Performance Analysis for Heterogeneous Cloud Servers Using Queueing Theory

Shuang Wang, Xiaoping Li, *Senior Member, IEEE*, and Rubén Ruiz

Abstract—In this article, we consider the problem of selecting appropriate heterogeneous servers in cloud centers for stochastically arriving requests in order to obtain an optimal tradeoff between the expected response time and power consumption. Heterogeneous servers with uncertain setup times are far more common than homogenous ones. The heterogeneity of servers and stochastic requests pose great challenges in relation to the tradeoff between the two conflicting objectives. Using the Markov decision process, the expected response time of requests is analyzed in terms of a given number of available candidate servers. For a given system availability, a binary search method is presented to determine the number of servers selected from the candidates. An iterative improvement method is proposed to determine the best servers to select for the considered objectives. After evaluating the performance of the system parameters on the performance of algorithms using the analysis of variance, the proposed algorithm and three of its variants are compared over a large number of random and real instances. The results indicate that proposed algorithm is much more effective than the other four algorithms within acceptable CPU times.

Index Terms—Cloud computing, heterogeneous servers, power consumption, response time, Markov process

1 Introduction

Service providers and service consumers establish a relationship through a Service Level Agreement (SLA) in cloud computing environments. However, each party has a different objective. Service providers are concerned with running costs, in particular with energy consumption, while service consumers care about the expected response time. It is estimated that cloud centers consumed about 70 billion kilowatt-hours of electricity, about 1.8 percent of the total electricity consumption of the United States, in 2014 alone [1]. From 2010-2014, the electricity consumption in cloud centers increased by about 4 percent and energy use is expected to continue to increase in the near future by 4 percent between 2014-2020. According to the current trend estimates, United States cloud centers are projected to consume approximately 73 billion kWh by 2020 [1]. The expected response time is a common Quality of Service (QoS) performance metric [2]. The consumer is much more satisfied if his/her requests are quickly processed, i.e, the sooner the system processes requests from consumers the higher the levels of satisfaction. Since servers are usually heterogeneous in real cloud centers, their distinct service rates result in different levels of performance compared to the homogeneous ones where all servers have identical service rates. In addition, different additional setup times for launching servers are needed by cloud providers to offer suitable servers with various service rates when consumer service requests arrive at the cloud center. Therefore, it is desirable to minimize the tradeoff between power consumption for service providers and the expected response time so as to satisfy SLAs for consumers. More and faster servers usually imply greater power consumption and shorter expected response times. On the contrary, fewer and slower servers generally mean less power consumption and longer expected response times. Therefore, these two objectives are often conflicting. It is critical to choose the most appropriate number and type of servers in a heterogeneous cloud center to optimize the tradeoff between the expected response time and power consumption.

In this paper, we consider the problem of scheduling independent stochastic requests to heterogeneous servers considering the tradeoff between the expected response time (from the service consumer's perspective) and power consumption (from the service provider's point of view). Requests arrive at the system stochastically. Since it is difficult to analyze the system performance of complex and dynamic real cloud centers, we consider the cloud center as a queueing system [6], [7]. The heterogeneity of the servers in the cloud center means that the servers have different service rates (speeds). Similarly to [3], setup durations of different servers are assumed to be exponentially distributed.

Stochastically arriving requests and heterogeneous servers lead to a very complex problem. The number and type of servers is estimated according to both the arrival of requests and system availability. It is ideal, but at the same time difficult, to process all requests on the available servers

- *S. Wang and X. Li are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China, and also with the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing 211189, China. E-mail: {wangshuang, xpli}@seu.edu.cn.*
- *R. Ruiz is with the Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46022 València, Spain. E-mail: rruiz@eio.upv.es.*

Manuscript received 12 Mar. 2019; revised 15 Sept. 2019; accepted 22 Nov. 2019. Date of publication 28 Nov. 2019; date of current version 10 Mar. 2020. (Corresponding author: Xiaoping Li.)
Recommended for acceptance by S. Ha.
Digital Object Identifier no. 10.1109/TC.2019.2956505

at any given moment. As the number of servers is limited, stochastically arriving requests might be rejected with a given probability. More servers mean a greater system availability. A greater system availability implies a lower rejection probability which further effects the selection of the number and type of servers. The main challenges for the problem under study lie in: (i) Performance analysis being crucial to evaluate the behavior of the system with stochastically arriving requests scheduled on heterogeneous servers. Both the heterogeneity of servers and the exponentially distributed stochastic setup durations make the performance analysis model much more difficult than in the homogeneous cases which are always analyzed with the traditional $M/M/N/N+R$ queuing model [4]. (ii) Since the power consumption is closely related to the number of servers, it is not necessary to turn on all the servers in the system all of the time (traditionally it is assumed that all system servers are turned on). The number of servers to be turned on is hard to estimate when both the expected response time and power consumption are considered. (iii) For a given number of servers, there are many possible combinations for heterogeneous servers. The heterogeneity of the servers in the cloud center makes the selection of which server and which type challenging.

To obtain an optimal tradeoff between the expected response time of stochastic requests and the power consumption of heterogeneous servers in a cloud center, the performance of the system is analyzed and optimization methods are proposed. The contributions of this paper are summarized as follows:

i) Under the system availability constraint, the rejection probability is mathematically modeled for stochastically arriving requests by constructing a queueing system using a Markov decision process for a cloud center with heterogeneous servers and a finite capacity buffer where servers have stochastic setup times.
ii) With respect to the obtained rejection probability, the number of servers is estimated using a binary search and the allocation of requests to the specific servers is carried out by an iterative improvement strategy.
iii) To optimize the tradeoff between the expected response time and the power consumption, an algorithm framework with the above three components is proposed.

The remainder of the paper is organized as follows. The related work is described in Section 2. Section 3 details the model and problem formulation. Performance analysis and optimization methods are proposed in Section 4. Experimental results are given in Section 5, followed by conclusions and future research in Section 6.

2 Related Work

The performance analysis of cloud centers with capacity queues has attracted considerable research attention. However, most of the existing studies focus on cloud centers with homogeneous servers. A general analytical model was proposed for an end-to-end performance analysis of a cloud service [5]. An analytical model was presented for the performance evaluation of a cloud service on important quality metrics such as rejection probability, system overhead rate and expected request completion time [6]. The $M/G/m/m+r$ model in queueing systems was used to analyze the performance of a cloud computing center with single task arrivals in [7]. The authors considered homogeneous servers with a general distribution service rate. A pool management model was proposed for multiple could centers with single task arrivals in [8]. The model incorporates important aspects of cloud centers such as pool management, compound requests, resource virtualization and realistic services. The model was evaluated for a cloud computing center with general single arrivals and general service in [9]. Due to the variability of cloud workloads, a $G/G/c$-like model was proposed to represent a cloud-based system and expected performance metrics were computed which represent general distributions for the arrival and service patterns [9]. Servers in cloud centers are studied with the same service distribution in [5], [6], [7], [8] and [9]. In addition, they evaluated the performance metrics but did not consider server setup times.

There are only a few studies concerning heterogeneous servers with queue capacities. A stochastic model was presented for cloud centers with heterogeneous server pools in [10]. A model was constructed for a queueing system with three heterogeneous servers in [11]. The average number of jobs and their average waiting times in the system queue were obtained by the developed model for finite heterogeneous servers with different service rates in [12]. The power consumption of servers was not considered in [10], [11] or [12]. Additionally, setup times for servers were not considered in [11] or [12].

There are a few existing papers on optimal control for a queuing system with heterogeneous servers and a queue capacity. The optimal control of $M/M/c/c$ queues with periodic arrival rates and two levels for the number of servers was dealt with in [13]. The objective is to optimally assign each job of batch arriving requests to minimize the long-run average number of jobs in the entire system in [14], which is a single objective problem. Two types of servers were considered in [13] and in this paper, there are N types of different servers. The goal in [14] is to minimize the expected number of jobs in the system. In this paper the objective is to balance the expected response time and power consumption. The factors considered in this paper include stochastic setup times for servers which improve upon the results of [13] and [14] in a significant way.

The expected response time and power consumption were balanced for cloud centers with heterogeneous servers with queues in [15]. Workload dependent dynamic power management was used to improve the expected response time and power consumption for cloud centers with heterogeneous servers with queues in [27]. The servers in [15] and [27] were partitioned into different groups based on their processing speeds, i.e., servers with the same speed are put into the same group. Therefore, the scenarios in [15] and [27] are different to the one considered in this paper which does not group the heterogeneous servers. A new hierarchical correlation model was constructed to analyze and evaluate reliability, performance and power consumption in [16]. For multi-agent cloud systems, a reliability-performance-energy

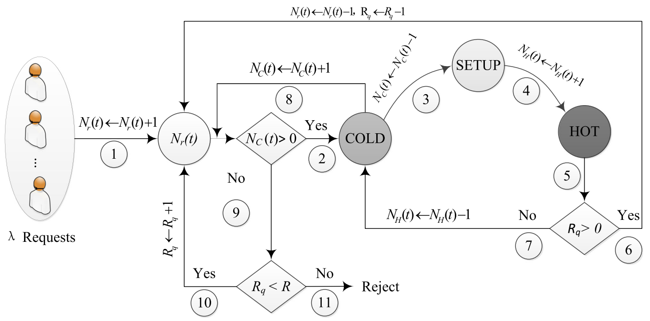
$N_r(t) \leftarrow N_r(t)-1,\ R_q \leftarrow R_q-1$
$N_C(t) \leftarrow N_C(t)+1$
$N_C(t) \leftarrow N_C(t)-1$
SETUP
$N_H(t) \leftarrow N_H(t)+1$
$N_r(t) \leftarrow N_r(t)+1$
$N_r(t)$
$N_C(t) > 0$
Yes
COLD
HOT
No
$R_q \leftarrow R_q+1$
$N_H(t) \leftarrow N_H(t)-1$
No
$R_q > 0$
Yes
λ Requests
Yes
$R_q < R$
No
Reject
1
2
3
4
5
6
7
8
9
10
11

Fig. 1. State transitions for the system.

correlation model was proposed to develop optimal request scheduling and resource management strategies in [17]. Setup times for servers were not taken into account in [15], [16], [27] or in [17].

To the best of our knowledge, the problem with setup times and optimal tradeoff between the expected response time and power consumption has never been considered for cloud centers with heterogeneous servers and a queue capacity in the literature. The studied model in this paper brings the research results closer to real life scenarios. However, the joint consideration of all these features results in a remarkably complex problem.

3 Problem Description and Model

For the considered scenario, requests arrive at the system stochastically. They are either allocated to servers or rejected. For each of the N servers in the cloud center, there are three possible states: COLD, SETUP and HOT. $N_C(t)$ and $N_H(t)$ are the number of servers in the COLD and HOT states at time t respectively. Since SETUP is transient, i.e., a server in SETUP would be HOT in a very short time, $N = N_C(t) + N_H(t)$. There is a queue with a buffer capacity R and the current number of requests in the queue R_q $(R_q = 0, \ldots, R)$. $N_r(t) = N_H(t) + R_q$ is the number of requests in the system. Transitions among the states are depicted in Fig. 1.

All requests are processed by the first-come, first-served (FCFS) rule. The server state transition process is described as follows:

i) An incoming request Req is allocated to a server S_k $(k = 1, \ldots, N_C(t))$ if there is at least one server (i.e., $N_C(t) > 0$) in the COLD state. $N_r(t) \leftarrow N_r(t) + 1$.
ii) S_k is set up before it can process request Req. $N_C(t) \leftarrow N_C(t) - 1$.
iii) S_k in the SETUP state means it is becoming ready. After set up, S_k transitions to the HOT state which means it is available to process requests. $N_H(t) \leftarrow N_H(t) + 1$.
iv) When Req finishes on S_k, the system checks whether there are requests waiting in the queue. $R_q > 0$ implies that there are R_q requests waiting for servers. S_k is allocated to the first request in the queue directly without changing its state (i.e., staying in the HOT state). $N_r(t) \leftarrow N_r(t) - 1$. $R_q \leftarrow R_q - 1$.
v) If there are no requests in the queue, i.e., $R_q = 0$, S_k transitions to the COLD state and $N_H(t) \leftarrow N_H(t) - 1$ as well as $N_C(t) \leftarrow N_C(t) + 1$.
vi) If none of the servers are in the COLD state (i.e., $N_C(t) = 0$), the system verifies if the queue is full. If not (i.e., $R_q < R$), Req is appended to the queue, i.e., $R_q \leftarrow R_q + 1$. Go to (iv).
vii) If the queue is full (i.e., $R_q = R$), Req is rejected.

3.1 Assumptions and Notation

For the considered system, we make the following assumptions:

- All requests arrive at the system with a Poisson distribution with the arrival rate λ as in [6] and [7]. Requests are served one by one.
- The processing time of requests is independent and exponentially distributed random variables as in [18] and [19]. For simplicity, all servers are sorted into non-increasing order of their service rates μ_k $(k = 1, \ldots, N)$, i.e., $\mathbb{S} = (S_1, S_2, \ldots, S_N)$ with $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_N$.
- The delay time for servers for setups is an independent and exponentially distributed random variable with rate θ as in [3].
- The system availability which is defined as the probability of an adequate level of service [20] is ξ. A server can be turned off (from HOT to COLD) immediately if there are no requests in the queue. Servers are turned on or off one by one, i.e., only one transition at time t.

Based on the state transition of each server, the state transition of the whole system is a stochastic process $Z(t) = (X(t), M(t))$ where $X(t)$ is the observed process and $M(t)$ denotes the control process [21]. $X(t)$ represents the system state at time t with $X(t) = (N_r(t), N_H(t))$. $N_r(t) (0 \leq N_r(t) \leq N + R)$ is the number of requests in the system at time t. $N_H(t) (0 \leq N_H(t) \leq N)$ is the number of servers processing the requests, i.e., there are $N_H(t)$ servers in the HOT state at time t (all servers are in the COLD state if $N_H(t) = 0$). The state space set of $X(t)$ $(t \in [0, +\infty))$ is denoted as Ω. For any time t, $J_C(t) = \{j | S_j \text{ in COLD}\}$ and $J_H(t) = \{j | S_j \text{ in HOT}\}$ are the sets of $N_C(t)$ COLD servers and $N_H(t)$ HOT servers respectively. The total service rates for COLD and HOT servers are $U_I(t) = \sum_{j \in J_C(t)} \mu_j$ and $U_H(t) = \sum_{j \in J_H(t)} \mu_j$ respectively. $M(t)$ controls the current state $X(t)$, i.e., a decision is made to transit from $X(t)$ to $X(t + \vartheta_t)$ according to the number of requests and the power consumption in the system where ϑ_t is the time period from $X(t)$ to the next state. As shown in Fig. 1, there are only three fundamental actions at time t.

i) $A^0(t)$: No server is turned on or off.
ii) $A_j^+(t)$: The j^{th} $(j \in J_C(t))$ server is turned on which means $N_H(t) \leftarrow N_H(t) + 1$ and $N_C(t) \leftarrow N_C(t) - 1$.
iii) $A_j^-(t)$: The j^{th} $(j \in J_H(t))$ server is turned off which means $N_H(t) \leftarrow N_H(t) - 1$ and $N_C(t) \leftarrow N_C(t) + 1$.

At any time t of the whole process, the state $X(t + \vartheta_t)$ is determined by $M(t)$ which conducts one and only one action $a(t) = \{A^0(t), A_j^+(t), A_j^-(t)\}$ according to the balance between the expected response time and power consumption.

Obviously, $Z(t) = (X(t), M(t))$ is a Markov Decision Process (MDP). Two fundamental bases $e_0 = (1, 0)$ and $e_1 = (0, 1)$ are introduced. For the current state $X(t)$, there are five possible states $X(t + \vartheta_t)$:

i) $(N_r(t)+1, N_H(t))$: A new request arrives while the number of HOT servers remains constant, i.e., $(N_r(t), N_H(t)) + e_0 \overset{A^0(t)}{\to} (N_r(t)+1, N_H(t))$.

ii) $(N_r(t), N_H(t)+1)$: A COLD server is turned on, i.e., $(N_r(t), N_H(t)) + e_1 \overset{A_j^+(t)}{\to} (N_r(t), N_H(t)+1)$.

iii) $(N_r(t)-1, N_H(t))$: A request finishes while the number of HOT servers remains constant, i.e., $(N_r(t), N_H(t)) - e_0 \overset{A^0(t)}{\to} (N_r(t)-1, N_H(t))$.

iv) $(N_r(t)-1, N_H(t)-1)$: A request finishes and a HOT server is powered down when $N_r(t+\vartheta_t) < N_H(t+\vartheta_t)$, i.e., $(N_r(t), N_H(t)) - e_0 - e_1 \overset{A_j^-(t)}{\to} (N_r(t)-1, N_H(t)-1)$.

v) $(N_r(t), N_H(t))$: The state does not change during ϑ_t. $(N_r(t), N_H(t)) \overset{A^0(t)}{\to} (N_r(t), N_H(t))$.

For the three states (COLD, SETUP and HOT) of server S_k, a sign function $\psi(k)$ is defined as below.

$$\psi(k) = \begin{cases} -1 & \text{if } S_k \text{ is in the SETUP state} \\ 0 & \text{if } S_k \text{ is in the COLD state} \\ 1 & \text{if } S_k \text{ is in the HOT state} \end{cases}.$$

In terms of K. Li [27], the power consumption of S_k in HOT state is determined by $P_k^H = wCV_k^2\eta_k$, where w is the switching activity, C the electrical capacitance, V_k the supply voltage and η_k the clock frequency. For any physical server in the HOT state with a rate μ_k, $\mu_k \propto \eta_k$ and $\eta_k \propto V_k^\phi$ with $0 < \phi \le 1$. $\eta_k \propto V_k^\phi$ implies $V_k \propto \eta_k^{1/\phi}$. According to [23], $\mu_k \propto \eta_k$ and $V_k \propto \eta_k$ imply $P_k \propto \mu_k^\alpha$ where $\alpha = 1 + 2/\phi \ge 3$, i.e., P_k^H can be represented by $\kappa\mu_k^\alpha$ where κ is a constant and S_k consumes the minimum amount of power if $\alpha = 3$. The power consumption is $\kappa\mu_k^\alpha$ when S_k is in the HOT state while the power consumption is constant (P_k^C) when S_k is in the COLD state. According to [30], the power consumption of S_k is $P_k^S = \frac{(\sum_{i=1}^N \mu_i - \lambda)\theta + \lambda(\sum_{i=1}^N \mu_i + \lambda)\kappa\mu_k^\alpha}{\sum_{i=1}^N \mu_i(\lambda+\theta)}$ when S_k in the SETUP state. Therefore, the power consumption for server S_k is calculated by

$$\begin{aligned} P_k = P_k^C &+ \frac{(\psi(k)+1)\psi(k)}{2}\kappa\mu_k^\alpha \\ &+ \frac{(\psi(k)-1)\psi(k)}{2}\,\frac{(\sum_{i=1}^N \mu_i - \lambda)\theta + \lambda(\sum_{i=1}^N \mu_i + \lambda)\kappa\mu_k^\alpha}{\sum_{i=1}^N \mu_i(\lambda+\theta)}. \end{aligned} \tag{1}$$

3.2 Detailed Model

For incoming requests with a given rate λ, more servers imply higher power consumption and reduced response times for the requests. The expected response time is positively correlated with the number of requests in the system with a given arrival rate according to the Little's Theorem [4]. The number of requests is used to measure the expected response time. The tradeoff between the expected response time and power consumption implies the allocation of a number and type of servers to the stochastically arriving requests.

Due to the stochastic and heterogeneous properties in the system, the studied problem is divided into three subproblems: (i) Calculating the rejection probabilities for server configurations in all states. (ii) Calculating the minimum required number of servers $n \le N$ for the $N_r(t)$ requests, and (iii) selecting the appropriate type for the servers for the $N_r(t)$ requests.

For the number of servers n, the state space $\Omega_n \subset \Omega_N$ is certain. In fact, the number of servers in HOT $N_H(t)$ is not bigger than the number of requests $N_r(t)$ if $N_r(t)$ is less than n whereas $N_H(t)$ is not bigger than n if $N_r(t)$ is larger than n, i.e., $\Omega_n = \{(N_r(t), N_H(t)) : N_r(t) = 0, \dots, n;\ N_H(t) = 0, \dots, N_r(t)\} \bigcup \{(N_r(t), N_H(t)) : N_r(t) = n+1, \dots, n+R; N_H(t) = 0, \dots, n\}$. Obviously, $N_H(t) \le n$. Only the SETUP server consumes power when $N_H(t) = 0$ and all HOT servers consume power when $N_H(t) = n$. When $n > N_H(t) > 0$, power consumption is determined by the HOT servers and the SETUP server. Among all the $|\Omega_n|$ states, the minimum power consumption is $P_{[1]}$ and the maximum is $\sum_{j=1}^n (P_{[j]}^C + \kappa\mu_{[j]}^\alpha)$. Since the number of requests in the system and the power consumption have different ranges and units, we employ a min-max normalization. $W(N_r(t)) = \frac{N_r(t)}{n+R}$ for the $N_r(t)$ requests in the current state and $W(P_{[j]}) = \frac{P_{[j]} - P_{[1]}}{\sum_{j=1}^n (P_{[j]}^C + \kappa\mu_{[j]}^\alpha)}$ for the power consumption of the jth server. $P_{[N_H(t)+1]}$ indicates the power consumption of the SETUP server. If $N_H(t) = n$, no more servers will be set up, i.e., $W(P_{[n+1]}) = 0$.

To optimize the tradeoff between the number of requests and power consumption, the objective is to minimize the function defined below.

$$Y(t) = \int_0^t y(x(\tau))d\tau, \tag{2}$$

where

$$y(x(t)) = \beta W(N_r(t)) + (1-\beta) \sum_{j=0}^{N_H(t)} W(P_{[j+1]}). \tag{3}$$

β is the weight of the objective and $y(x(t))$ is the weighting function of the two objectives at time t.

4 Proposed Methods

With the given request arrival rate λ and server configurations for all states, the rejection probability $P_R(\vec{f})$ is calculated using the Markov process analysis method. According to $P_R(\vec{f})$ and the system availability ξ, the minimum number of servers $n \le N$ is determined by Binary-Search. The Markov decision process is adopted to select n appropriate servers to minimize the number of requests and power consumption simultaneously. Since there are $\frac{N!}{(N-n)!}$ possible combinations when selecting n servers from N heterogeneous servers, it is not hard to show that the server selection is NP-hard which further implies that the considered problem is NP-hard.

To balance the expected response time and power consumption, the BETP (Balanced Expected response Time and Power consumption) algorithm is proposed for the considered problem. The framework of BETP, as depicted in Algorithm 1, contains three components: (i) Evaluating the rejection probability according to server configurations. (ii) Determining the minimum number of hot servers. (iii) Selecting the appropriate servers.

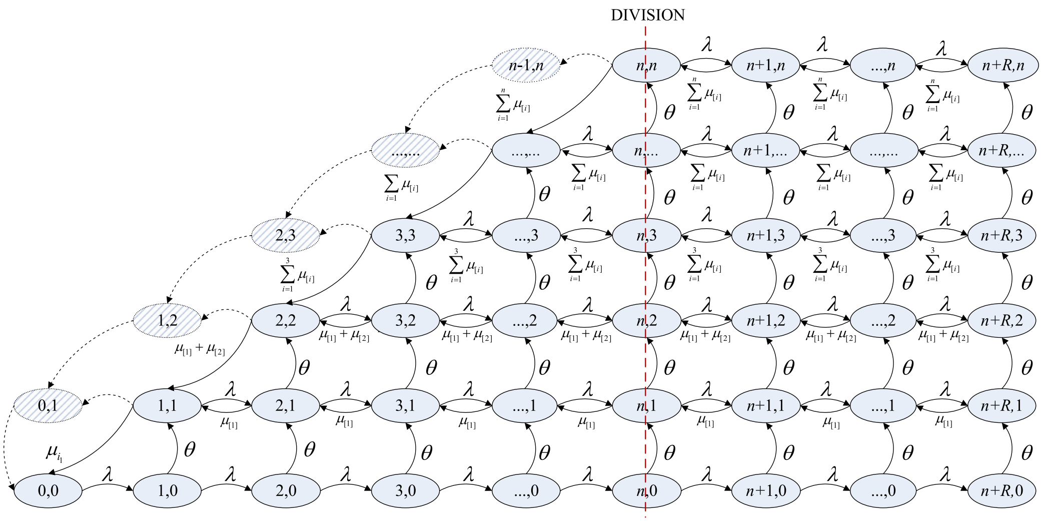

Fig. 2. The state transition process in the proposed queueing system.

```
Algorithm 1. Balanced Expected Response Time and Power Consumption (BETP)
1 begin
2    Calculate rejection probabilities according to server configurations;
3    Call the Determining Server Number procedure;
4    Call the Server Selection procedure;
5 return.
```

4.1 Calculation of the Rejection Probability

The queueing system forms a Markov decision process which is a two-dimensional state space. If the policy of the dispatched servers is certain, the state space is reduced to a Markov process. The state space $\Omega_n \subset \Omega_N$ with states $X(t) = (N_r(t), N_H(t))$, is reduced to a Markov process. According to the transition rules, the state transitions are depicted in Fig. 2 (where light gray states are immediate states in the dynamic procedure of immediate transitions). A policy $\vec{f} = (\vec{h}^{(0)}, \vec{h}^{(1)}, \ldots, \vec{h}^{(n)})$ is an M-dimensional server rate vector. $\vec{h}^{(i)}$ is the vector corresponding to the ith ($i = 0, \ldots, n$) row in Fig. 2. $\vec{h}^{(i)}$ contains $n + R + 1 - i$ identical elements $\mu_{[i]}$. All $n + R + 1$ elements of $\vec{h}^{(0)}$ are 0. The Matrix-Geometric method [20], [30] and the delay time of servers setting up in [24] is considered simultaneously to analyze the Markov process.

Fig. 2 demonstrates that there are $M = \frac{(n+2+2R)(n+1)}{2}$ states in total when the process becomes steady. To calculate the rejection probability $P_R(\vec{f})$, the infinitesimal generator matrix $\mathbf{Q}^{\vec{f}}$ is first obtained as follows [20]:

$$\mathbf{Q}^{\vec{f}} = \begin{pmatrix} \mathbf{A}_0 & \mathbf{C}_0 & & & & & \\ \mathbf{B}_1 & \mathbf{A}_1 & \mathbf{C}_1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \mathbf{B}_n & \mathbf{A}_n & \mathbf{C}_n & & \\ & & & \mathbf{B}_{n+1} & \mathbf{A}_{n+1} & \mathbf{C}_{n+1} & \\ & & & & \ddots & \ddots & \ddots \\ & & & & & \mathbf{B}_{n+R} & \mathbf{A}_{n+R} \end{pmatrix}_{M\times M}. \quad (4)$$

All M states are sorted by the order of those in Fig. 2 from bottom to top and from left to right. The kth ($0 \le k \le n + R$) row of $\mathbf{Q}^{\vec{f}}$ corresponds to the transition rates of the states in the kth column to the state space $\boldsymbol{\Omega}_n$ with the state order $(0, 0), (1, 0), (1, 1), (2, 0), \ldots, (n, n + R)$ in Fig. 2. For example, $\mathbf{A}_0$ and $\mathbf{C}_0$ correspond to the state (0,0); $\mathbf{A}_n$, $\mathbf{B}_n$ and $\mathbf{C}_n$ correspond to the state order $(n, 0)$, $(n, 1), \ldots, (n, n)$ (the column with the DIVISION line). Each diagonal element of the matrix A_k represents the output rates for the corresponding states. The delay rate θ of the matrix A_k means the delay when setting-up servers. If $k = 0$, the matrix A_0 is described as $\mathbf{A}_0 = (-\lambda)$. If $0 < k \le n$, the output rate of each state on the main diagonal of $\mathbf{A}_k$ is closely related to $\eta = -(\lambda + \theta)$. The $(k + 1) \times (k + 1)$ matrix $\mathbf{A}_k$ is

$$\mathbf{A}_k = \begin{pmatrix} \eta & \theta & & & \\ & \eta - \mu_{[1]} & \theta & & \\ & & \ddots & \ddots & \\ & & & \eta - \sum_{i=1}^{k-1} \mu_{[i]} & \theta \\ & & & & -\lambda - \sum_{i=1}^{k} \mu_{[i]} \end{pmatrix}.$$

If $n < k \le n + R$, the output rate of each state on the main diagonal of $\mathbf{A}_k$ is closely related to $\eta = -(\lambda + \theta)$. The $(n + 1) \times (n + 1)$ matrix $\mathbf{A}_k = \mathbf{A}_n$. If $k = n + R$, the $(n + 1) \times (n + 1)$ matrix $\mathbf{A}_k$ is

$$\mathbf{A}_k = \begin{pmatrix} -\theta & \theta & & \\ & -\theta - \mu_{[1]} & \theta & \\ & & \ddots & \ddots \\ & & & -\sum_{i=1}^{n} \mu_{[i]} \end{pmatrix}.$$

Every matrix $\mathbf{B}_k$ ($1 \le k \le n + R$) describes the service rate of the system for the corresponding states described in Fig. 2. If $1 \le k \le n$, the matrix $\mathbf{B}_k$ is derived as

$$\mathbf{B}_k = \begin{pmatrix} 0 & & & & \\ \mu_{[1]} & & & & \\ & \ddots & & & \\ & & \sum_{i=1}^{k}\mu_{[i]} & & \\ & & & \sum_{i=1}^{k}\mu_{[i]} & \end{pmatrix}_{(k+1)\times(k+1)}.$$

If $n < k \leq n+R$, the matrix $\mathbf{B}_k$ is derived as

$$\mathbf{B}_k = \begin{pmatrix} 0 & & & & \\ \mu_{[1]} & & & & \\ & \ddots & & & \\ & & \sum_{i=1}^{n-1}\mu_{[i]} & & \\ & & & \sum_{i=1}^{n}\mu_{[i]} & \end{pmatrix}_{(n+1)\times(n+1)}.$$

Each matrix $\mathbf{C}_k$ $(0 \leq k \leq n+R)$ illustrates the arrival rates for the corresponding states of the system. For $0 \leq k \leq n$, the matrix $\mathbf{C}_k$ is a $k+1$ dimensional matrix obtained as

$$\mathbf{C}_k = \begin{pmatrix} \lambda & & & & \\ & \lambda & & & \\ & & \ddots & & \\ & & & \lambda & \\ & & & & \lambda \end{pmatrix}_{(k+1)\times(k+1)}.$$

For $n \leq k \leq n+R$, the matrix $\mathbf{C}_k$ is a $n+1$ dimensional matrix obtained as $\mathbf{C}_n$.

After the infinitesimal generator matrix $\mathbf{Q}^{\vec{f}}$ is obtained, we obtain the steady state probabilities vector $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_M)$ for the state space Ω_n which corresponds to the states $\{(0,0),(1,0),(1,1),\ldots,(n+R,n)\}$ as shown in Fig. 2. According to $\mathbf{Q}^{\vec{f}}$ and $\boldsymbol{\pi}$, the balance vector equation is

$$\boldsymbol{\pi}\mathbf{Q}^{\vec{f}} = \mathbf{0}, \tag{5}$$

in which the steady state probability vector $\boldsymbol{\pi}$ satisfies the following equation:

$$\sum_{i=1}^{M}\pi_i = 1. \tag{6}$$

In terms of Equations (5) and (6), the steady state probabilities $\pi_1, \ldots, \pi_M$ can be obtained exactly. The incoming request would be rejected when the number of requests in the system equals $n+R$. According to the steady state probability vector $\boldsymbol{\pi}$, the rejection probability $P_R(\vec{f})$ is calculated by the states of the last column shown in Fig. 2, i.e.,

$$P_R(\vec{f}) = \sum_{i=M-n}^{M}\pi_i. \tag{7}$$

To illustrate the above procedure, we give an example with $n=1, R=1, \mu_{[1]}=2, \theta=3$ and $\lambda=1$. The obtained $\mathbf{Q}^{\vec{f}}$ is

$$\mathbf{Q}^{\vec{f}} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -4 & 3 & 1 & 0 \\ 2 & 0 & -3 & 0 & 1 \\ 0 & 0 & 0 & -3 & 3 \\ 0 & 0 & 2 & 0 & -2 \end{pmatrix}. \tag{8}$$

The determinant of $\mathbf{Q}^{\vec{f}}$ is 0. In terms of Equations (5) and (6), the determinant becomes non-zero by replacing any column of $\mathbf{Q}^{\vec{f}}$ with the vector $(1,1,1,1,1)^T$. The 5 unknown quantities are obtained with $\boldsymbol{\pi} = (0.45, 0.11, 0.23, 0.04, 0.17)$ using the elimination method.

4.2 Determining the Number of Servers

According to Equation (7), the rejection probability $P_R(\vec{f})$ is closely related to the number of selected servers n. In addition, $P_R(\vec{f})$ is not more than $1-\xi$, i.e., $P_R(\vec{f})$ is constrained by the system availability ξ. With an increase in n, the rejection probability $P_R(\vec{f})$ decreases whereas power consumption increases. On the contrary, power consumption is decreased by decreasing n while an increase in $P_R(\vec{f})$ would result in $P_R(\vec{f}) > 1-\xi$ which does not meet the system availability requirements. Therefore, it is desirable to find an appropriate value for n. In this paper, the appropriate number n is found by the Binary Search method as depicted in Algorithm 2.

The lower and upper bounds n_{min} and n_{max} on the number of selected servers are initialized as 1 and N respectively. n is initialized to the median $\frac{\lfloor n_{min}+n_{max}\rfloor}{2}$. To guarantee at least one choice of the n servers is feasible in the following server selection process, we consider those servers with the maximum service rates first. The first n servers $\vec{S}=(S_1, S_2, \ldots, S_n)$ with the highest service rates $\vec{\boldsymbol{\mu}}=(\mu_1, \mu_2, \ldots, \mu_n)$ are initially selected. At any time, the system state is one of the M states as depicted in Fig. 2. A policy $\vec{f} = (\vec{h}^{(0)}, \vec{h}^{(1)}, \ldots, \vec{h}^{(n)})$ is an M-dimensional server rate vector. All $n+R+1$ elements of $\vec{h}^{(0)}$ are 0. During the process to determine the number of servers, all the $n+R+1-i$ elements of $\vec{h}^{(i)}$ are μ_i (some of them might change in the following server selection strategy). In terms of Equations (5), (6) and (7), $P_R(\vec{f})$ is closely related to $\vec{f}$ which depends on n. $P_R(\vec{f}) < 1-\xi$ implies that the appropriate number lies within $[n_{min}, \frac{\lfloor n_{min}+n_{max}\rfloor}{2}]$, i.e., $\frac{\lfloor n_{min}+n_{max}\rfloor}{2}$ should be the upper bound n_{max}. On the contrary, the appropriate number is between $\frac{\lfloor n_{min}+n_{max}\rfloor}{2}$ and n_{max}, i.e., the lower bound n_{min} is set to $\frac{\lfloor n_{min}+n_{max}\rfloor}{2}$. The updating strategy leads to a new $\vec{f}$. In terms of Equations (5), (6) and (7), we obtain a different $P_R(\vec{f})$ which is compared to $1-\xi$ again. The procedure terminates when the condition $n_{min} \leq n_{max}$ is met.

To illustrate the above procedure, the following example is given: $N=6, \lambda=5, R=2, \theta=1, \beta=0.5, \mu_1=6, \mu_2=4, \mu_3=3, \mu_4=0.3, \mu_5=0.2, \mu_6=0.1, \xi=0.5$. The values of the variables and vectors in Algorithm 2 are shown in Table 1. The minimum number of servers is 2, i.e., $n=2$. The computational time complexity of the algorithm to determine the number of servers is $O(\log(N))$.

4.3 Server Selection Strategy

For the λ requests stochastically arriving at the system, the required number of servers is not more than n where n is determined by Algorithm 2 when the system is in a steady

TABLE 1
Values of the Binary Search Procedure Example

n_{min}	n_{max}	n	$\vec{S}$	$\vec{\mu}$	$\vec{f}$	$P_R(\vec{f})$
1	6	3	(S_1, S_2, S_3)	(6,4,3)	(0,0,0,0,0,0,6,6,6,6,6,4,4,4,4,3,3,3)	0.401
2	3	2	(S_1, S_2)	(6,4)	(0,0,0,0,0,6,6,6,6,4,4,4)	0.492
1	2	1	(S_1)	6	(0,0,0,0,6,6,6)	0.619
2	2	2	(S_1, S_2)	(6,4)	(0,0,0,0,0,6,6,6,6,4,4,4)	0.492

state. Using the Markov decision process, servers with the minimum long-run expected reward $E(Y(t))$ are selected. $E(Y(t))$ depends on the initial system state $X(0)$ and the policy $\vec{f}$, i.e., $E(Y(t)) = E^{\vec{f}}_{X(0)}Y(t)$. $E(Y(t))$ is the sum of the expected rewards of all the included states Ω_n, when the system is in the steady state. A policy $\vec{f}$ determines the controlling process $M(t)$ of $Z(t)$ for Ω_n. All the states form a Markov chain as shown in Fig. 2. In terms of Rykov and Efrosinin [21], the probability distribution $P^{\vec{f}}_x$ of the state $x \in \Omega_n$ in $Z(t)$ can be obtained for each given policy $\vec{f}$ when the system is in a steady state. The corresponding long-run expected reward of each time unit $g^{\vec{f}}$ is the expectation of $Z(t)$ with probability distribution $P^{\vec{f}}_x$, i.e., $g^{\vec{f}} = \lim_{t\to\infty} \frac{1}{t}E^{\vec{f}}_{X(0)}Y(t)$. $g^{\vec{f}}$ is closely related to $\vec{f}$ but not to $X(0)$. Therefore, the expected reward function $Y(t)$ at any time t is determined by $E(Y(t)) = E^{\vec{f}}_{X(0)}Y(t) = tg^{\vec{f}} + v^{\vec{f}}_{X(0)}$. $v^{\vec{f}}_{X(0)}$ is a deviation function of $\vec{f}$ and $X(0)$ when the system is in a steady state. In this way, $\min E(Y(t))$ is equivalent to $\min_{\vec{f}}\{tg^{\vec{f}} + v^{\vec{f}}_{X(0)}\}$.

Algorithm 2. Determining Server Number

```
   Input: $\lambda, N, \theta, \mu_1, \mu_2, \dots, \mu_N, \xi, \beta$
 1 begin
 2   $n_{min} \leftarrow 1, n_{max} \leftarrow N$;
 3   while $n_{min} \leq n_{max}$ do
 4     $n \leftarrow \frac{\lfloor n_{min}+n_{max} \rfloor}{2}$;
 5     Construct $\vec{\mu}, \vec{f}$;
 6     Calculate $\mathbf{Q}$, $\boldsymbol{\pi}$ and $P_R(\vec{f})$ using Equations (4), (5), (6)
       and (7) respectively;
 7     if $P_R(\vec{f}) \leq 1 - \xi$ then
 8       $n_{max} \leftarrow n$;
 9     else
10       $n_{min} \leftarrow n$ +1;
11 return $n, \vec{f}$;
```

4.3.1 Server Selection Framework

Both $g^{\vec{f}}$ and $v^{\vec{f}}_{X(0)}$ are closely related to $\vec{f}$. We develop the following optimal server selection principle to minimize the objective $\min_{\vec{f}}\{tg^{\vec{f}} + v^{\vec{f}}_{X(0)}\}$ according to the optimal principle for a one-chain Markov decision problem [25]:

i) The gain $g^* = \min_{\vec{f}} g^{\vec{f}}$ exists.

ii) Let $\vec{f}$ be a given reasonable policy, i.e., $\lambda \leq \sum_{i=1}^{n} \mu_{[i]}$. The system state is $X(\zeta)$ after a period of time ζ from the initial state $X(0)$. $g^{\vec{f}}$ and $v^{\vec{f}}_{X(0)}$ are determined by $v^{\vec{f}}_{X(0)} = r^a_{X(0)} - g^{\vec{f}}\zeta + \sum_{j\in\Omega_n} p^a_{X(0),X(\zeta)} v^{\vec{f}}_{X(\zeta)}$ according to Theorem 1. $p^a_{X(0),X(\zeta)}$ is the transition probability from $X(0)$ to $X(\zeta)$ by the action a of $\vec{f}$ (which can be obtained by (13)). $p^a_{X(0),X(\zeta)}$ is obtained from Equation (4) [4].

iii) Since both $g^{\vec{f}}$ and $v^{\vec{f}}_{X(0)}$ are closely related to policy $\vec{f}$, a newly constructed policy $\vec{f}'$ updates $\vec{f}$ if $v^{\vec{f}'}_{(X(0))} \leq v^{\vec{f}}_{X(0)}$ by Theorem 2.

Theorem 1. *For a given reasonable policy $\vec{f}$ and the current state space Ω_n, $X(\zeta)$ is the reachable state from state $X(0)$ with probability $p^a_{X(0),X(\zeta)}$ by action a on server S_a corresponding to the service rate in $\vec{f}$. ζ is the average time on all possible reachable states from $X(0)$ in the next decision epoch and $r^a_{X(0)}$ is the actual reward during this period. The deviation value of $v^{\vec{f}}_{X(0)}$ satisfies the following equation.*

$$v^{\vec{f}}_{X(0)} = r^a_{X(0)} - g^{\vec{f}}\zeta + \sum_{X(\zeta)\in\Omega_n} p^a_{X(0),X(\zeta)} v^{\vec{f}}_{X(\zeta)}. \tag{9}$$

Proof. For any two different states $s_1, s_2 \in \Omega_n$, s_1 is reachable from s_2, i.e., any state $s \in \Omega_n$ is recurrent. Let s be a recurrent state under the policy $\vec{f}$. $T^{\vec{f}}_{X(0),s}$ and $K^{\vec{f}}_{X(0),s}$ denote the expected time and the expected reward respectively, of the first reach from the initial state $X(0)$ to the state s controlled by policy $\vec{f}$, i.e., $T^{\vec{f}}_{X(0),s} = \zeta + \sum_{X(\zeta)\in\Omega_n, X(\zeta)\neq s} p^a_{X(0),X(\zeta)} T^{\vec{f}}_{X(\zeta),s}$ and $K^{\vec{f}}_{X(0),s} = r^a_{X(0)} + \sum_{X(\zeta)\in\Omega_n, X(\zeta)\neq s} p^a_{X(0),X(\zeta)} K^{\vec{f}}_{X(\zeta),s}$. In addition, $g^{\vec{f}} = \frac{K^{\vec{f}}_{s,s}}{T^{\vec{f}}_{s,s}}$ according to the above assumptions and the renewal-reward theorem in [4]. In terms of the above definition of deviation, $v^{\vec{f}}_{X(0)} = K^{\vec{f}}_{X(0),s} - g^{\vec{f}}T^{\vec{f}}_{X(0),s} = r^a_{X(0)} - g^{\vec{f}}\zeta + \sum_{X(\zeta)\in\Omega_n, X(\zeta)\neq s} p^a_{X(0),X(\zeta)}\{K^{\vec{f}}_{X(\zeta),s} - g^{\vec{f}}T^{\vec{f}}_{X(\zeta),s}\}$. The fact that s is recurrent (i.e., from state s back to s) implies that $K^{\vec{f}}_{s,s} - g^{\vec{f}}T^{\vec{f}}_{s,s} = 0$. Therefore, $v^{\vec{f}}_{X(0)} = y^a_{X(0)} - g^{\vec{f}}\zeta + \sum_{X(\zeta)\in\Omega_n} p^a_{X(0),X(\zeta)} v^{\vec{f}}_{X(\zeta)}$. □

Theorem 2. *A new policy $\vec{f}'$ is constructed by replacing the action of state $X(0)$ in $\vec{f}$ with action a. $\vec{f}'$ improves $\vec{f}$ if $G(X(0), a, \vec{f}) \leq v^{\vec{f}}_{X(0)}$ where $G(X(0), a, \vec{f}) = r^a_{X(0)} - g^{\vec{f}}\zeta + \sum_{X(\zeta)\in\Omega_n} p^a_{X(0),X(\zeta)} v^{\vec{f}}_{X(\zeta)}$.*

Proof. Let $\pi_{X(0)}$ $(\forall X(0) \in \Omega_n)$ be the steady state probability under the policy $\vec{f}'$, i.e., $\sum_{X(0)\in\Omega_n} \pi_{X(0)} = 1$. Since $G(X(0), a, \vec{f}) = r^a_{X(0)} - g^{\vec{f}}\zeta + \sum_{X(\zeta)\in\Omega_n} p^a_{X(0),X(\zeta)} v^{\vec{f}}_{X(\zeta)} \leq v^{\vec{f}}_{X(0)}$, the following formula is true:

$$\sum_{X(0)\in\Omega_n} \pi_{X(0)} r^a_{X(0)} + \sum_{X(0)\in\Omega_n} \pi_{X(0)} \sum_{X(\zeta)\in\Omega_n} p^a_{X(0),X(\zeta)} v^{\vec{f}}_{X(\zeta)}$$
$$-g^{\vec{f}}\zeta \leq \sum_{X(0)\in\Omega_n} \pi_{X(0)} v^{\vec{f}}_{X(0)}.$$

According to the definition of $r^a_{X(0)}$ in Theorem 1 and $\sum_{X(0)\in\Omega_n} \pi_{X(0)} = 1$, we obtain $g^{\vec{f}'}\zeta - g^{\vec{f}}\zeta + \sum_{X(\zeta)\in\Omega_n} \pi_{X(\zeta)} v^{\vec{f}}_{X(\zeta)a} \leq \sum_{x\in\Omega_n} \pi_{X(0)} v^{\vec{f}}_{X(0)}$, i.e., $g^{\vec{f}'} \leq g^{\vec{f}}$ which means that the new policy $\vec{f}'$ improves $\vec{f}$ by action a. □

The above two theorems illustrate that a smaller $v^{\vec{f}}_{X(0)}$ also implies a smaller $g^{\vec{f}}$ and a better policy $\vec{f}$.

The server selection procedure for a given number of servers n is iterative rather than a one-pass process. The policy $\vec{f}$ starts from the initial feasible policy $\vec{f}_0$ obtained by

Algorithm 2. Deviation values for all the involved states and the expected reward $g^{\vec{f}}$ are obtained by the subsequent Policy Evaluation algorithm which is based on Theorem 1. An improved policy $\vec{f}'$ is searched for by the following Policy Improvement algorithm which is based on Theorem 2. $\vec{f}'$ is set as $\vec{f}^*$ if the rejection probability is not higher than the system availability ξ. The policy evaluation and improvement procedures are repeated until $\vec{f}' = \vec{f}$. The optimal policy $\vec{f}^*$ and the expected reward per unit time g^* are obtained. The proposed server selection framework is depicted in Algorithm 3. The computational time complexity for the server selection algorithm is $O(\max(M^3, nNM^2)$.

Algorithm 3. Server Selection

```
   Input: $n, \vec{f}, \xi$
1  begin
2    $Flag \leftarrow True$;
3    while $Flag = True$ do
4      foreach $x \in \Omega_n$ do
5        Calculate $v^{\vec{f}}_{X(0)}$ and $g^{\vec{f}}$ for $\vec{f}$ using the Policy Evaluation algorithm;
6       Construct a new policy $\vec{f}'$ using the Policy Improvement algorithm;
7      Compute $P_R(\vec{f}')$ using Equation (7);
8      if $P_R(\vec{f}') \le 1 - \xi$ then
9        $\vec{f}^* \leftarrow \vec{f}'$, $g^* \leftarrow g^{\vec{f}}$;
10     if $\vec{f}' = \vec{f}$ then
11       $Flag \leftarrow False$;
12     else
13       $Flag \leftarrow True$, $\vec{f} \leftarrow \vec{f}'$;
14 return $\vec{f}^*, g^*$.
```

4.3.2 Policy Evaluation

Since $\min E(Y(t))$ is equivalent to $\min_{\vec{f}}\{tg^{\vec{f}} + v^{\vec{f}}_{X(0)}\}$, both $g^{\vec{f}}$ and $v^{\vec{f}}_{X(0)}$ depend on $\vec{f}$. Since ζ is the average time of all possible reachable states from $X(0)$ in the next decision epoch which is included in t (generally several epoches are included in t), Equation (9) cannot be applied directly to obtain $g^{\vec{f}}$ and $v^{\vec{f}}_{X(0)}$. A different $\vec{f}$ leads to different $g^{\vec{f}}$ and $v^{\vec{f}}_{X(0)}$. $g^{\vec{f}}$ is closely related only to $\vec{f}$. According to the optimal server selection principle, $v^{\vec{f}}_{X(0)}$ depends on both the policy $\vec{f}$ and the initial state $X(0)$. The above two theorems indicate that $g^{\vec{f}}$ and $v^{\vec{f}}_{X(0)}$ interact with each other. They can be obtained by Theorem 1. For all the M states in Fig. 2, there are $M+1$ unknown quantities (M deviation values $v^{\vec{f}}_{X(0)}$ and $g^{\vec{f}}$) in M equations for each step in the state transition in terms of Equation (9). In addition, the special state (0,0) is first recurrent by visiting (0,0) and (1,1) sequentially. According to the proof of Theorem 1, the deviation value of the first recurrence of (0,0) is $v^{\vec{f}}_{(0,0)}$. Let $\zeta_0, \zeta_1, \zeta_2$ be the time periods from (0,0) to (1,0), from (1,0) to (1,1) and from (1,1) to (0,0) respectively. In terms of Equation (9), we obtain another equation as

$$\begin{aligned} v^{\vec{f}}_{(0,0)} =& r^a_{(0,0)} - g^{\vec{f}}\zeta_0 + p^a_{(0,0),(1,0)}\Big[r^a_{(1,0)} - g^{\vec{f}}\zeta_1 \\ &+ p^a_{(1,0),(1,1)}(r^a_{(1,1)} - g^{\vec{f}}\zeta_2 + p^a_{(1,1),(0,0)}v^{\vec{f}}_{(0,0)})\Big]. \end{aligned} \tag{10}$$

The values of all the $M+1$ unknown quantities $v^{\vec{f}}_{X(0)}$ for each state and $g^{\vec{f}}$ can be calculated by the corresponding $M+1$ equations because the time ζ in each equation can be obtained by matrix Q (details will be given in the following).

TABLE 2
Transition Types by Combinations of the Variables z_1, z_2 and z_3

z_1	0	0	0	0	1	1	1	1
z_2	0	0	1	1	0	0	1	1
z_3	0	1	0	1	0	1	0	1
Transition type	(iii)	×	(iv)	(ii)	×	×	×	(i)

Before the calculation, we determine ζ, $p^a_{X(0),X(\zeta)}$ and $r^a_{X(0)}$ respectively.

- In the state sequence $(0,0), (1,0), (1,1), \ldots, (n+R, n)$, the position m of $X(0)$ is searched for. The time period ζ is $-\frac{1}{Q_{mm}}$ according to [4] where Q_{mm} is the mth diagonal element.
- For each state $X(0) = (j,i)$ where j represents the number of requests and i denotes the number of servers, there are at most four transitions as depicted in Fig. 2 which can be uniformly denoted as

$$\begin{aligned} X(\zeta) =& X(0) - S_f(i)S_f(j)(1-z_1)(1-z_3)e_0 \\ &- S_f(i)S_f(j)(1-S_f(j-i))(1-z_1)(1-z_3)z_2e_1 \\ &+ S_f(n+R-j)z_1z_2z_3e_0 \\ &+ S_f(n-i)(1-z_1)z_2z_3e_1, \end{aligned} \tag{11}$$

where $S_f(l)$ is a binary function.

$$S_f(l) = \begin{cases} 1 & \text{if } l > 0 \\ 0 & \text{if } l \le 0 \end{cases}. \tag{12}$$

$S_f(l) = 1$ if $l > 0$ and $S_f(l) = 0$ if $l \le 0$. Different combinations of the three binary variables z_1, z_2 and z_3 are shown in Table 2. $z_1 \in \{0,1\}$ denotes whether a new request can enter the system ($z_1 = 1$) or not ($z_1 = 0$) which corresponds to the first possible state in Section 3.1, i.e, the number of requests is less than $n+R$ ($z_1 = 1$) or is $n+R$ ($z_1 = 0$). $z_2 \in \{0,1\}$ represents whether the server operated by a is changed to the COLD state ($z_2 = 1$) or is kept in the HOT state ($z_2 = 0$). z_2 corresponds to the third and fourth possible states. $z_3 \in \{0,1\}$ indicates whether a new server is added ($z_3 = 1$) or not, which corresponds to the second possible state. $p^a_{X(0),X(\zeta)}$ is calculated in terms of matrix Q as follows:

$$\begin{aligned} p^a_{X(0),X(\zeta)} =& S_f(i)S_f(j)(1-z_1)(1-z_3)U_H(0)\zeta \\ &+ S_f(n+R-j)z_1z_2z_3\lambda\zeta + S_f(n-i) \\ &\times (1-z_1)z_2z_3\theta\zeta, \end{aligned} \tag{13}$$

where $U_H(0)$ can be obtained from $\vec{f}$.
- $r^a_{X(0)} = y(X(0))\zeta$ is calculated in terms of the definition of $r^a_{X(0)}$ in Theorem 1.

Therefore, the deviation values $v^{\vec{f}}_{X(0)}$ and $g^{\vec{f}}$ can be calculated with Equations (9) and (10).

We use the vector $\vec{v} = (v^{\vec{f}}_{(0,0)}, v^{\vec{f}}_{(1,0)}, v^{\vec{f}}_{(1,1)}, \ldots, v^{\vec{f}}_{(n+R,n)}, g^{\vec{f}})$ to denote the $M+1$ unknown quantities and the first M elements correspond to the state sequence $(0,0), (1,0),$

$(1,1),\ldots,(n+R,n)$. All coefficients of the unknown quantities are kept in a matrix $[V]_{(M+1)\times(M+2)}$. U_1 is the total service rates for servers in hot state. Since different combinations of z_1, z_2 and z_3 determine different states and coefficients, we use $k+z\times\min(n,j)+z_0$ where $z=(z_1-1)(1-z_2)(1-z_3)$ $+ z_1z_2z_3$, $z_0=z_2z_3+(z_3-1)(1-z_1)$ to denote the four possible locations of $X(\zeta)$ which are $k+\min(n,j)+1$, $k+1$, $k-1$, $k-\min(n,j)-1$. In terms of Equations (11) and (13), the values of $[V]_{(M+1)\times(M+2)}$ are obtained. By the elimination method, all unknown quantities are calculated. The policy evaluation process is formally described in Algorithm 4.

Algorithm 4. Policy Evaluation

```
   Input: $n, \vec{f}, Q, \lambda, \theta, \Omega_n$
 1 begin
 2   $[V]_{(M+1)\times(M+2)} \leftarrow \mathbf{0}$;
 3   for $i=0$ to $n$ do
 4     for $j=i$ to $n+R$ do
 5       $x \leftarrow (j,i), U_1 \leftarrow 0, k \leftarrow 1$;
 6       for $q=0$ to $j-1$ do
 7         if $q \le n$ then
 8           $k \leftarrow k+q+1$;
 9         else
10           $k \leftarrow k+n+1$;
11       $k \leftarrow k+i, \zeta \leftarrow -\frac{1}{Q_{kk}}$;
12       Calculate $y(x)$ using Equation (3);
13       $V_{kk} \leftarrow 1, V_{k(M+1)} \leftarrow \zeta, V_{k(M+2)} \leftarrow y(x)\zeta$;
         /* $V_{k(M+2)}$ is the actual reward $r^a_{(j,i)}$ of the
         state $(j,i)$.    */
14       for $q=0$ to $i$ do
15         $U_1 \leftarrow U_1 + \vec{f}((n+R+1)q - \frac{q(q+1)}{2} + j + 1)$;
16       $\mu_a \leftarrow \vec{f}((n+R+1)i - \frac{i(i+1)}{2} + j + 1)$;
17       for $z_1 = 0$ to $1$ do
18         for $z_2 = 0$ to $1$ do
19           for $z_3 = 0$ to $1$ do
20             Determine $x(\zeta)$ using Equ. (11);
21             Compute $p^a_{x,x(\zeta)}$ by Equ. (13);
22             $z \leftarrow (z_1-1)(1-z_2)(1-z_3) + z_1z_2z_3$;
23             $z_0 \leftarrow z_2z_3 + (z_3-1)(1-z_1)$;
24             $k' \leftarrow k + z\min(n,j) + z_0$;
25             $V_{kk'} \leftarrow -p^a_{x,x(\zeta)}$;
26     $V_{(M+1)(M+2)} \leftarrow V_{1(M+2)} + V_{12}V_{2(M+2)} + V_{23}V_{3(M+2)}$, $V_{(M+1)}$
   $(M+1) \leftarrow \frac{1}{-Q_{11}} - V_{12}(\frac{1}{Q_{22}} + V_{23}\frac{1}{Q_{33}})$, $V_{(M+1)1} \leftarrow 1 - V_{12}V_{23}$
   $V_{31}$ using Equation (10);
27   for $k=1$ to $M$ do
28     for $i=2$ to $M+1$ do
29       for $j=k$ to $M+2$ do
30         $V_{ij} \leftarrow \frac{V_{ij}V_{kk}}{V_{ik}} - V_{kj}$;
31   $\vec{v}_{(M+1)} \leftarrow \frac{V_{(M+1)(M+2)}}{V_{(M+1)(M+1)}}$;
32   for $i=M$ to $1$ do
33     $s \leftarrow 0$;
34     for $j=i+1$ to $M+1$ do
35       $s \leftarrow s + V_{ij}\vec{v}_j$;
36     $\vec{v}_i \leftarrow \frac{V_{i(M+2)}-s}{V_{ii}}$;
37 return $\vec{v}$.
```

The time complexity of computing $[V]_{(M+1)\times(M+2)}$ (steps 2-25) is $O(nM)$ and that of the elimination method is $O(M^3)$ (steps 26-36). Therefore, the time complexity of Algorithm 4 is $O(M^3)$ as $M \gg n$.

To illustrate the above process, the example used in Section 4.2 is used again with $n=2$ and a policy $\vec{f}=(0,0,0,0,0,6,6,6,6,4,4,4)$. The set of states is $\{x_1=(0,0), x_2=(1,0), x_3=(1,1),\ldots,x_M=(4,2)\}$. The other parameters are identical to those in Section 4.2. In terms of Algorithm 4, the deviation values of the states are {0.087,-0.914,-0.1.04,-0.696, -0.107,0.078,-0.386,-0.0074,-0.162,0.234,0.280,0.086} and expected reward $g^{\vec{f}}=1.149$.

4.3.3 Policy Improvement

In terms of Theorem 2, the current policy $\vec{f}$ can be improved if $G(X(0),a,\vec{f}) \le v^{\vec{f}}_{X(0)}$. An average service rate $\hat{\mu}$ can be estimated for a given system availability ξ and the value obtained n from Equation (14) since $\xi \le 1-p'_{n+R}$. The average rate on the n rates of $\vec{f}$, i.e., $\vec{\boldsymbol{\mu}}=(\mu_1,\mu_2,\ldots,\mu_n)$, is $\overline{\mu}=\frac{\sum_{i=1}^n \mu_i}{n}$. $\hat{\mu} \le \overline{\mu}$ means that a is feasible to update $\vec{f}$ to the new $\vec{f}'$. The new $\vec{f}'$ is updated with the $\min_{X(0)\in\Omega_n}$ $\sum G(X(0),a,\vec{f})$.

Assume the n servers are homogeneous with the same service rate $\hat{\mu}$ when the system availability is ξ, arrival rate λ and queue capacity R. The system would be a traditional $M/M/n/n+R$ queuing model [26]. The probability p'_i for the state or the number of requests i ($i=0,1,\ldots,n+R$) can be easily calculated by

$$p'_i = S_f(n+1-i)\frac{\lambda^i}{i!\hat{\mu}^i}p'_0 + S_f(i-n)\frac{\lambda^i}{n^{(i-n)}n!\hat{\mu}^i}p'_0,$$

where $S_f(l)$ is determined in Equation (12). Since $\sum_{i=0}^{n+R} p'_i = 1$, we can obtain $p'_0 = (\sum_{i=0}^{n}\frac{\lambda^i}{i!\hat{\mu}^i} + \frac{n^n}{n!}\sum_{i=n+1}^{n+R}\frac{\lambda^i}{n^i\hat{\mu}^i})^{-1}$. In addition, $\frac{n^n}{n!}\sum_{i=n+1}^{n+R}\frac{\lambda^i}{n^i\hat{\mu}^i} = \frac{n^n}{n!}\frac{(\frac{\lambda}{n\hat{\mu}})^{n+1}(1-(\frac{\lambda}{n\hat{\mu}})^R)}{1-\frac{\lambda}{n\hat{\mu}}} = \frac{\lambda^{(n+1)}[(n\hat{\mu})^R-\lambda^R]}{n!n^R\hat{\mu}^{(n+R)}(n\hat{\mu}-\lambda)}$. Therefore, $p'_0 = (\sum_{i=0}^{n}\frac{\lambda^i}{i!\hat{\mu}^i} + \frac{\lambda^{(n+1)}[(n\hat{\mu})^R-\lambda^R]}{n!n^R\hat{\mu}^{(n+R)}(n\hat{\mu}-\lambda)})^{-1}$. For the artificial homogenous system, the rejection probability is

$$p'_{n+R} = \frac{\lambda^{n+R}}{n^R n!\hat{\mu}^{n+R}}p'_0. \tag{14}$$

The system availability is ξ which implies that $p'_{n+R} \le 1-\xi$. Because of its convenience for solving unknown equations, Matlab R2010b is applied to estimate the service rate $\hat{\mu}$ with the following equation.

$$(1-\xi)\left(\sum_{i=0}^{n}\frac{\lambda^i}{i!\hat{\mu}^i} + \frac{\lambda^{(n+1)}[(n\hat{\mu})^R-\lambda^R]}{n!n^R\hat{\mu}^{(n+R)}(n\hat{\mu}-\lambda)}\right) = \frac{\lambda^{n+R}}{n^R n!\hat{\mu}^{n+R}}. \tag{15}$$

If $\hat{\mu} > \overline{\mu}$, $\vec{f}$ cannot be improved by conducting a, i.e., a is unfeasible.

Let $\vec{f}'$ be the best policy of the current improvement which is initialized to $\vec{f}$. For the current policy $\vec{f}$, $\vec{v}_i$(i=1, ..., M) are obtained by Algorithm 4. Each server $\vec{S}_{[i]}$ ($i=1,\ldots,n$) is tested for a replacement by any of the servers $S_{i'}$ ($i' \in J_C$). For the updated policy $\vec{f}''$, $G(x_k, S_{i'}, \vec{f}'')(k=1,\ldots,M)$ is calculated in terms of Theorem 2. The process is repeated until all servers in $\vec{S}_{[i]}$ are tested. The policy matrix $\mathbf{F}_{[n(N-n)+1]\times M}$ contains all possible policies (which also incudes the updated policy $\vec{f}''$ and $\vec{f}$) corresponding to what deviation values are kept in matrix $\mathbf{G}_{[n(N-n)+1]\times M}$. $\vec{f}'(i)$ is replaced by F_{ki} if $G_{min}=G_{ki}$. The policy improvement procedure is formally depicted in Algorithm 5. Though it is hard to estimate the computational time complexity of Algorithm 5 because of the complicated computation of $\hat{\mu}$ using Equation (15), it is much faster in practical testing. As for Algorithm 5, the time complexity is $O(n(N-n)M)$ since $M < n(N-n)M$.

Algorithm 5. Policy Improvement

```
   Input: $n, \vec{f}, \vec{v}, \vec{S}, J_C$
1  begin
2    $\vec{f}' \leftarrow \vec{f}$, $\mathbf{G}_{[n(N-n)+1]\times M} \leftarrow \mathbf{0}$;
3    $l \leftarrow 1$, $\mathbf{F}_{[n(N-n)+1]\times M} \leftarrow \mathbf{0}$;
4    Calculate $\hat{\mu}$ using Equation (15);
5    for $i = 1$ to $M$ do
6      $G_{1i} \leftarrow \vec{v}_i$;
7    $\vec{F}_1 \leftarrow \vec{f}'$;
8    for $i = 1$ to $n$ do
9      foreach $j \in J_C$ do
10       $\vec{\boldsymbol{\mu}}_{[i]} \leftarrow \mu_j$;
11       $\overline{\mu} \leftarrow \frac{\sum_{i=1}^{n} \vec{\boldsymbol{\mu}}_{[i]}}{n}$;
12       if $\overline{\mu} \geq \hat{\mu}$ then
13         $l \leftarrow l + 1$;
14         Construct $\vec{f}''$ in terms of $\vec{\boldsymbol{\mu}}$;
15         $\vec{F}_l \leftarrow \vec{f}''$;
16         for $k = 1$ to $M$ do
17           Compute $G(x_k, S_{i'}, \vec{f})$ by Theorem 2;
18           $G_{lk} \leftarrow G(x_k, S_{i'}, \vec{f})$;
19   for $i = 1$ to $M$ do
20     $G_{min} \leftarrow G_{1i}, k \leftarrow 1$;
21     for $j = 2$ to $l$ do
22       if $G_{ji} < G_{min}$ then
23         $G_{min} \leftarrow G_{ji}, k \leftarrow j$;
24     $\vec{f}'(i) \leftarrow F_{ki}$;
25 return $\vec{f}'$.
```

To illustrate the above process, the example used in Section 4.3.2 is used with policy $\vec{f}$=(0,0,0,0,0,6,6,6,6,4,4,4). The service rate $\hat{\mu} = 1.326$ is evaluated in terms of Equation (15). The two servers with rates 6 and 4 are in the HOT state and $J_C = \{3, 4, 5, 6\}$ contains the substitution candidates. There are eight candidate policies based on J_C for each one of the servers (one substitution for each server). A new policy $\vec{f}'$= (0,0,0,0,0,0.1,0.1,0.1,0.1,4,4,4) is determined by choosing the minimal $G_{ki} (i = 1, \ldots, M)$ using Algorithm 5. $\vec{f}'$ is reevaluated by Algorithm 4. There are eight candidate policies to improve $\vec{f}'$ among which four satisfy the $\hat{\mu}$ constraint. The policy $\vec{f}'$=(0,0,0,0,0,0.1,0.1,0.1,0.1,4,4,4) is selected. Since there is no further improvement, Algorithm 3 stops and returns the optimal policy $\vec{f}^*$=(0,0,0,0,0,0.1,0.1,0.1,0.1,4,4,4) with $g^* = 0.497$. Details are given in Table 3.

5 Experimental Evaluation

Since there are no parameters and no algorithm component candidates in the proposed BETP algorithm, no parameter or component calibration is needed. This is actually a desirable characteristic of the proposed BETP algorithm. However, there are many system parameters which might affect the performance of the proposal. First we test the effect of the system parameters and then four similar algorithms are compared with the BETP algorithm over both random and real instances [31]. All compared algorithms are coded in Matlab R2010b and run on an Intel Core i5-3470 CPU @3.20 GHz with 8 GBytes of RAM.

TABLE 3
Optimal Policy Procedure

Iteration times	$\vec{f}'$	$g^{\vec{f}'}$
0	(0,0,0,0,0,6,6,6,6,4,4,4)	1.149
1	(0,0,0,0,0,0.1,0.1,0.1,0.1,4,4,4)	0.497
2	(0,0,0,0,0,0.1,0.1,0.1,0.1,4,4,4)	0.497

Equation (3) indicates the objective value for a state, the performance is measured by the average expected value $\overline{y}$ of all the states calculated by

$$\overline{y} = \sum_{i=1}^{M} \pi_i \left[\beta W(N_r^i(t)) + (1-\beta) \sum_{j=0}^{N_H^i(t)} W(P_{[j]}) \right] \times 100\%, \tag{16}$$

where π_i is the steady probability of the ith state determined by Equations (5) and (6). $N_r^i(t)$ and $N_H^i(t)$ are the number of requests and servers in the ith state respectively.

5.1 System Parameters Influence on Performance

Similar to the problem of performance analysis for cloud centers in [7], the effects of the system parameters on the proposal are analyzed on randomly generated testing instances. The studied system parameters are: the system availability parameter $\xi \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$, the arrival rate $\lambda \in \{\{1, \ldots, 10\}, \{11, \ldots, 20\}, \{21, \ldots, 30\}\}$, the maximum number of servers $N \in \{5, 10, 15, 20\}$, the queue capacity $R \in \{10, 20, 30, 40\}$, the delay rate $\theta \in \{10, 20, 30\}$ and the range of service rate $\mu \in \{(0, 5], [5, 10], [10, 15], [15, 20]\}$ (no test is needed for β because it is the weight of the objectives and it depends on the preferences of users). There are 2,880 parameter combinations in total. Three instances are generated randomly for each arrival rate λ and every service rate μ, i.e., nine instances are generated for each combination. Therefore, 25,920 instances in total are tested for the combinations of all parameter values.

Experimental results are analyzed by the multi-factor analysis of variance (ANOVA) statistical technique. Three main hypotheses (normality, homoscedasticity, and independence of the residuals) are analyzed from the residuals of the experiments. All three hypotheses can be accepted considering the well-known robustness of the ANOVA technique. The p-values are less than 0.05 which means that all the studied factors have a significant effect on the response variables at the 95 percent confidence level within the ANOVA.

The means plot of the six studied factors on the average expected value $\overline{y}$ with 95 percent HSD (Tukey's Honest Significance Differences) intervals are given in intervals is shown in Fig. 3.

From Fig. 3, it can be observed that:

i) ξ exerts a great influence on $\overline{y}$. With an increase in ξ, $\overline{y}$ decreases. The differences are statistically significant. $\overline{y}$ is the minimum (about 44 percent) when $\xi = 0.9$. The reason lies in that a higher ξ implies that

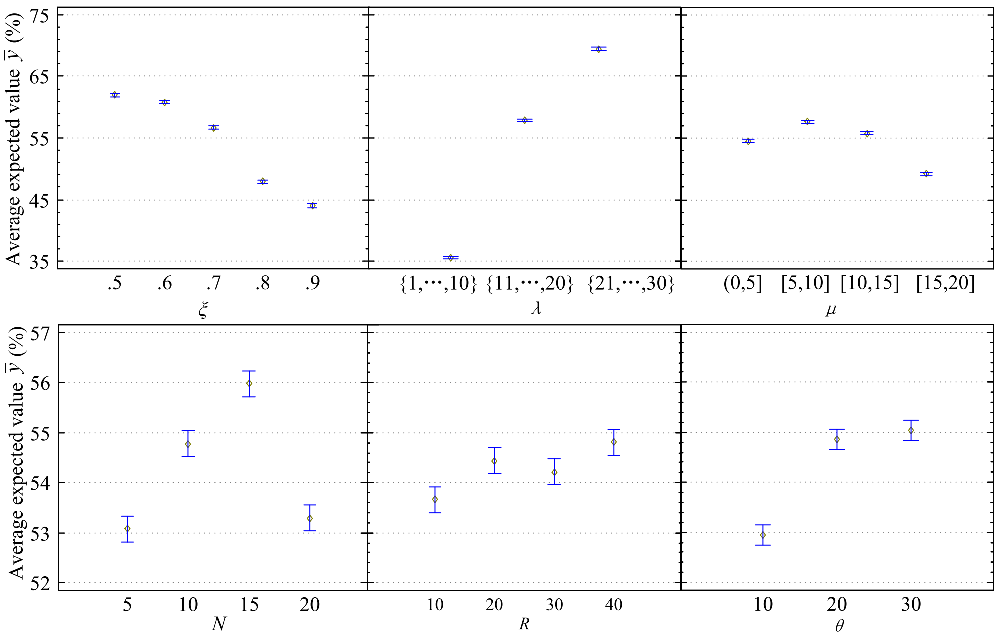

Fig. 3. Means plot of the six studied system parameters with 95 percent confidence level Tukey HSD intervals.

faster servers are provided which further decreases the rejection probability.

ii) λ exerts a great influence on $\overline{y}$. With an increase in the upper bound for λ, $\overline{y}$ increases with statistically significant differences. $\overline{y}$ is the minimum (about 35 percent) when λ takes values from $\{1, \ldots, 10\}$. The reason lies in that fewer arriving requests means that more requests are accepted.

iii) Similarly, μ has a big impact on $\overline{y}$ and the differences are statistically significant. However, the tendency of $\overline{y}$ is not monotone with an increase in μ because increasing μ means an increase in power consumption and a decrease in the number of requests in the system which cannot be determined in advance. $\overline{y}$ is the minimum when μ takes a value from [15,20].

iv) N has little influence on $\overline{y}$. It seems that a greater N results in more choices for the arriving requests and possibly shorter response time. However, a greater N also means more power consumption. According to Equation (3), the tradeoff between the response time and the power consumption makes N insensitive to $\overline{y}$.

v) R slightly influences $\overline{y}$. In Equation (3), only $W(N_r(t))$ is closely related to R with $W(N_r(t)) = \frac{N_r(t)}{n+R}$. A greater capacity R shows that more requests can be accepted by the system, i.e., greater $N_r(t)$. On the contrary, a lower R results in a lower $N_r(t)$. Therefore, the differences of the ratio $W(N_r(t))$ are not statistically significant in R.

vi) θ has little influence on $\overline{y}$ with differences being statistically insignificant. The reason is similar to that of N.

5.2 Performance Comparison

Server selection is crucial for the performance of algorithms for scheduling requests. In this paper, we compare four BETP algorithms which are based on the BETP framework: BETP, MAX, MIN and RAND. BETP is the algorithm proposed in this paper. Similarly to the strategy adopted in [12], MIN selects the n slowest servers. MAX selects the n fastest servers for the requests. RAND selects n servers randomly. The RATE dispatch policy given in [28] and [29] selects servers with a probability $\frac{\mu_i^r}{\sum_{i=1}^N \mu_i^r}$. For the considered problem, we found that there is little difference when $r = 2$ and $r = 3$. Therefore, we just set $r \rightarrow 2$ to select n servers using the RATE policy with the probability of $\frac{\mu_i^2}{\sum_{i=1}^N \mu_i^2}$. Both random instances and real instances from Alicloud[31] are compared on the algorithms, respectively.

5.2.1 *Performance Comparison Over Random Instances*

In terms of the above analysis, instance parameters ξ, λ, μ take the same values as those in Section 5.1. β takes values from {0.3, 0.6, 0.9}. N, R, θ are all set to 10 because of their statistically nonsignificant differences in $\overline{y}$ as per the previous experiment. There are 180 combinations. Because there are no benchmark instances for the considered problem, nine instances are randomly generated for each combination. Therefore, 1,620 instances are conducted on each of the five algorithms with the results shown in Fig. 4.

Fig. 4 indicates that when the arrival rate λ takes a value from $\{1, \ldots, 10\}$, MAX obtains the smallest $\overline{y}$ while MIN obtains the largest. BETP and RAND perform similarly. For the other two cases, BETP obtains the smallest $\overline{y}$ compared to MAX, MIN, RAND and RATE. In other words, with an increase in service arrival rate λ, BETP becomes more effective than the other three algorithms. MIN always demonstrates the worst performance among the five compared algorithms.

With an increase in ξ from 0.5 to 0.9, BETP always results in the smallest $\overline{y}$ whereas MIN obtains the largest. RATE is always worse than MAX with a larger $\overline{y}$ while RATE is better than RAND with a smaller $\overline{y}$. The higher values of ξ demonstrate the superiority of BEPT.

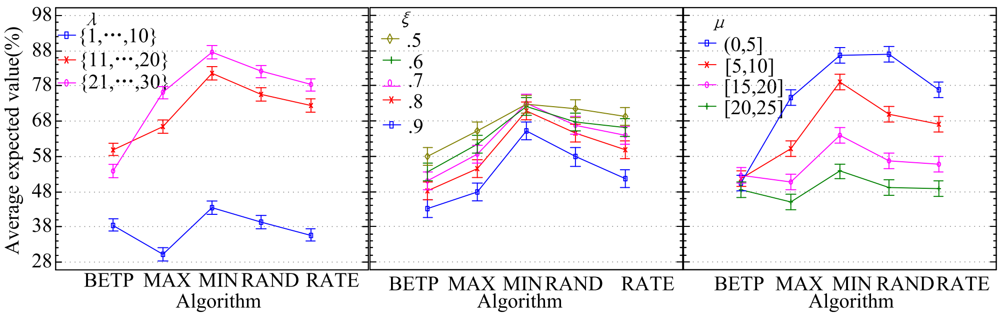

Fig. 4. Mean plots of the interaction between the five compared algorithms and the three main system parameters with 95 percent confidence levels Tukey HSD intervals.

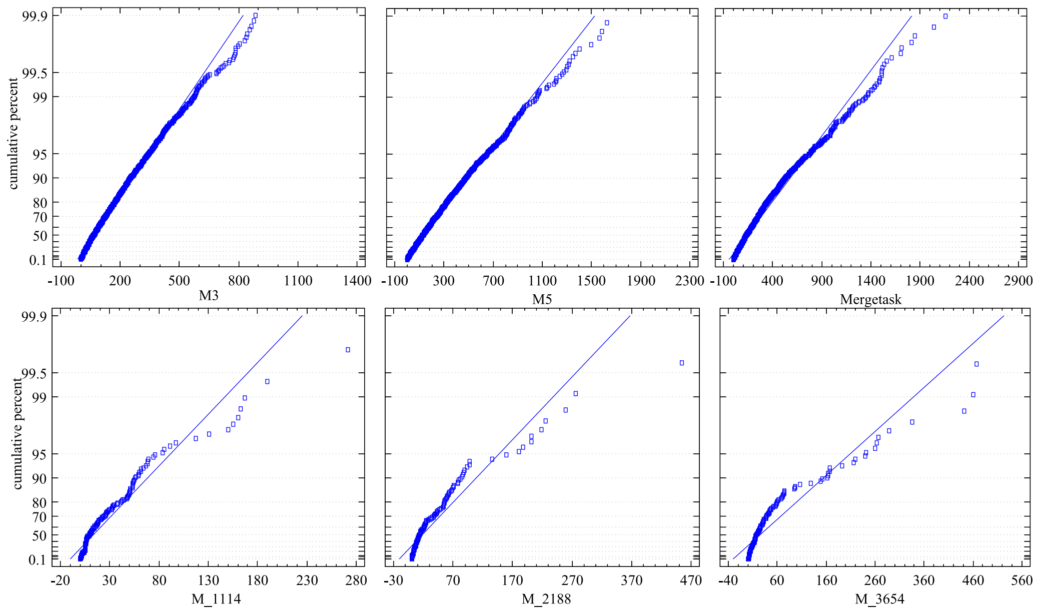

Fig. 5. Exponential probability plot for different tasks and machines.

RAND has the largest $\overline{y}$ which is even a little higher than MIN when μ takes a value from the interval (0,5]. For the other three cases, MIN obtains the largest $\overline{y}$. RATE, RAND are worse than MAX. BETP is much more robust than the other three algorithms, i.e., with an increase in μ, the performance of BETP fluctuates less than the other three. When μ takes values from [20, 25], MAX outperforms BETP and the latter is similar to RAND. The reason lies in that arriving requests can be processed by faster servers in a shorter time no matter which strategy is adopted.

To compare the algorithms comprehensively, the average performances on effectiveness (the average expected value) and efficiency (CPU time) are shown in Table 4. According to Table 4, we can observe that BETP obtains the smallest $\overline{y}$, 50.8 percent, followed by 57.5 percent of MAX. MIN obtains the largest $\overline{y}$ 70.8 percent. $\overline{y}$ of RAND is 65.7 percent which is better than MIN but worse than RATE. $\overline{y}$ of RATE is 62.1 percent which is worse than that of MAX. BETP has the longest CPU time of 2.245 seconds among the five algorithms. The CPU time of MAX, MIN, RAND and RATE is 0.246s, 0.272s , 0.444s, and 0.260s respectively. However, the CPU time of BETP is acceptable in practice.

5.2.2 Performance Comparison over Real Instances

To evaluate the performance in real systems, the real production Cluster-trace-v2018 [31] published by the Alibaba Group is analyzed which contains eight-day sample data from one of the production clusters. By analyzing the $start\_time$ [32] of requests, the arriving time interval is obtained. According to $start\_time$ and end_time [33], the execution times of all servers are calculated. The distributions of the time interval of each request and the execution time of each server is exponential with different arrival rates and service rates. The distributions of the time interval of each request are exponential which implies that the arrival rates are Poison distributed. Fig. 5 depicts the exponential cumulative percents of the time intervals of three different independent request types $M3, M5, Mergetask$ [32] and

TABLE 4
Algorithm Comparisons

	BETP	MAX	MIN [12]	RAND	Rate-based [28]
$\overline{y}$ (%)	50.8	57.5	70.8	65.7	62.1
CPU time	2.145	0.246	0.272	0.444	0.260

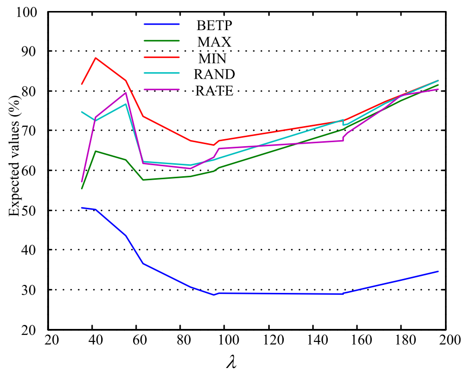

Fig. 6. Expected values among the compared algorithms.

those of the execution time of three different servers M_1114, M_2188, M_3654 [33].

The arrival rate λ is analyzed by different types of requests [32]. The service rates $\mu_{[i]}(i = 1, \ldots, N)$ are evaluated by different types of servers [33]. Similar to the random instances, N, R, θ are set to 10. The service rates are obtained with {14.5, 15.4, 16.9, 17.4, 18.5, 19.4, 20.4, 21.3, 22.8, 23.9} in terms of [33]. Fig. 6 shows the expected values of the five compared algorithms. It can be observed that BETP algorithm always obtains the smallest values as λ increases. MAX is always better than the MIN and RAND algorithms. RATE fluctuates as λ increases. It is obvious that all the compared algorithms have similar performances over both random and real instances.

6 Conclusion and Future Work

In this paper, a queuing system is constructed to balance the expected response time and power consumption for cloud centers with heterogeneous servers and setup time. Heterogeneity results in different expected response time and power consumption for different servers. The rejection probability is constrained by system availability which determines the number of servers for stochastically arriving requests. The proposed BETP algorithm obtains a suitable number of servers and selects the appropriate server types to balance the expected response time and power consumption. The best policy can be iteratively calculated by the infinitesimal generator matrix of the state transition process. Among all the studied system parameters, request arrival rate, system availability and service rates of servers have a great influence on the objective (the average expected value) with statistically significant differences. BETP outperforms the other three algorithms which are also based on the BETP framework over a comprehensive set of random and real instances.

For heterogeneous cloud centers there are still many open issues that are worth studying in the future, e.g., the impatience of service consumers in cloud centers and segmentation of servers into multiple queues for quickly processing service requests etc.

Acknowledgments

This work is supported by the National Key Research and Development Program of China Grant No. 2017YFB1400801, the National Natural Science Foundation of China Grant Nos. 61572127, 61872077, 61832004 and Collaborative Innovation Center of Wireless Communications Technology. Rubén Ruiz is partly supported by the Spanish Ministry of Science, Innovation, and Universities, under the project “OPTEP-Port Terminal Operations Optimization” (No. RTI2018-094940-B-I00) financed with FEDER funds.

References

[1] A. Shehabi *et al.*, “United States data center energy usage report,” United States Data Center Energy Usage Report, Lawrence Berkeley National Laboratory, 2016.
[2] L. Wang *et al.*, “Cloud computing: A perspective study,” *New Gener. Comput.*, vol. 28, no. 2, pp. 137–146, 2010.
[3] A. Gandhi, M. Harchol-Balter, and I. Adan,“Server farms with setup costs,” *Perform. Eval.*, vol. 67, no. 11, pp. 1123–1138, 2010.
[4] H. C. Tijms, *A First Course in Stochastic Models*. Hoboken, NJ, USA: Wiley, 2004.
[5] R. Ghosh, K. S. Trivedi, V. K. Naik, and D. S. Kim, “End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach,” in *Proc. IEEE Pacific Rim Int. Symp. Depend. Comput.*, 2010, pp. 125–132.
[6] Y. Xia, M. C. Zhou, X. Luo, Q. Zhu, J. Li, and Y. Huang, “Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds,” *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 162–170, Jan. 2015.
[7] H. Khazaei, J. Misic, and V. B. Misic, “Performance analysis of cloud computing centers using $M/G/m/m+r$ queuing systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 5, pp. 936–943, May 2012.
[8] H. Khazaei, J. Misic, V. B. Misic, and S. Rashwand, “Analysis of a pool management scheme for cloud computing centers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 5, pp. 849–861, May 2013.
[9] T. Atmaca, T. Begin, A. Brandwajn, and H. Castel-Taleb, “Performance evaluation of cloud computing centers with general arrivals and service,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2341–2348, Aug. 2016.
[10] H. Khazaei, J. Ic, V. B. Ic, and N. B. Mohammadi, “Modeling the performance of heterogeneous IAAS cloud centers,” in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. Workshops*, 2013, pp. 232–237.
[11] C. Misra and P. K. Swain, “Performance analysis of finite buffer queueing system with multiple heterogeneous servers,” in *Proc. Int. Conf. Distrib. Comput. Internet Technol.*, 2010, pp. 180–183.
[12] F. Alves, H. C. Yehia, and L. Pedrosa, “Upper bounds on performance measures of heterogeneous M/M/C queues,” *Math. Problems Eng.*, vol. 2011, no. 4, pp. 1884–1902, 2011.
[13] A. Tirdad, W. K. Grassmann, and J. Tavakoli, “Optimal policies of M (t)/M/ C/C queues with two different levels of servers,” *Eur. J. Oper. Res.*, vol. 249, no. 3, pp. 1124–1130, 2016.
[14] Z. Zhangab, “Analysis of job assignment with batch arrivals among heterogeneous servers,” *Eur. J. Oper. Res.*, vol. 217, no. 1, pp. 149–161, 2012.
[15] Y. Tian, C. Lin, and K. Li, “Managing performance and power consumption tradeoff for multiple heterogeneous servers in cloud computing,” *Cluster Comput.*, vol. 17, no. 3, pp. 943–955, 2014.
[16] X. Qiu, Y. Dai, Y. Xiang, and L. Xing, “A hierarchical correlation model for evaluating reliability, performance, and power consumption of a cloud service,” *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 46, no. 3, pp. 401–412, Mar. 2016.
[17] P. Sun, Y. Dai, and X. Qiu, “Optimal scheduling and management on correlating reliability, performance, and energy consumption for multiagent cloud systems,” *IEEE Trans. Rel.*, vol. 66, no. 2, pp. 547–558, Jun. 2017.
[18] J. Cao, H. Kai, K. Li, and A. Y. Zomaya, “Optimal multiserver configuration for profit maximization in cloud computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1087–1096, Jun. 2013.
[19] J. Mei, K. Li, A. Ouyang, and K. Li, “A profit maximization scheme with guaranteed quality of service in cloud computing,” *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3064–3078, Nov. 2015.
[20] G. Bolch, S. Greiner, H. D. Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications*, 2nd ed. Hoboken, NJ, USA: Wiley-Interscience, 2006.
[21] V. Rykov and D. Efrosinin, “Optimal control of queueing systems with heterogeneous servers,” *Queueing Syst.*, vol. 46, pp. 389–407, 2004.

[22] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, “Low-power CMOS digital design,” *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.

[23] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, “Theoretical and practical limits of dynamic voltage scaling,” in *Proc. 41st Des. Autom. Conf.*, 2004, pp. 868–873.

[24] Z. G. Zhang and N. Tian, “Analysis of queueing systems with synchronous single vacation for some servers,” *Queueing Syst.*, vol. 45, no. 2, pp. 161–175, 2003.

[25] V. Rykov and M. Y. Kitaev, “Controlled queueing systems,” *J. Appl. Math. Stochastic Anal.*, vol. 8, no. 4, pp. 433–435, 1995.

[26] D. Gross, J. F. Shortie, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory, Fourth Edition*. New York, NY, USA: Wiley-Interscience, 2013.

[27] K. Li, “Improving multicore server performance and reducing energy consumption by workload dependent dynamic power management,” *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 122–137, Apr.–Jun. 2016.

[28] L. Na and D. A. Stanford, “Multi-server accumulating priority queues with heterogeneous servers,” *Eur. J. Oper. Res.*, vol. 252, no. 3, pp. 866–878, 2016.

[29] S. Doroudi, R. Gopalakrishnan, and A. Wierman, “Dispatching to incentivize fast service in multi-server queues,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 39, no. 3, pp. 43–45, 2011.

[30] A. Gandhi, S. Doroudi, M. Harchol-Balter, and A. Scheller-Wolf, “Exact analysis of the M/M/k/setup class of markov chains via recursive renewal reward,” *Queueing Syst. Theory Appl.*, vol. 77, no. 2, pp. 177–209, 2014.

[31] Alibaba, 2018. [Online]. Available: https://github.com/alibaba/clusterdata

[32] Alibaba, 2018. [Online]. Available: http://clusterdata2018pubcn.oss-cn-beijing.aliyuncs.com/batch_task.tar.gz

[33] Alibaba, 2018. [Online]. Available: http://clusterdata2018pubcn.oss-cn-beijing.aliyuncs.com/batch_instance.tar.gz

Shuang Wang received the BSc degree from the College of Sciences, Nanjing Agricultural University, in 2015. She is currently working toward the PhD degree with the School of Computer Science and Engineering, Southeast University, Nanjing, China. Her main interests include cloud computing, task scheduling.

Xiaoping Li (M’09-SM’12) received the BSc and MSc degrees in applied computer science from the Harbin University of Science and Technology, in 1993 and 1999 respectively, and the PhD degree in applied computer science from the Harbin Institute of Technology, in 2002. He is a full professor with the School of Computer Science and Engineering, Southeast University, Nanjing, China. He is the author or co-author over more than 100 academic papers, some of which have been published in international journals such as the *IEEE Transactions on Parallel and Distributed Systems*; *IEEE Transactions on Services Computing*; *IEEE Transactions on Cybernetics*; *IEEE Transactions on Automation Science and Engineering*; *IEEE Transactions on Cloud Computing*; *IEEE Transactions on Systems, Man and Cybernetics: Systems*; *Information Sciences*; *Omega*, *European Journal of Operational Research*; *International Journal of Production Research*; *Expert Systems with Applications* and *Journal of Network and Computer Applications*. His research interests include scheduling in cloud computing, scheduling in cloud manufacturing, service computing, big data and machine learning.

Rubén Ruiz is full professor of Statistics and Operations Research at the Universitat Politècnica de València, Spain. He is co-author of more than 80 papers in International Journals and has participated in presentations of more than a hundred and fifty papers in national and international conferences. He is editor of the Elseviers journal *Operations Research Perspectives (ORP)* and co-editor of the JCR-listed journal *European Journal of Industrial Engineering (EJIE)*. He is also associate editor of other important journals like *TOP* as well as member of the editorial boards of several journals most notably *European Journal of Operational Research* and *Computers and Operations Research*. He is the director of the Applied Optimization Systems Group (SOA, http://soa.iti.es) at the Instituto Tecnológico de Informática (ITI, http://www.iti.es) where he has been principal investigator of several public research projects as well as privately funded projects with industrial companies. His research interests include scheduling and routing in real life scenarios.