\rho}{1+2\rho} \cdot (1+\alpha)(\rho + p_1) \pi_0, \quad (E2)$$

$$\pi_0 = \frac{1-\rho}{1+\rho\left[1+\left(1+\alpha^2\right)\rho - \left(1-\alpha^2\right)p_1\right]/\alpha(1+2\rho)}. \quad (E3)$$

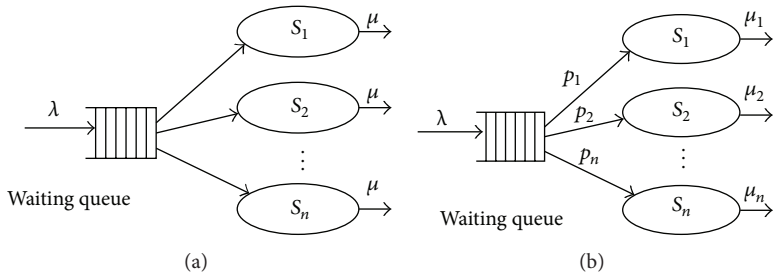

Figure 2: Other queuing models for reference.

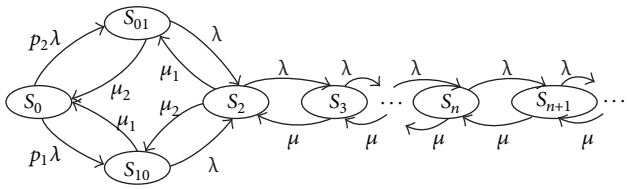

Figure 3: State-transition-probability diagram for $N = 2$.

From $(E1)$–$(E3)$, we can see that if $N \geq 3$, it can use $\pi_n = \rho\pi_{n-1}$ to calculate the probability of each state in Figure 3. The state-transition-probability diagram for $N \geq 3$ will be more complex.

Although it considers the heterogeneity of data centers, this model still does not accurately describe the service process in the cloud center; also, if $N \geq 3$, the performance analysis will be more difficult.

3.3. Queuing Model for Performance Analysis. A queuing model of heterogeneous data centers with a large number of execution servers with multiple cores of different generations is shown in Figure 4. The two concatenated queuing systems—the scheduler queuing system and the execution queuing system—make up the complex queuing model that characterizes the heterogeneity of the data center and of the service processes involved in cloud computing. This model uses a monitor to obtain performance metrics—including the task number in the queuing, L_q, the mean waiting time of a task, W_q, and the utilization of the execution server, ρ. The model then sends this information to the balance controller and optimizes performance by proposing an optimal strategy based on these metrics. This is a project we will consider doing in future.

The complex queuing model is presented as follows.

The master server works as the main scheduler and maintains scheduler queuing for all requests from all users. Since the process of allocating resources for tasks in the cloud computing environment should consider all resources in the data center, the master server is treated as an $M/M/1/k$ queuing system with finite capacity k.

Each node in a data center works as an execution server, which is a multicore server that has m_i identical cores with the core execution speed f_i (GIPS, measured by giga instructions per second). Since each multicore execution server can parallel-process multiple tasks, it is treated as an $M/M/c$ queuing system.

The stream of tasks is a Poisson process with arrival rate λ (the task interarrival times are independent and identically distributed exponential random variables with a rate of $1/\lambda$). The related notations of the system parameters are listed in the Notations.

Since arrivals are independent of the queuing state and the scheduler queuing system is a finite capacity queuing model, the blocking probability of the scheduling system p_b will be greater than or equal to 0 ($p_b \geq 0$) and $\lambda \geq \lambda_e$, to yield the following equations:

$$\lambda_e = \lambda\left(1 - p_b\right), \tag{1}$$

$$\lambda_i = \lambda_e p_i, \quad 1 \leq i \leq n, \ \sum p_i = 1, \tag{2}$$

$$\mu_i = \frac{1}{\left(\bar{I}/f_i\right)} = \frac{f_i}{\bar{I}}, \tag{3}$$

$$\rho_i = \frac{\lambda_i}{\left(C_i\mu_i\right)} = \frac{\left(\lambda_i\bar{I}\right)}{\left(C_i f_i\right)}, \quad \rho_i \in \left[\delta_1, \delta_h\right]. \tag{4}$$

4. Performance Metrics of the Queuing Model

In this section, we will use the complex queuing model with the two concatenated queuing systems proposed in Section 3.3 to analyze the performance problems of a heterogeneous data center in cloud computing. We consider the performance metrics of the scheduler queuing system, the execution queuing systems, and the entire queuing system.

4.1. The Scheduler Queuing System. Since management of computing resources and scheduling tasks across entire data center and cloud environments are done by a unified resources management platform (e.g., Mesos [23] and YARN [24]) and the system should be accompanied by finite capacity, the master server is treated as an $M/M/1/k$ queuing system with finite capacity k. The state-transition-probability diagram for the scheduler queuing model is shown in Figure 5.

If the scheduler utilization $\rho_s = \lambda/\mu_s$ and $\rho_s < 1$, then the probability that i tasks are in the queuing system $\pi_i^{(s)}$ is [22]

$$\pi_0^{(s)} = \frac{1 - \rho_s}{1 - \rho_s^{k+1}},$$
$$\pi_i^{(s)} = \frac{1 - \rho_s}{1 - \rho_s^{k+1}}\rho_s^i, \quad i = 1, 2, \ldots, k. \tag{5}$$

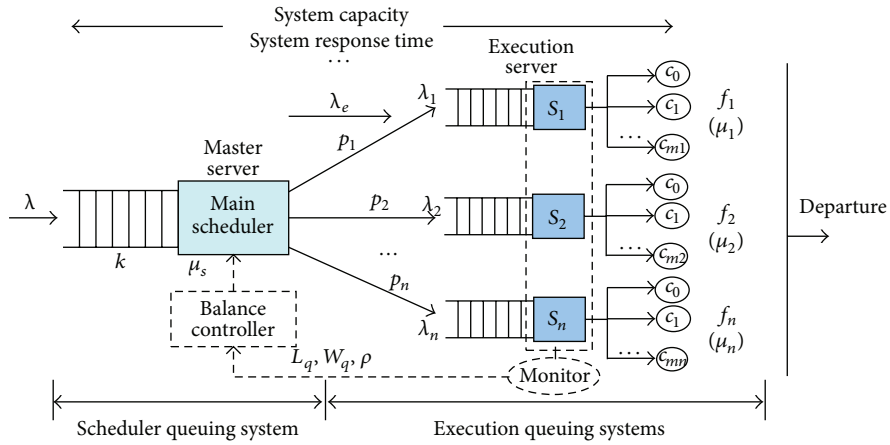

Figure 4: Queuing model of heterogeneous data centers.

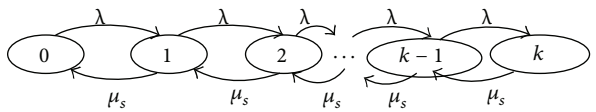

Figure 5: State-transition-probability diagram for the scheduler queuing model.

If $i \geq k$, the task request could not enter the queuing system. The blocking probability p_b is

$$P_b = \pi_k^{(s)} = \frac{1-\rho_s}{1-\rho_s^{k+1}}\rho_s^k. \tag{6}$$

From (1) and (6), the master server throughput rate (the effective arrival rate of the execution queuing systems) λ_e is

$$\lambda_e = \lambda\left(1-p_b\right) = \lambda\frac{1-\rho_s^k}{1-\rho_s^{k+1}}. \tag{7}$$

The tasks waiting in the queue $\overline{C}_w^{(s)}$ and the tasks scheduled by the main scheduler $\overline{C}_{\text{sch}}^{(s)}$ are, respectively,

$$\begin{aligned}\overline{C}_w^{(s)} &= \sum_{i=0}^{k-1} k\pi_{i+1}^{(s)} = \rho_s^2\pi_0^{(s)}\sum_{i=1}^{k-1} k\rho_s^{k-1},\\ \overline{C}_{\text{sch}}^{(s)} &= \left(1-\pi_0^{(s)}\right) = \frac{\rho_s-\rho_s^{k+1}}{1-\rho_s^{k+1}}.\end{aligned} \tag{8}$$

From (8), the mean task number (including the tasks waiting in the queue and the tasks scheduled by the main scheduler) in the scheduler queuing system $\overline{C}_{\text{total}}^{(s)}$ is

$$\overline{C}_{\text{total}}^{(s)} = \overline{C}_w^{(s)} + \overline{C}_{\text{sch}}^{(s)} = \frac{\rho_s}{1-\rho_s} - \frac{(k+1)\,\rho_s^{k+1}}{1-\rho_s^{k+1}}. \tag{9}$$

Using (7) and (9), $\rho_s = \lambda/\mu_s$; applying Little's formulas (the response time t = the tasks number in the system/the tasks arrival rate), the mean task response time of the scheduler queuing system $\overline{T}^{(s)}$ is

$$\begin{aligned}\overline{T}^{(s)} &= \frac{\overline{C}_{\text{total}}^{(s)}}{\lambda_e} = \frac{1-(k+1)\,\rho_s^k + k\rho_s^{k+1}}{\mu_s\left(1-\rho_s\right)\left(1-\rho_s^k\right)}\\ &= \frac{1}{\mu_s-\lambda} - \frac{k\lambda^k}{\mu_s^{k+1}-\lambda^k}.\end{aligned} \tag{10}$$

4.2. The Execution Queuing Systems. Assume that there are n execution servers in the data center. Each execution server has C_i cores and can be treated as an $M/M/C_i$ queuing system.

The utilization of one core in the ith execution server ρ_{i0} is

$$\rho_{i0} = \frac{\lambda_i}{\mu_i} = \frac{\lambda_i\overline{I}}{f_i}. \tag{11}$$

From the state-transition-probability diagram for an $M/M/C_i$ queuing system [22], let $\pi_{i,m}^{(e)}$ indicate the probability that there are m task requests (including waiting in the queue and being executed) in the execution queuing system of the execution server S_i:

$$\pi_{i,m}^{(e)} = \begin{cases}\pi_{i,0}^{(e)}\cdot\dfrac{\rho_{i0}^m}{m!} = \pi_{i,0}^{(e)}\cdot\dfrac{C_i^m}{m!}\rho_i^m, & 0 \leq m < C_i\\ \pi_{i,0}^{(e)}\cdot\dfrac{\rho_{i0}^m}{C_i!C_i^{m-C_i}} = \pi_{i,0}^{(e)}\cdot\dfrac{C_i^{C_i}}{C_i!}\rho_i^m, & m \geq C_i.\end{cases} \tag{12}$$

Under the stability condition, there are constraints as follows:

(1) $\sum_{m=0}^{\infty}\pi_{i,m}^{(e)} = \{\sum_{m=0}^{C_i-1}(1/m!)\rho_{i0}^m + (C_i/C_i!)\rho_{i0}^{C_i-1}(\rho_i/(1-\rho_i))\}\pi_{i,0} = 1$.

(2) $\rho_i < 1$.

We can derive the probability that there are 0 task requests in the execution queuing system $\pi_{i,0}^{(e)}$:

$$\pi_{i,0}^{(e)} = \left(\sum_{m=0}^{C_i-1} \frac{\rho_{i0}^m}{m!} + \frac{\rho_{i0}^{C_i}}{C_i!} \cdot \frac{1}{1-\rho_i} \right)^{-1}. \tag{13}$$

The probability that a newly arrived task request from the main scheduler to the ith execution server will be executed immediately q_i is

$$\begin{aligned} q_i &= 1 - \sum_{m=C_i}^{\infty} \pi_{i,m}^{(e)} = 1 - \sum_{m=C_i}^{\infty} \frac{C_i^{C_i}}{C_i!} \rho_i^m \pi_{i,0}^{(e)} \\ &= \frac{C_i - \rho_{i0} - C_i \pi_{i,C_i}^{(e)}}{C_i - \rho_{i0}}. \end{aligned} \tag{14}$$

The mean probability of a newly arrived task request entering the execution queuing systems $\overline{q}^{(e)}$ is

$$\overline{q}^{(e)} = \sum_{i=1}^{n} p_i q_i. \tag{15}$$

In the ith execution server, the tasks waiting in the queue $\overline{C}_{w\_i}^{(e)}$ and the tasks executed by the ith execution server $\overline{C}_{\text{exc}\_i}^{(e)}$ are, respectively,

$$\begin{aligned} \overline{C}_{w\_i}^{(e)} &= \sum_{m=C_i}^{\infty} (m - C_i) \pi_{i,m}^{(e)}, \\ \overline{C}_{\text{exc}\_i}^{(e)} &= \sum_{m=0}^{C_i} m \pi_{i,m}^{(e)} + C_i \sum_{m=C_i+1}^{\infty} \pi_{i,m}^{(e)}. \end{aligned} \tag{16}$$

From (16), the mean task number (including the tasks waiting in the queue and the tasks executed by the execution server) in the ith execution server $\overline{C}_{\text{total}\_i}^{(e)}$ is

$$\begin{aligned} \overline{C}_{\text{total}\_i}^{(e)} &= \overline{C}_{w\_i}^{(e)} + \overline{C}_{\text{exc}\_i}^{(e)} \\ &= \pi_{i,0}^{(e)} \cdot \frac{\rho_{i0}^{C_i+1}}{(C_i - 1)! (C_i - \rho_{i0})^2} + \rho_{i0}. \end{aligned} \tag{17}$$

Applying Little's formulas, the mean task response time of the ith execution queuing systems $\overline{T}_i^{(e)}$ and the mean waiting time for execution of a task in the ith execution queuing systems $\overline{W}_i^{(e)}$ are, respectively,

$$\begin{aligned} \overline{T}_i^{(e)} &= \frac{\overline{C}_{\text{total}\_i}^{(e)}}{\lambda_i} = \frac{\overline{I}}{f_i} \left(1 + \pi_{i,0}^{(e)} \cdot \frac{\rho_{i0}^{C_i}}{(C_i - 1)! (C_i - \rho_{i0})^2} \right), \\ \overline{W}_i^{(e)} &= \frac{\overline{C}_{w\_i}^{(e)}}{\lambda_i} = \frac{1}{\lambda_i} \sum_{m=C_i}^{\infty} (m - C_i) \pi_{i,m}^{(e)} \\ &= \pi_{i,0}^{(e)} \cdot \frac{\overline{I} \cdot \rho_{i0}^{C_i}}{f_i (C_i - 1)! (C_i - \rho_{i0})^2}. \end{aligned} \tag{18}$$

The mean response time of the execution queuing systems $\overline{T}^{(e)}$ for a group of n execution servers in the data center is

$$\overline{T}^{(e)} = \sum_{i=1}^{n} P_i \overline{T}_i^{(e)}. \tag{19}$$

The mean waiting time of the execution queuing systems $\overline{W}^{(e)}$ in a group of n execution servers in the data center is

$$\overline{W}^{(e)} = \sum_{i=1}^{n} P_i \overline{W}_i^{(e)}. \tag{20}$$

4.3. The Performance Metrics of the Entire Queuing System. In order to analyze the performance of heterogeneous data centers in cloud computing, we consider the main performance metrics of a complex queuing system, which consists of two concatenated queuing systems, as follows.

(i) Response Time. Based on analyzing the two concatenated queuing systems, the equilibrium response time in heterogeneous data centers is the sum of the response time of the two phases. The response time equation can be formulated as

$$\overline{T} = \overline{T}^{(s)} + \overline{T}^{(e)} = \overline{T}^{(s)} + \sum_{i=1}^{n} P_i \overline{T}_i^{(e)}. \tag{21}$$

(ii) Waiting Time. The mean waiting time for a task request is

$$\begin{aligned} \overline{W} &= \overline{W}^{(s)} + \overline{W}^{(e)} = \frac{\overline{C}_W^{(s)}}{\lambda_e} + \sum_{i=1}^{n} P_i \overline{W}_i^{(e)} \\ &= \frac{\rho_s}{\mu_s} \left(\frac{1}{1-\rho_s} - \frac{k \rho_s^{k-1}}{1-\rho_s^k} \right) + \sum_{i=1}^{n} P_i \overline{W}_i^{(e)}. \end{aligned} \tag{22}$$

(iii) Loss (Blocking) Probability. Since the scheduler queuing system is a finite capacity queuing model, blocking will occur:

$$P_1 = P_b. \tag{23}$$

From (6), the loss (blocking) probability of the system is related to the finite capacity of the scheduler queuing system k and the scheduling rate of the main scheduler server μ_s. The parameters of k or μ_s can be adjusted to obtain different loss (blocking) probabilities.

(iv) Probability of Immediate Execution. Here, the probability of immediate execution indicates the probability of a task request being executed immediately by an execution server after being scheduled by the scheduler server and sent to the execution server:

$$\overline{q} = \overline{q}^{(e)}. \tag{24}$$

(v) Execution Server Utilization Optimization Model Based on the Arrival Rate λ_e. Our analytical model can be used to

optimize the performance for a data center. This is one of the significant applications of the model.

Based on the utilization threshold $[\delta_l, \delta_h]$ of the execution servers, we can determine the number of execution servers and the utilization of each execution server in the system, by using the calculation model, which is based on our analytical model, to configure the execution servers in the data center to deal with tasks with the arrival rate λ_e. It can minimize the mean response time of the system.

Assume the use of FCFS as the scheduling strategy, and set the heterogeneous data center as a group of n heterogeneous execution servers $S_1, S_2, \ldots, S_n$ with multicores $C_1, C_2, \ldots, C_n$ and execution speeds of $f_1, f_2, \ldots, f_n$. The execution server utilization optimization model based on arrival rate λ_e can be formulated as follows:

$$
\begin{aligned}
\underset{\{\lambda_e,\bar{I},f_1,\ldots,f_n,C_1,\ldots,C_n\}}{\text{Min}} \quad & \sum_{i=1}^{n} Z_i P_i \overline{T}_i^{(e)} \\
\text{S.T.} \quad & (1)\ \rho_i = \begin{cases} 0, & Z_i = 0 \\ \in [\delta_l, \delta_h], & Z_i = 1, \end{cases} \quad 1 \le i \le n \\
& (2)\ \lambda_i = \frac{\rho_i C_i f_i}{\bar{I}}, \quad 1 \le i \le n \\
& (3)\ \sum_{i=1}^{n} \lambda_i = \lambda_e \\
& (4)\ P_i = \frac{\rho_i C_i f_i}{\bar{I}\lambda_e}, \quad 1 \le i \le n \\
& (5)\ \sum_{i=1}^{n} P_i = 1 \\
& (6)\ \overline{T}_i^{(e)} = \frac{\bar{I}}{f_i}\left[1 + \left(\sum_{m=0}^{C_i-1} \frac{(C_i\rho_i)^m}{(C_i-1)!} + \frac{1}{(C_i-1)!}\cdot\frac{(C_i\rho_i)^{C_i}}{1-\rho_i}\right)^{-1} \cdot \frac{(C_i\rho_i)^{C_i-1}\rho_i}{C_i!\,(1-\rho_i)}\right], \quad 1 \le i \le n \\
& (7)\ Z_i \in \{0, 1\}, \quad 1 \le i \le n.
\end{aligned}
\tag{25}
$$

The utilization of the execution servers will determine mean response time, occupied time, and resource capacity; therefore, the execution server utilization optimization model based on arrival rate λ_e can be used to analyze the problem of optimal resource cost or the problem of energy efficiency. We will accomplish this research in future work.

In this paper, we just consider the special case—the condition of $\lambda_e \in [(\delta_l/\bar{I})\sum_{i=1}^{m} C_i f_i, (\delta_h/\bar{I})\sum_{i=1}^{m} C_i f_i]$. In a small case, we can use a numerical method to gain the values of ρ_i for optimal performance. Under the condition of $\lambda_e \in [(\delta_l/\bar{I})\sum_{i=1}^{m} C_i f_i, (\delta_h/\bar{I})\sum_{i=1}^{m} C_i f_i]$, the parameter of Z_i in (25) is $Z_i = 1$; (25) can be rewritten as follows:

$$
\begin{aligned}
\underset{\{\lambda_e,\bar{I},f_1,\ldots,f_n,C_1,\ldots,C_n\}}{\text{Min}} \quad & \sum_{i=1}^{n} \frac{\rho_i C_i}{\lambda_e}\left[1 + \left(\sum_{m=0}^{C_i-1} \frac{(C_i\rho_i)^m}{(C_i-1)!} + \frac{1}{(C_i-1)!}\cdot\frac{(C_i\rho_i)^{C_i}}{1-\rho_i}\right)^{-1} \cdot \frac{(C_i\rho_i)^{C_i-1}\rho_i}{C_i!\,(1-\rho_i)}\right] \\
\text{S.T.} \quad & (1)\ \delta_l \le \rho_i \le \delta_h, \quad 1 \le i \le n \\
& (2)\ \sum_{i=1}^{n} \frac{\rho_i C_i f_i}{\bar{I}} = \lambda_e \\
& (3)\ \sum_{i=1}^{n} \frac{\rho_i C_i f_i}{\bar{I}\lambda_e} = 1.
\end{aligned}
\tag{26}
$$

Table 1: Hosts and VMs resources setting.

HostID (PM)	Setting: ST1			Setting: ST2		
	CoreNum. (VM Num., C_i)	Computing power (GIPS, f_i)	VMs' setting	CoreNum. (VM Num., C_i)	Computing power (GIPS, f_i)	VMs' setting
Host#1	2	2	1 Core/2 G	2	2	1 Core/2 G
Host#2	4	2	1 Core/2 G	2	2	1 Core/2 G
Host#3	6	2	1 Core/2 G	2	2	1 Core/2 G
Host#4	8	2	1 Core/2 G	4	2	1 Core/2 G
Host#5	10	2	1 Core/2 G	4	2	1 Core/2 G
Host#6	12	2	1 Core/2 G	4	2	1 Core/2 G
Host#7	14	2	1 Core/2 G	8	4	1 Core/4 G
Host#8	16	2	1 Core/2 G	8	4	1 Core/4 G
Host#9	18	2	1 Core/2 G	12	5	1 Core/5 G
Host#10	20	2	1 Core/2 G	12	5	1 Core/5 G

5. Numerical Validation

In order to validate the performance analysis equations presented above, we built a heterogeneous cloud computing data center farm in the CloudSim platform and a tasks generator with using the discrete event tool MATLAB. In the numerical examples and the simulation experiments we present, the parameters are illustrative and can be changed to suit the cloud computing environment.

5.1. Performance Simulation. In this section, we describe the simulation experiments that we conducted to validate the performance analysis equations of the performance metrics. In all cases, the mean number of instructions of a task to be executed by the execution server was $\overline{I} = 1$ (giga instructions). In the CloudSim, we considered a heterogeneous data center with a group of $n = 10$ multicore execution servers Host#1, Host#2,..., Host#10 and a main scheduler server Ms. The traffic intensity of the main scheduler server was $\rho_s = 0.80$. The finite capacity of the scheduler $k = \lceil 0.1\lambda \rceil$. The different settings of the Host#i ($1 \le i \le 10$) are shown in Table 1. The allocation policy of the VM-scheduler is space-shared, which makes one task in one VM. The total computing ability (computing power) of these two groups of execution servers with different settings was the same.

In Table 1, the heterogeneous data centers of ST1 and ST2 are configured with 10 PMs and 110 VMs and 10 PMs and 58 VMs, respectively. The total computing powers of ST1 and ST2 are all 220 (GIPS). The hardware environment is 2×Dell PowerEdge T720 (2×CPUs: Xeon 6-core E5-2630 2.3 G, 12 cores in total).

The interarrival times of task requests were independent and followed an exponential distribution with arrival rate λ (the number of task requests per second). The cloudlet information is created by tasks generator, which generates task arrival interval time, task scheduling time, and task length.

In the experiments, we assigned each multicore execution server—which had a dispatched probability according to its core speeds or set traffic intensity—to each multicore execution server to control a task. The rule for setting the dispatched probability and the traffic intensity, which can be changed by administrators, was as follows.

(i) Setting the Dispatched Probability. Consider

$$P_i = \frac{C_i f_i}{\sum_{i=1}^{10} C_i f_i}, \quad 1 \le i \le 10. \tag{27}$$

Under this pattern, the traffic intensity of each execution server was the same and may have exceeded the scope of the utilization threshold $[\delta_l, \delta_h]$. Using these two groups of execution servers (ST1 and ST2), the dispatched probability of each execution server was set using (27).

Simulation Experiments 1. We have the following:

(1) Tasks number: cloudletsNo = 100000, tasks length (GIPS): $-\log(\text{rand}(1, \text{cloudletNo}))/(f_i/\overline{I})$, and λ: {157.5, 160.5, 163.5, 166.5, 169.5, 172.5, 175.5, 178.5, 181.5, 184.5, 187.5}.

(2) The main scheduler computing power: $\mu_s = \lambda/\rho_s$, hosts (PMs) and VMs setting: ST1 and St2 (shown in Table 1), and the probability of host (PM) dispatched: P_i (27).

(3) Process: simulation experiment times: 30. In every time, we recorded the loss tasks number in the main scheduler, the immediate execution tasks number (if the arrival time equates to start time), the response time, and the waiting time of each host. After 30 times' experiments, we calculated the average values of the loss (blocking) probability $P_l(P_b)$, the probability of immediate execution $\overline{q}$, the mean response time $\overline{T}$, and the mean waiting time $\overline{W}$. Then, we calculated the 95% confidence interval (95% C.I.) for each metric to validate these metrics in the analysis model.

(4) The analytical results are calculated by using the analytical model with (21)–(24). The settings of each

Table 2: The simulations and analytical results of $P_l(P_b)$.

λ	Host set	Simulation 95% C.I. for $P_l(P_b)$			Analytical results
		Mean	Lower	Upper	
157.5	ST1	0.0058	0.005683	0.005917	0.005759
	ST2	0.00573	0.005491	0.005969	0.005759
172.5	ST1	0.00452	0.004404	0.004636	0.004586
	ST2	0.00462	0.004504	0.004736	0.004586
187.5	ST1	0.00295	0.002853	0.003047	0.002916
	ST2	0.00296	0.002842	0.003078	0.002916

Table 3: The simulations and analytical results of $\overline{q}$.

λ	Host set	Simulation 95% C.I. for $\overline{q}$			Analytical results
		Mean	Lower	Upper	
157.5	ST1	0.80593	0.803737	0.808123	0.806097
	ST2	0.72404	0.723025	0.725055	0.724773
172.5	ST1	0.6867	0.683968	0.689432	0.688503
	ST2	0.60165	0.599881	0.603419	0.60225
187.5	ST1	0.526	0.524375	0.527625	0.525132
	ST2	0.44677	0.445745	0.447795	0.447063

Table 4: The simulations and analytical results of $\overline{W}$.

λ	Host set	Simulation 95% C.I. for $\overline{W}$			Analytical results
		Mean	Lower	Upper	
157.5	ST1	0.05987	0.059035	0.060705	0.060115
	ST2	0.08227	0.081676	0.082864	0.082617
172.5	ST1	0.09591	0.094468	0.097352	0.096286
	ST2	0.1249	0.123706	0.126094	0.12491
187.5	ST1	0.17872	0.174778	0.182662	0.178457
	ST2	0.21112	0.207124	0.215116	0.213743

Table 5: The simulations and analytical results of $\overline{T}$.

λ	Host set	Simulation 95% C.I. for $\overline{T}$			Analytical results
		Mean	Lower	Upper	
157.5	ST1	0.56496	0.565194	0.56374	0.56618
	ST2	0.35094	0.350361	0.351519	0.351333
172.5	ST1	0.60029	0.598638	0.601942	0.600923
	ST2	0.39317	0.391785	0.394555	0.393184
187.5	ST1	0.6829	0.678949	0.686851	0.682724
	ST2	0.47903	0.474915	0.483145	0.481646

parameter are λ: {157.5, 160.5, 163.5, 166.5, 169.5, 172.5, 175.5, 178.5, 181.5, 184.5, 187.5}, $\rho_s = 0.80$, $k = \lceil 0.1\lambda \rceil$, $\overline{I} = 1$, Pi (27), and C_i and f_i shown in Table 1.

The simulations and the analytical results (λ: {157.5, 172.5, 187.5}) are shown in Tables 2–5.

The comparisons between the analytical and the simulations results (λ: {157.5, 160.5, 163.5, 166.5, 169.5, 172.5, 175.5, 178.5, 181.5, 184.5, 187.5}) are shown in Figures 6–9.

By comparing the simulation results obtained from CloudSim and the analytical results calculated by using the model, we can observe that the analytical results of $P_l(P_b)$, $\overline{q}$, $\overline{W}$, and $\overline{T}$ are all in the range of 95% C.I., which are shown in Tables 2–5 and Figures 6–9. It demonstrates that the analytical results are in agreement with the CloudSim simulation results. It also confirms that the metrics of $P_l(P_b)$, $\overline{q}$, $\overline{W}$, and $\overline{T}$ in the analytical model can be trusted under the confidence level of 95%.

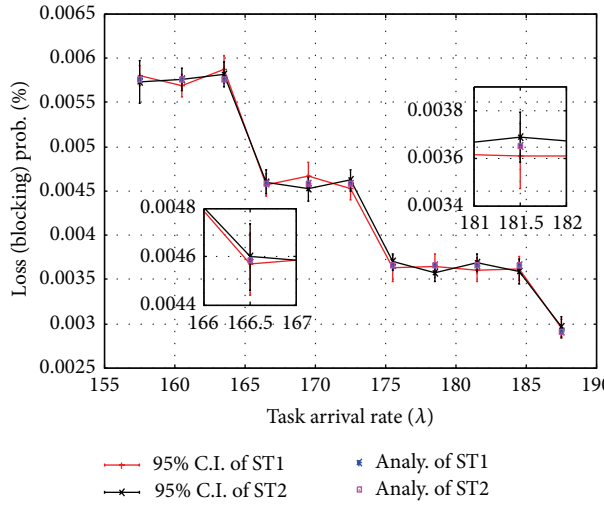

Figure 6: Simulation 95% C.I. for Pl versus analytical result.

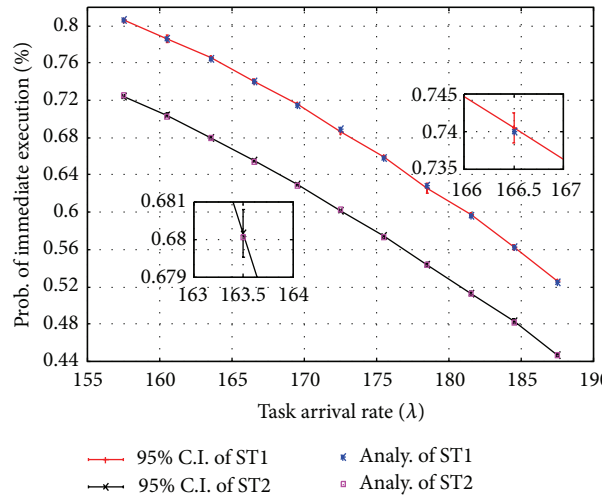

Figure 7: Simulation 95% C.I. for $\overline{q}$ versus analytical result.

Simulation Experiments 2. Use the same dataset (task number: 100000, ST1 and ST2) as the simulation experiments 1 to complete 30 times' experiments under the arrival rate λ = 187.5. From this experiment, we recorded the response and waiting times of each host in ST1 and ST2 ($\overline{T}_i^{(e)}$ and $\overline{W}_i^{(e)}$, $1 \le i \le 10$). Then, we can get the 95% confidence interval (95% C.I.) of $\overline{T}_i^{(e)}$ and $\overline{W}_i^{(e)}$ to validate these metrics in the model.

The analytical results are gotten by using (18) and (19).

In Tables 6 and 7, we demonstrate the simulation results and the 95% C.I. of $\overline{T}_i^{(e)}$ and $\overline{W}_i^{(e)}$ ($1 \le i \le 10$) of every host in ST1 and ST2 with a task request arrival rate λ = 187.5.

Table 6: The simulations and analytical results of each host in ST1.

HostID	Simulation 95% C.I. for $\overline{T}_i^{(e)}$			Analytical results of $\overline{T}_i^{(e)}$	Simulation 95% C.I. for $\overline{W}_i^{(e)}$			Analytical results of $\overline{W}_i^{(e)}$
	Mean	Lower	Upper		Mean	Lower	Upper	
Host#1	1.808109	1.730226	1.885993	1.79946	1.308227	1.231192	1.385263	1.29946
Host#2	1.090659	1.064057	1.117261	1.073279	0.590882	0.565038	0.616725	0.573279
Host#3	0.840732	0.82654	0.854924	0.845942	0.341209	0.327489	0.354929	0.345942
Host#4	0.737109	0.730784	0.743434	0.738017	0.237314	0.231432	0.243195	0.238017
Host#5	0.674368	0.66885	0.679887	0.676187	0.174809	0.169671	0.179947	0.176187
Host#6	0.634818	0.630513	0.639123	0.636685	0.135545	0.13165	0.139441	0.136685
Host#7	0.607036	0.603298	0.610775	0.609575	0.107477	0.104036	0.110919	0.109575
Host#8	0.589295	0.58681	0.59178	0.59	0.089395	0.087273	0.091518	0.09
Host#9	0.576559	0.572962	0.580156	0.57532	0.076586	0.073578	0.079594	0.07532
Host#10	0.565364	0.563253	0.567474	0.563981	0.065318	0.063482	0.067154	0.063981

Table 7: The simulations and analytical results of each host in ST2.

HostID	Simulation 95% C.I. for $\overline{T}_i^{(e)}$			Analytical results of $\overline{T}_i^{(e)}$	Simulation 95% C.I. for $\overline{W}_i^{(e)}$			Analytical results of $\overline{W}_i^{(e)}$
	Mean	Lower	Upper		Mean	Lower	Upper	
Host#1	1.846295	1.774547	1.918044	1.79946	1.347014	1.275931	1.418096	1.29946
Host#2	1.79655	1.742504	1.850596	1.79946	1.296655	1.243793	1.349516	1.29946
Host#3	1.806673	1.742178	1.871167	1.79946	1.307555	1.244333	1.370776	1.29946
Host#4	1.082836	1.062659	1.103014	1.073279	0.582073	0.562148	0.601997	0.573279
Host#5	1.065777	1.042977	1.088577	1.073279	0.5667	0.544379	0.589021	0.573279
Host#6	1.055886	1.03763	1.074143	1.073279	0.556777	0.538945	0.574609	0.573279
Host#7	0.369245	0.367006	0.371485	0.369009	0.119423	0.117318	0.121527	0.119009
Host#8	0.368186	0.365717	0.370656	0.369009	0.118227	0.115911	0.120543	0.119009
Host#9	0.255173	0.254154	0.256191	0.254674	0.055091	0.054138	0.056044	0.054674
Host#10	0.254464	0.253357	0.255571	0.254674	0.054359	0.053383	0.055335	0.054674

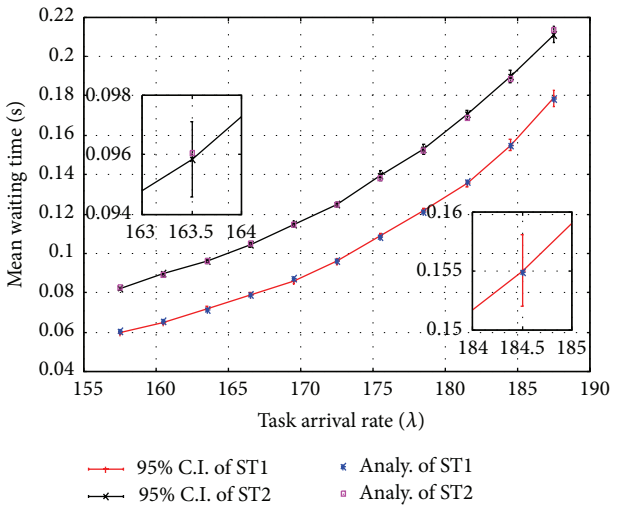

Figure 8: Simulation 95% C.I. for $\overline{W}$ versus analytical result.

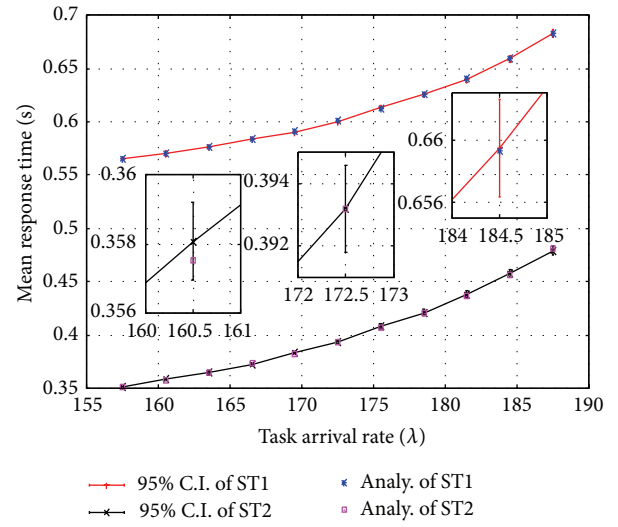

Figure 9: Simulation 95% C.I. for $\overline{T}$ versus analytical result.

The comparisons between the analytical and the simulations results of the hosts are shown in Figures 10-11.

As shown in Tables 6 and 7 and Figures 10 and 11, we can observe that the analytical results of $\overline{T}_i^{(e)}$ and $\overline{W}_i^{(e)}$ for all the hosts in ST1 and ST2 are all in the range of 95% C.I. It demonstrates that the analytical results are in agreement with the CloudSim simulation results and the hosts' analytical model can be trusted under the confidence level of 95%.

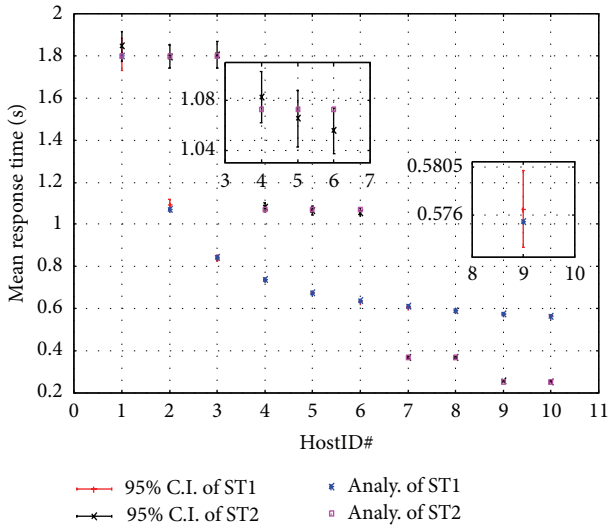

Figure 10: Simulation 95% C.I. for $\overline{T}_i^{(e)}$ versus analytical result.

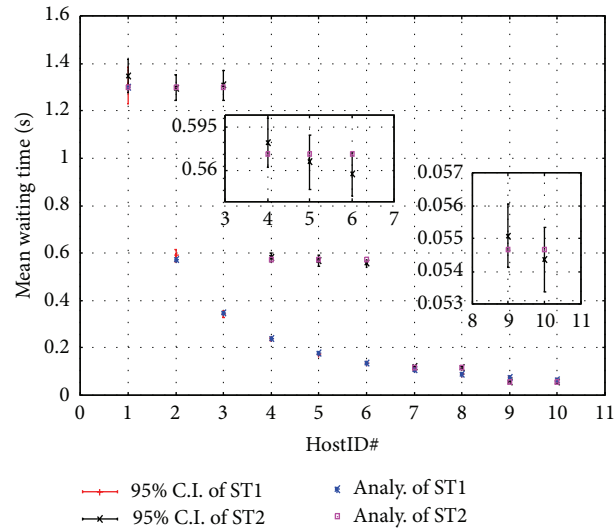

Figure 11: Simulation 95% C.I. for $\overline{W}_i^{(e)}$ versus analytical result.

After comparing the results, we also conclude that

(1) the relative errors of the simulation and the analytics are less than 3.5%,

(2) under the same arrival rate λ_e, the performance metrics will be different with different settings despite having the same computing abilities.

(ii) Setting the Traffic Intensity. The traffic intensity/utilization ρ_i $(1 \le i \le 10)$ was set in the scope of $[\delta_l, \delta_h]$ (setting $\delta_l = 0.65$ and $\delta_h = 0.85$). Under this pattern, the scope of the task request arrival rate of the ith execution server will be $\lambda_i \in [\delta_l C_i f_i/\overline{I}, \delta_h C_i f_i/\overline{I}]$ $(1 \le i \le 10)$ according to (4). A group of m $(m \le n)$ multicore execution servers can handle the effective arrival rate λ_e':

$$\lambda_e' = \sum_{i=1}^{m} \lambda_i \Longrightarrow \frac{\delta_l}{\overline{I}} \sum_{i=1}^{m} C_i f_i \le \lambda_e' \le \frac{\delta_h}{\overline{I}} \sum_{i=1}^{m} C_i f_i. \tag{28}$$

Simulation Experiments 3. Let us consider using the same hosts set to deal with the same arrival rate $\lambda = 172$ ((28), $172 \in [220 \times 0.65,\ 220 \times 0.85]$) at designated rates of traffic intensity/utilization ρ_i $(1 \le i \le 10)$ setting. The different ρ_i $(1 \le i \le 10)$ sets are shown in Table 8 as TIs1 and TIs2, respectively.

We completed 30 times with these settings to get the mean values. The mean response time results of the execution queuing systems $\overline{T}_i^{(e)}$ $(1 \le i \le 10)$ are shown in Table 8.

From Table 8, we can see that the maximum relative error between the simulation and the analysis is 0.51%. By setting the traffic intensity of TIs1 and TIs2 for each host in the group of ST1, we find different mean response times for the execution queuing systems. The $\overline{T}^{(e)}$ in TIs1 has fallen from 0.571203 (second) to 0.5607632 (second) in TIs2—an increase of 1.8%. It shows that the same hardware resource will have different performance with different traffic intensity setting for each host.

We can compare the results of our analytical calculation with the results of simulation data shown in Tables 2–8 and Figures 6–11, and all of the analytical results' relative errors are less than 3.5%. The results also showed that all of the analytical results are in the range of 95% C.I. for the metrics. This shows that our analysis agrees with our simulation results, which confirms the validity of our analytical model.

5.2. Numerical Examples. We demonstrate some numerical examples to analyze the relationship among performance metrics with using the analytical model.

(i) Numerical Example 1. We use the two groups of execution servers (hosts in ST1 and ST2) shown in Table 1 with the same total computing ability to handle an arrival rate $\lambda = 155$. The execution servers in ST1 and ST2 are sorted by computing ability (computing power). In the experiments, we adjust the traffic intensity of each execution server, so that we can analyze the mean response time $\overline{T}$ and the mean waiting time $\overline{W}$ of the system. We select 16 sets of traffic intensity, and each set has 10 ρ_i $(1 \le i \le 10)$ to fit 10 execution servers.

The results of the experiments are shown in Figures 12 and 13. In the 16 experiments, we set similar traffic intensity for the ith execution server in ST1 and ST2. Although ST1 and ST2 have the same total computing ability, the mean response time $\overline{T}$ and the mean waiting time $\overline{W}$ of the system differ significantly from each other.

In Figures 12 and 13, we can see that adjusting the traffic intensity (utilization) of each execution server can enhance the response and waiting times. In ST1, the mean response time $\overline{T}$ is decreased from 0.707159 (second) to 0.554670 (second), and response time has been enhanced by 21.56%. The mean waiting time $\overline{W}$ is decreased from 0.201998 (second) to 0.049509 (second), and response time has been

Table 8: The results of each host with different traffic intensity in ST1.

HostID	Traffic intensity set 1 (TIs1)				Traffic intensity set 2 (TIs2)			
	ρ_i	Simulation	Analytical results	Relative error	ρ_i	Simulation	Analytical results	Relative error
Host#1	0.751	1.143158	1.146792	0.32%	0.651	0.87184	0.867756	0.47%
Host#2	0.755	0.76765	0.764214	0.45%	0.665	0.642525	0.640293	0.35%
Host#3	0.756	0.648089	0.647759	0.05%	0.686	0.580317	0.583304	0.51%
Host#4	0.759	0.595157	0.597003	0.31%	0.735	0.578816	0.577729	0.19%
Host#5	0.761	0.571153	0.572848	0.30%	0.749	0.560845	0.560706	0.02%
Host#6	0.764	0.554577	0.555183	0.11%	0.782	0.56216	0.562954	0.14%
Host#7	0.766	0.541873	0.543499	0.30%	0.786	0.551055	0.550656	0.07%
Host#8	0.771	0.534157	0.535349	0.22%	0.807	0.551097	0.551994	0.16%
Host#9	0.815	0.54695	0.547189	0.04%	0.811	0.542748	0.544765	0.37%
Host#10	0.798	0.52541	0.525306	0.02%	0.813	0.538181	0.538257	0.01%

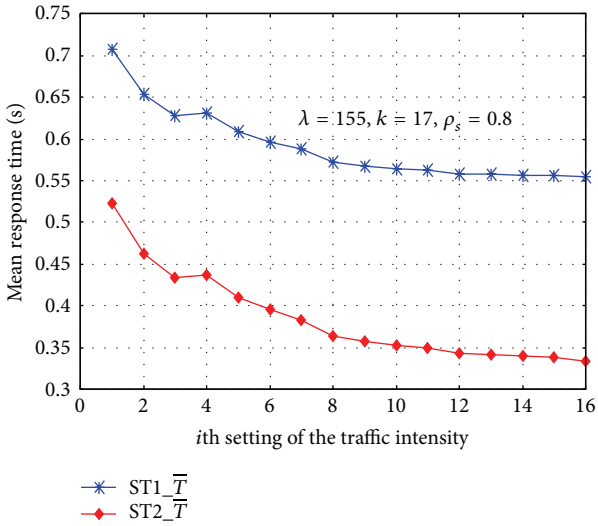

Figure 12: The mean response times of ST1 and ST2.

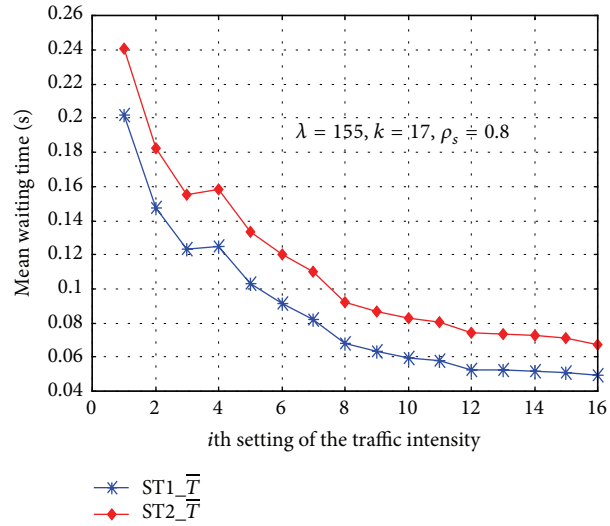

Figure 13: The mean waiting times of ST1 and ST2.

enhanced by 75.49%. In ST2, the mean response time $\overline{T}$ is decreased from 0.523104 (second) to 0.333339 (second), and response time has been enhanced by 36.27%. The mean waiting time $\overline{W}$ is decreased from 0.240814 (second) to 0.067228 (second), and response time has been enhanced by 72.08%. We can see that the traffic intensity (utilization) of each execution server has a significant impact on the performance of the heterogeneous data center.

The configuration of a server cluster in a heterogeneous data center is an important factor that greatly impacts the system's performance. Again, Figures 12 and 13 show the results of our 16 experiments, and we can see that the mean response time $\overline{T}$ of ST2 is better than ST1 under a similar traffic intensity (utilization) setting. Mean response time improves by 35.18% on average, and the maximum is 39.9%. However, the mean waiting time $\overline{W}$ of ST2 is worse than ST1. Mean waiting time is decreased by 24.95% on average and the maximum is 29%. It is important for a cloud provider to configure the server cluster into a reasonable structure to provide services for the customer. This conclusion will be confirmed in numerical example 2.

(ii) Numerical Example 2. Let us consider the third group of execution servers, named ST3, that includes 10 servers, each of which is configured as $C_i = 4$ and $f_i = 5.5$ $(1 \le i \le 10)$. The other conditions are the same as the conditions in simulation experiments 1. Assume that the arrival rate λ is set from 150 to 187.5, which means that the traffic intensity of each execution server would be in the threshold $[\delta_l, \delta_h]$.

The performance results ($\overline{T}$, $\overline{W}$, and $\overline{q}$) of the three groups (ST1, ST2, and ST3) of execution servers are shown in Figures 14–16. Different configurations of server clusters will have different performance results. The best performance of the mean response time is group ST3, and the worst is ST1, as shown in Figure 14. The best performance of mean waiting time is group ST1, and the worst is ST3, as shown in Figure 15. In Figure 16, ST1 has the maximum immediate execution probability, while the ST3 has the minimum value. We can use this queuing model to estimate the performance results of a server cluster with a certain configuration under different λ. In order to enhance the performance of mean response time, configuring the server cluster in a reasonable structure is an efficient method.

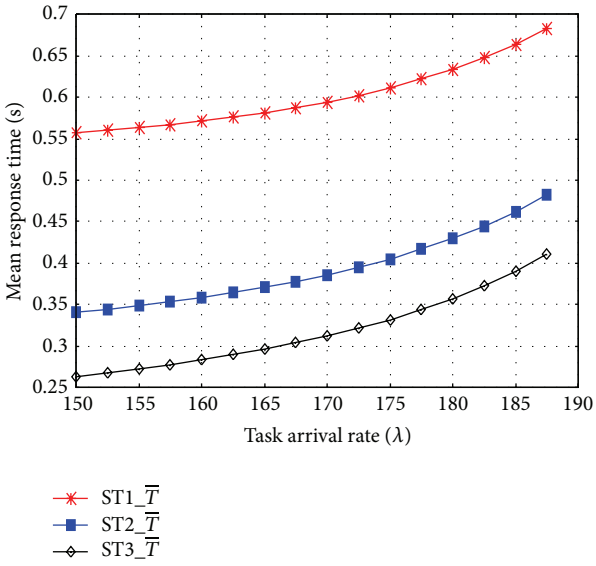

Figure 14: The mean response times of ST1, ST2, and ST3.

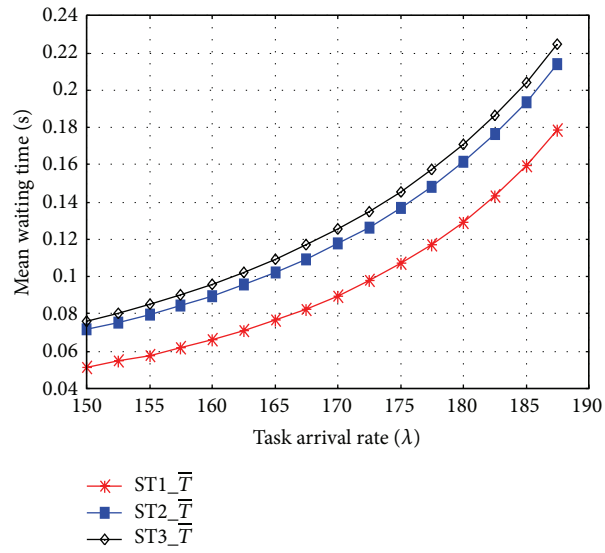

Figure 15: The mean waiting times of ST1, ST2, and ST3.

In practice, users will present some constraints when they ask cloud computing providers for services. They will present constraints as follows:

(1) The task request arrival rate $\lambda_{in} \in [\lambda_l, \lambda_h]$.

(2) The mean response time $\tau_r \leq T_l$.

(3) The mean wasting time $\tau_w \leq \overline{W}_l$ or the immediate execution probability $\zeta \geq \overline{q}_l$.

Cloud computing providers could configure the server cluster to serve customers under certain constraints by using our queuing model. In our future work, we will use the complex queuing model to further research dynamic cluster technology in a heterogeneous data center to learn how it handles different tasks with different arrival rates under various constraints.

(iii) Numerical Example 3. One application of our analytical model is to use this model for optimal system performance by finding a reasonable traffic intensity setting for each execution server. Numerical example 3 shows the optimization under the condition of $(\delta_l/\overline{I}) \sum_{i=1}^{m} C_i f_i \leq \lambda_e \leq (\delta_h/\overline{I}) \sum_{i=1}^{m} C_i f_i$ to get the values of ρ_i.

Assume using three execution servers S_1 ($C_1 = 4$, $f_1 = 4$), S_2 ($C_2 = 4$, $f_2 = 3$), and S_3 ($C_3 = 6$, $f_3 = 4$) to handle the arrival rate $\lambda_e = 38.3$ ([(16 + 12 + 24) × 0.65, (16 + 12 + 24) × 0.85]). Equations (25) and (26)—the execution server utilization optimization model based on the arrival rate λ_e—can use a numerical method to get the minimum mean response time of this system under the condition of $\lambda_e = 38.3$. As Figure 17 shows, we can see that the minimum mean response time $\overline{T}_{\min} = 0.36317$ when $\rho_1 = 0.72712$, $\rho_2 = 0.67627$, and $\rho_3 = 0.77295$. Controlling the traffic intensity (or utilization) of each execution server allows for the control of the dispatched probability of each execution server and the optimization of system performance.

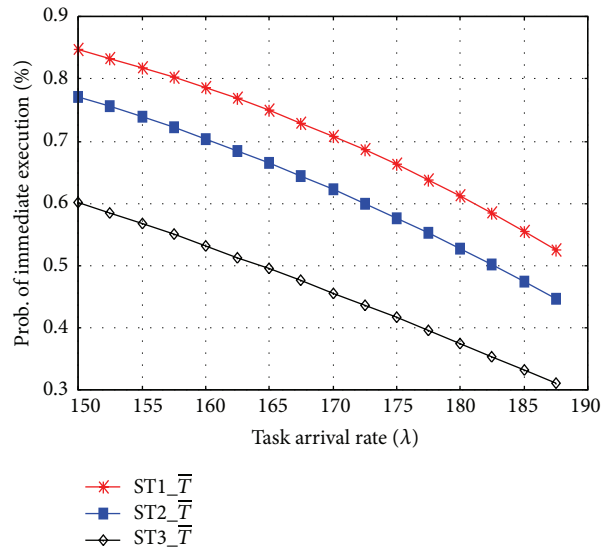

Figure 16: Immediate execution probability of ST1, ST2, and ST3.

The traffic intensity (or utilization) of each execution server will impact the response time, computing resource allocation, and energy usage of the system; therefore, we will further optimize the execution server utilization optimization model in future work.

6. Conclusions and Future Work

Performance analysis of heterogeneous data centers is a crucial aspect of cloud computing for both cloud providers and cloud service customers. Based on an analysis of the characteristics of heterogeneous data centers and their service processes, we propose a complex queuing model composed of two concatenated queuing systems—the main schedule queue and the execution queue—to evaluate the performance

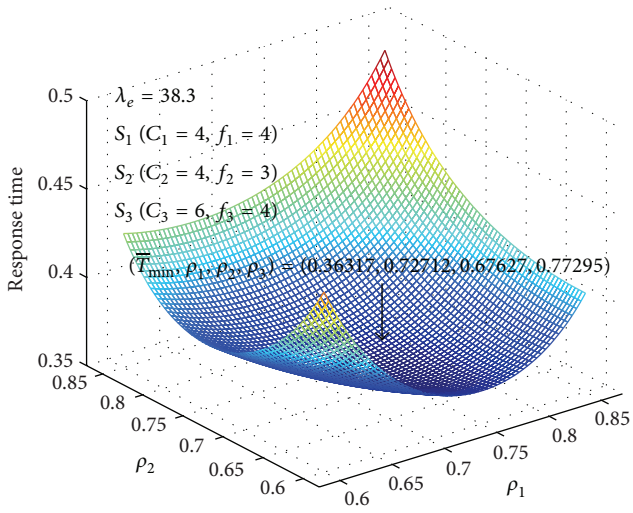

FIGURE 17: The mean response time with different traffic intensity settings.

of heterogeneous data centers. We theoretically analyzed mean response time, mean waiting time, and other important performance indicators. We also conducted simulation experiments to validate the complex queuing model. The simulation results and the calculated values demonstrate that the complex queuing model provides results with a high degree of accuracy for each performance metric and allows for a sophisticated analysis of heterogeneous data centers.

We have further conducted some numerical examples to analyze factors such as the traffic intensity (or utilization) of each execution server and the configuration of server clusters in a heterogeneous data center; these are factors that significantly impact the performance of the system. Based on this complex queuing model, we plan to extend our analytical model to the study of dynamic cluster technology in a heterogeneous data center. In doing so, we aim to help service providers optimize resource allocation using the execution server utilization optimization model.

Notations

k: The finite capacity of the scheduler queuing system
μ_s: The scheduling rate of the master server
λ_e: The master server throughput rate/the effective arrival rate of the execution queuing systems
p_i: The probability that execution server S_i has been dispatched to execute a task/the probability of a task request sent to the execution server S_i
λ_i: The arrival rate of the task request distributed to the execution server S_i
S_i: The ith node/the ith execution server
C_i: The number of cores in the ith execution server S_i
μ_i: The execution rate of the core in the ith execution server S_i
f_i: The core execution speed of the ith execution server S_i (measured by giga instructions per second (GIPS))
$\overline{I}$: The mean number of instructions in a task to be executed in the execution server (giga)
ρ_i: The utilization/traffic intensity of the ith execution server S_i
n: The number of execution servers in the data center.

Conflict of Interests

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

Acknowledgments

This project is supported by the Funds of Core Technology and Emerging Industry Strategic Project of Guangdong Province (Project nos. 2011A010801008, 2012A010701011, and 2012A010701003), the Guangdong Provincial Treasury Project (Project no. 503-503054010110), and the Technology and Emerging Industry Strategic Project of Guangzhou (Project no. 201200000034).

References

[1] B. Furht, "Cloud computing fundamentals," in *Handbook of Cloud Computing*, pp. 3–19, Springer, New York, NY, USA, 2010.

[2] W. Voorsluys, J. Broberg, and R. Buyya, "Introduction to cloud computing," in *Cloud Computing*, pp. 1–41, 2011.

[3] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[4] C. Delimitrou and C. Kozyrakis, "Paragon: QoS-aware scheduling for heterogeneous datacenters," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 1, pp. 77–88, 2013.

[5] J. Mars, L. Tang, and R. Hundt, "Heterogeneity in 'homogeneous' warehouse-scale computers: a performance opportunity," *IEEE Computer Architecture Letters*, vol. 10, no. 2, pp. 29–32, 2011.

[6] C. Vecchiola, S. Pandey, and R. Buyya, "High-performance cloud computing: a view of scientific applications," in *Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks (I-SPAN '09)*, pp. 4–16, IEEE, Kaohsiung, Taiwan, December 2009.

[7] 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, IEEE, November 2008.

[8] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*, John Wiley & Sons, 2013.

[9] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities," in *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC '08)*, pp. 5–13, IEEE, September 2008.

[10] A. Wolke and G. Meixner, "Twospot: a cloud platform for scaling out web applications dynamically," in *Towards a Service-Based Internet*, vol. 6481 of *Lecture Notes in Computer Science*, pp. 13–24, Springer, Berlin, Germany, 2010.

[11] 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.

[12] J. Vilaplana, F. Solsona, I. Teixidó, J. Mateo, F. Abella, and J. Rius, "A queuing theory model for cloud computing," *The Journal of Supercomputing*, vol. 69, no. 1, pp. 492–507, 2014.

[13] L. Guo, T. Yan, S. Zhao, and C. Jiang, "Dynamic performance optimization for cloud computing using m/m/m queueing system," *Journal of Applied Mathematics*, vol. 2014, Article ID 756592, 8 pages, 2014.

[14] X. Nan, Y. He, and L. Guan, "Optimal resource allocation for multimedia cloud based on queuing model," in *Proceedings of the 3rd IEEE International Workshop on Multimedia Signal Processing (MMSP '11)*, pp. 1–6, November 2011.

[15] X. Nan, Y. He, and L. Guan, "Queueing model based resource optimization for multimedia cloud," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 928–942, 2014.

[16] B. Yang, F. Tan, and Y.-S. Dai, "Performance evaluation of cloud service considering fault recovery," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 426–444, 2013.

[17] X. Zheng and Y. Cai, "Achieving energy proportionality in server clusters," *International Journal of Computer Networks*, vol. 1, no. 1, pp. 21–35, 2009.

[18] 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.

[19] Y.-J. Chiang and Y.-C. Ouyang, "Profit optimization in SLA-aware cloud services with a finite capacity queuing model," *Mathematical Problems in Engineering*, vol. 2014, Article ID 534510, 11 pages, 2014.

[20] J. Cao, K. Li, and I. Stojmenovic, "Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 45–58, 2014.

[21] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[22] C. L. Zhao, *Queuing Theory*, Buptprss, Beijing, China, 2nd edition, 2004.

[23] B. Hindman, A. Konwinski, M. Zaharia et al., "Mesos: a platform for fine-grained resource sharing in the data center," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI '11)*, vol. 11, pp. 295–308, 2011.

[24] V. K. Vavilapalli, A. C. Murthy, C. Douglas et al., "Apache hadoop YARN: yet another resource negotiator," in *Proceedings of the 4th Annual Symposium on Cloud Computing (SoCC '13)*, ACM, October 2013.