Hindawi
Complexity
Volume 2021, Article ID 6646881, 18 pages
https://doi.org/10.1155/2021/6646881

Research Article

Performance Optimization of Cloud Data Centers with a Dynamic Energy-Efficient Resource Management Scheme

Yu Cui,[1,2] Shunfu Jin,[1] Wuyi Yue,[3] and Yutaka Takahashi[4]

[1]*School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China*
[2]*College of Mathematics and Information Science & Technology, Hebei Normal University of Science & Technology, Qinhuangdao 066004, China*
[3]*Department of Intelligence and Informatics, Konan University, Kobe 658-8501, Japan*
[4]*Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan*

Correspondence should be addressed to Shunfu Jin; jsf@ysu.edu.cn

Received 24 December 2020; Revised 18 January 2021; Accepted 21 January 2021; Published 13 February 2021

Academic Editor: Yongsheng Hao

As an advanced network calculation mode, cloud computing is becoming more and more popular. However, with the proliferation of large data centers hosting cloud applications, the growth of energy consumption has been explosive. Surveys show that a remarkable part of the large energy consumed in data center results from over-provisioning of the network resource to meet requests during peak demand times. In this paper, we propose a solution to this problem by constructing a dynamic energy-efficient resource management scheme. As a way of saving energy as well as maintaining cloud user's quality of experience, the scheme presents a multitier cloud architecture by configuring physical machines (PMs) into two pools: a hot (running) pool and a warm (turned on, but in dynamic sleep) pool. Each PM is configured with a resource search engine (RSE) that finds an available virtual machine (VM) for the request, and a synchronous sleep mechanism is introduced to the warm pool. To analyze the end-to-end performance of the cloud system's service with the proposed scheme, we establish a hybrid queueing system composed of three stochastic submodels by using a matrix-geometric solution. Accordingly, the average latency of requests and the energy-saving rate of the system are derived. Through numerical results, we show the influence of the synchronous sleep mechanism on the system performance. Moreover, from the perspective of economics, we build a system cost function to study the trade-off between different performance measures. An improved Salp Swarm Algorithm (SSA) is presented to minimize the system cost and optimize the sleep parameter.

1. Introduction

As a direct result of the rapid growth in the number of cloud users, some cloud providers have already built large numbers of data centers to satisfy the resources demands [1]. The consequences are massive increases in energy consumption, an excessive increase in carbon emissions, and a reduction in benefits for the cloud providers [2]. Statistical results show that the average data center can consume as much energy as 25,000 ordinary households [3]. Therefore, based on the concept of green computing, obviously, the development of a greener, more energy-efficient resource management mechanism for cloud systems is becoming more desirable [4, 5].

The main contributions of this paper are summarized as follows:

(i) We present a cloud architecture composed of a task-scheduling decision layer, a resource-provisioning layer, and an actual service layer. Over the multitier cloud architecture, we propose an energy-efficient resource management scheme with a synchronous sleep mechanism.

(ii) We establish a queueing model composed of three subqueues to capture the proposed scheme. By using a Markov chain-based approach, we derive two performance measures: the average latency of requests and the energy-saving rate of the system.

(iii) Taking into account the trade-off between the average latency of requests and the energy-saving rate of the system, we build a system cost function and present an improved Salp Swarm Algorithm (SSA) to optimize the sleep mechanism.

2. Related Work

In this section, we review the related work on energy conservation research in cloud systems based on virtualization technology, sleep mode, and multitier cloud architecture. And then, we set forth the motivation for our research.

2.1. Virtualization Technology-Based Energy Conservation Research. In recent years, for utilizing the physical resource optimally, the study of energy conservation strategy for virtual machine (VM) configuration, migration, and consolidation has become a focus of energy conservation research in cloud systems.

Auday et al. considered migration and placement of VMs to enhance the energy efficiency in cloud infrastructure. In order to minimize the additional energy consumption generated by the VM migration, they proposed a distributed approach to an energy-efficient dynamic VM consolidation policy. The approach determined which VMs are migrated and where the selected VMs for migration are placed [6]. For solving the problem of under-utilization of servers in a cloud system, Zakarya et al. used VM consolidation to reduce the number of hosts in use. They explored the impact of VM allocation on energy efficiency and proposed a dynamic VM migration approach, in which the VMs are migrated only if the migration cost could be recovered [7].

Through modeling the energy-aware allocation and consolidation, Ghribi et al. presented an optimal allocation algorithm with a consolidation algorithm relying on migration of VMs to minimize the overall energy consumption in the cloud system. The allocation algorithm was solved as a bin-packing problem aiming to minimize the energy consumption. The consolidation algorithm was based on a linear and integer formulation of VM migration to adapt the placement for released resources [8]. Aiming to save energy and minimize resource wastage, Sharma et al. proposed a multiobjective VM allocation and migration scheme, in which the allocation of VMs was carried out using a hybrid approach of a genetic algorithm and particle swarm optimization [9]. Based on the virtualization technology, the above research improved the utilization rate of the physical resources in use and contributed to the energy conservation.

2.2. Sleep Mode-Based Energy Conservation Research. The sleep mode-based energy conservation strategy is implemented by switching the idle server to a low-power sleep state for the purpose of reducing idle energy consumption in the cloud system.

Jin et al. proposed a clustered VM allocation strategy on the resource layer of the cloud system based on a sleep mode with a wake-up threshold. By establishing a queue with an N-policy and asynchronous vacations of partial servers, they derived the performance measures in terms of the average latency of requests and the energy-saving rate of the system [10]. By using a hybrid shuffled frog leaping algorithm, Luo et al. proposed a dynamic VM allocation scheme, which applied a live VM migration strategy and switched some free resource nodes into a sleep mode to reduce energy consumption [3]. Farahnakian et al. developed a dynamic VM consolidation method to solve the optimization problem for setting the number of active hosts based on the utilization of existing resources. The proposed method could make a decision on when to switch a host into the working or sleep mode [11].

Sridharshini et al. proposed an energy-aware scheduling algorithm and a live migration algorithm to efficiently utilize the resources in a cloud system. These two algorithms were used to consolidate heterogeneous workloads to minimize the number of physical machines (PMs) and switch the idle PMs to the sleep mode to reduce energy consumption [12]. The studies mentioned above showed a certain degree of enhanced energy efficiency due to the introduction of a sleep mode.

2.3. Energy Conservation Research under a Multitier Cloud Architecture. A multitier cloud architecture contains multiple separate parts such as an “application layer,” a “management layer,” and a “resource layer” [13]. Some works have appeared examining the energy consumption management in a multitier cloud architecture.

Usman et al. proposed a cloud architecture composed of four modules: broker, cloud manager, VM manager, and resource scheduler. By using an Interior Search Algorithm (ISA), they developed an energy-efficient VM allocation technique to overcome high energy consumption and reduce under-utilized resources in a cloud system [14]. Aiming to use the computing resources productively and energy efficiently, Beloglazov presented a three-tier cloud architecture composed of a global resource manager, user applications, and resource pools. He proposed a distributed dynamic VM consolidation approach utilizing fine-grained fluctuations in the application workloads to minimize the number of active physical nodes [15].

Zhu et al. proposed a cloud framework composed of four modules: application agent, VM allocation center, global scheduling center, and resource pools. In addition, they designed a resource allocation and scheduling strategy to reduce the energy consumption on both the system level and the component level [16]. In order to promote energy efficiency in a cloud system, Ghosh et al. developed a multitier cloud architecture composed of a resource provisioning decision layer, a VM deployment layer, and an actual service layer. Furthermore, for reducing the complexity of performance analysis, they developed a multilevel interactive stochastic submodel method to derive the performance measures of the system [17]. Obviously, it is more reasonable to study the energy consumption problem by considering a multitier cloud architecture.

2.4. Motivation for Our Research. Inspired by the work mentioned above, in this paper, we propose a dynamic energy-efficient resource management scheme in a cloud system. Considering that it is more realistic to study energy conservation under a multitier cloud architecture, we present a cloud architecture composed of a task scheduling decision layer, a resource provisioning layer, and an actual service layer. It's noted that switching all the idle servers to a low-power sleep state may deteriorate the response performance. To save energy as well as to maintain the cloud user's quality of experience, we configure PMs into two pools: a hot pool and a warm pool. The PMs in the hot pool keep working continuously to provide cloud services instantly for the arriving requests. The PMs in the warm pool are turned on, but remain in a dynamic sleep mode to reduce energy consumption.

In addition, this paper also considers the provisioning process of VMs in both of the two pools. Concretely, each PM is configured with a resource search engine (RSE) that finds an available VM for each request, and the RSE is set to sleep synchronously with all the VMs on the PM to conserve energy. To analyze the proposed scheme, we establish a hybrid queueing system composed of three stochastic submodels with synchronous multiple vacations, and we study the system performance through theoretical analysis and numerical experiments. By building a system cost function, we study the trade-off between different performance measures and present an improved SSA to optimize the sleep mechanism.

The remainder of this paper is organized as follows. In Section 3, by considering a multitier cloud architecture and two PM pools, we propose an energy-efficient resource management scheme with a synchronous sleep mechanism. In Section 4, we establish a hybrid queueing system composed of three submodels. In Section 5, we analyze the steady-state probability distribution of the queueing system by establishing a three-dimensional Markov chain. In Section 6, based on model analysis results, we evaluate the average latency of requests and the energy-saving rate of the system. In Section 7, we show the influence of the sleep mechanism on the performance measures by using numerical results. In Section 8, we present an improved intelligent algorithm to optimize the sleep mechanism. Finally, we summarize the whole paper in Section 9.

3. Scheme Description

Proper deployment of VMs is critical for the energy conservation and the Quality of Service (QoS) guarantee in a cloud system. In order to save energy and maintain the QoS, this paper proposes a dynamic energy-efficient resource management scheme, where the PMs are grouped into two pools: a hot pool and a warm pool. In the hot pool, the PMs are running continuously and the VMs hosted on a PM are always available. This means that the requests allocated to the hot pool can be served quickly so that the QoS of the cloud system can be guaranteed. In the warm pool, a synchronous sleep mechanism is introduced for the purpose of achieving a better energy-saving effect. The service provided by the warm pool can be delayed by the sleep mechanism. We call the PMs, the RSE, and the VMs in hot pool, the hot PMs, the hot RSE, and the hot VMs. And we call the PMs, the RSE, and the VMs in warm pool, the warm PMs, the warm RSE, and the warm VMs. Based on a multitier cloud architecture and a grouping approach for the PMs, we propose a novel resource management scheme shown in Figure 1.

In Figure 1, we assume that each PM is equipped with a RSE and the maximum number of VMs deployed on one PM is m. We also assume that the numbers of the identical PMs in the hot pool and the warm pool are n_h and n_w, respectively, where $n_h = 1, 2, \ldots$ and $n_w = 1, 2, \ldots$. The life cycle of a request with the resource management scheme proposed in this paper is illustrated as follows:

(1) All the requests are assumed to be homogeneous and enter a first-come, first-served (FCFS) queue in the system buffer. The request at the head of the queue firstly receives the service of the Task Scheduling Decision Engine (TSDE). As long as the hot pool is not full, the request will be allocated by the TSDE to the hot pool. Otherwise, the request will be allocated to the warm pool.

(2) The request allocated to the hot pool randomly enters the FCFS queue in one of the hot PM buffers. The request at the head of the queue is processed by a RSE, which is used to find a VM on the selected PM for resource provision. If at least one idle VM exists on one of the hot PMs, the RSE provisions an available VM to the request, and the request is immediately served by the running VM. After the service is completed, the request will depart the system.

(3) The request allocated to the warm pool randomly enters the FCFS queue in one of the warm PM buffers. The request at the head of the queue can have its service delayed due to the introduction of the sleep mechanism. On one of the warm PMs, once all the requests are processed, the RSE together with all the VMs enter a sleep period. Meanwhile, a sleep timer is started. When the sleep timer expires, if at least one request exists in the warm buffer, the RSE and all the VMs on the PM will wake up, otherwise they will enter the next sleep period.

Then, we build a hybrid queueing system to mathematically derive the system performance measures and to solve the performance optimization problem with the proposed scheme.

4. System Model

In this section, we model the proposed scheme as three submodels based on the continuous-time environment as follows. Then, we obtained the continuous-time Markov chains (CTMC) of the hot PM and the warm PM, respectively.

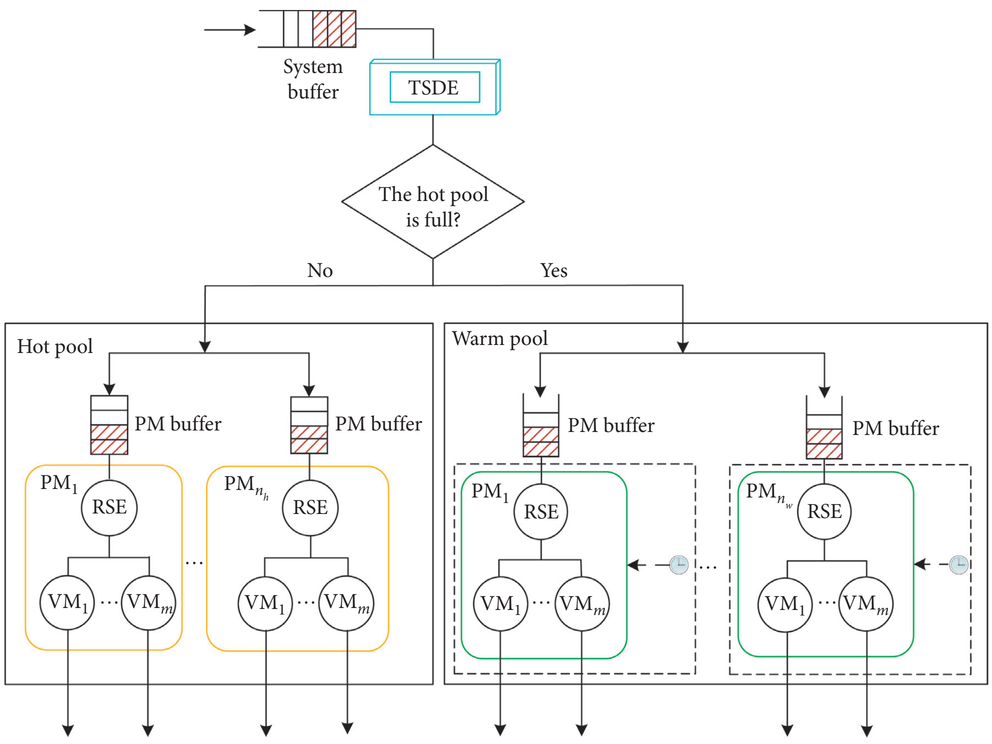

FIGURE 1: Multitier cloud architecture-based resource management scheme proposed.

4.1. TSDE Submodel. In cloud systems, some practical requests are independent with each other, while other practical requests are correlated. The computing requests initiated by users are usually uncorrelated. Therefore, the arrival process with Poisson distribution is considered to be appropriate for capturing the stochastic behavior of a cloud computing system with uncorrelated traffic [18].

In this research, we focus on user's initiated requests. Therefore, we can make the following assumptions. In the request scheduling decision process, we assume that the arrival intervals of requests and the service times of requests are independent, identically distributed (i.i.d) random variables. Request arrivals at the cloud system presented in this paper are supposed to follow a Poisson process with arrival rate $\lambda_0, \lambda_0 > 0$. The service time of a request processed by the TSDE is supposed to follow an exponential distribution with service rate $\delta, \delta > 0$.

Therefore, we build a single server queue for the task-scheduling decision process. We define the service intensity ρ_0 of the TSDE to be the number of request arrivals at the TSDE during the service time of a request. ρ_0 is given as follows:

$$\rho_0 = \frac{\lambda_0}{\delta}. \tag{1}$$

We define the latency W_{dec} of a request in the TSDE buffer to be the time duration from the instant of a request arriving at the TSDE buffer to the instant of the request departing the TSDE buffer. The average latency $E[W_{\text{dec}}]$ of requests in the TSDE buffer is obtained as follows:

$$E[W_{\text{dec}}] = \frac{\rho_0}{\delta(1-\rho_0)}. \tag{2}$$

Substituting equation (1) into equation (2), we have

$$E[W_{\text{dec}}] = \frac{\lambda_0}{\delta(\delta-\lambda_0)}. \tag{3}$$

4.2. Hot Pool Submodel. In this paper, we focus on a hot PM to build a queue model as a submodel of the system called the hot pool submodel and study the performance of the hot pool. Let $L, L < +\infty$, be the capacity of the hot PM buffer. Let random variable $N_1(t) = i, i \in \{0, 1, \ldots, L\}$, be the number of requests in the hot PM buffer at instant $t, t \geq 0$. Let random variable $J_1(t) = j, j \in \{0, 1\}$, be the state of the RSE, whether it is busy with provisioning a VM $(j = 1)$ or not $(j = 0)$. Each hot VM processes a request by loading a software environment (SE). Let random variable $S_1(t) = k, k \in \{0, 1, \ldots, m\}$, be the number of hot VMs loaded with an SE at instant t. We call $N_1(t)$ the system level, $J_1(t)$ the system stage, and $S_1(t)$ the system phase. $\{(N_1(t), J_1(t), S_1(t)), t \geq 0\}$ constitutes a three-dimensional continuous-time stochastic process with state space Ω_1 as follows:

$$\begin{aligned}\Omega_1 = &\{(0, j, k) \mid j \in \{0, 1\}, k \in \{0, 1, \ldots, m-1\}\} \\ &\cup \{(i, 0, m) \mid i \in \{0, 1, \ldots, L\}\} \\ &\cup \{(i, 1, k) \mid i \in \{1, 2, \ldots, L\}, k \in \{0, 1, \ldots, m-1\}\}.\end{aligned} \tag{4}$$

We assume that a newly arriving request is randomly allocated to one of the hot PMs. The decomposition of a

Poisson process yields multiple Poisson processes [19]. The request arrivals at each hot PM are supposed to follow a Poisson process with arrival rate λ_1. We have

$$\lambda_1 = \frac{\lambda_0}{n_h}. \tag{5}$$

We assume that the service time of a request processed by the hot RSE follows an exponential distribution with service rate $\beta_1, \beta_1 > 0$. The service time of a request processed by the hot VM loaded with SE is supposed to follow an exponential distribution with service rate $\mu_1, \mu_1 > 0$.

Based on these assumptions, the stochastic process $\{(N_1(t), J_1(t), S_1(t)), t \geq 0\}$ can be regarded as a CTMC.

We define $\pi_{i,j,k}$ as the steady-state probability distribution of the hot PM for the system level being equal to i, the system stage being equal to j, and the system phase being equal to k. $\pi_{i,j,k}$ is expressed as follows:

$$\pi_{i,j,k} = \lim_{t \longrightarrow \infty} P\{N_1(t) = i, J_1(t) = j, S_1(t) = k\}, \tag{6}$$

where $(i, j, k) \in \Omega_1$.

We define $\boldsymbol{\pi}_i$ as the steady-state probability distribution vector of the system level being equal to i. $\boldsymbol{\pi}_i$ can be given as follows:

$$\boldsymbol{\pi}_i = \begin{cases} \left(\pi_{0,0,0}, \pi_{0,0,1}, \ldots, \pi_{0,0,m-1}, \pi_{0,1,0} \ldots, \pi_{0,1,m-1}\right), & i = 0, \\ \left(\pi_{i,0,m}, \pi_{i,1,0}, \ldots, \pi_{i,1,m-1}\right), & 0 < i \leq L. \end{cases} \tag{7}$$

The steady-state probability distribution $\boldsymbol{\Pi}_1$ of the CTMC $\{(N_1(t), J_1(t), S_1(t)), t \geq 0\}$ is composed of $\boldsymbol{\pi}_i, 0 \leq i \leq L$. $\boldsymbol{\Pi}_1$ is given as follows:

$$\boldsymbol{\Pi}_1 = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_L). \tag{8}$$

4.3. Warm Pool Submodel. In order to evaluate the performance of the warm pool, we focus on a warm PM to build a queue model as another submodel of the system called the warm pool submodel. We assume that the capacity of the warm PM buffer is infinite. Let $N_2(t) = i, i \in \{0, 1, \ldots\}$, be the number of requests in the warm PM buffer at instant t. Unlike the hot PMs, a synchronous sleep mechanism is introduced to each warm PM. The RSE and all the VMs on one warm PM will go to sleep synchronously if possible. Let $J_2(t) = j, j \in \{0, 1, 2\}$, be the state of the warm RSE. $j = 0$ means the warm RSE is asleep, $j = 1$ means the warm RSE is idle, and $j = 2$ means the warm RSE is busy with provisioning a VM for a request. Just like those in the hot pool, each warm VM also needs to load an SE for processing a request. Let $S_2(t) = k, k \in \{0, 1, \ldots, m\}$, be the number of warm VMs loaded with an SE at instant t. We call $N_2(t)$ the system level, $J_2(t)$ the system stage, and $S_2(t)$ the system phase. $\{(N_2(t), J_2(t), S_2(t)), t \geq 0\}$ constitutes a three-dimensional continuous-time stochastic process with state space Ω_2 as follows:

$$\begin{aligned} \Omega_2 = &\{(0, 1, k) | k \in \{1, 2, \ldots, m-1\}\} \\ &\cup \{(i, 0, 0) | i \in \{0, 1, \ldots\}\} \\ &\cup \{(i, 1, m) | i \in \{0, 1, \ldots\}\} \\ &\cup \{(i, 2, k) | i \in \{0, 1, \ldots\}, k \in \{0, 1, \ldots, m-1\}\}. \end{aligned} \tag{9}$$

The general input flow is split into two streams, one is into the hot pool and the other is into the warm pool. In Section 4.1, the general request arrivals are assumed to follow a Poisson process, so the request arrivals at the warm pool also follow a Poisson process. We assume that a newly arriving request is randomly allocated to one of the warm PMs. The arrival rate of the requests at each warm PM is given as follows:

$$\lambda_2 = \frac{\lambda_0 (1-q)}{n_w}, \tag{10}$$

where λ_0 is the arrival rate of the requests at the TSDE submodel and q is the probability that a newly arriving request can be accepted by the hot pool. q is calculated as follows:

$$q = 1 - \left(\sum_{i=0}^{m-1} \pi_{L,1,i} + \pi_{L,0,m}\right)^{n_h}. \tag{11}$$

We assume that the service time of a request processed by the warm RSE follows an exponential distribution with service rate $\beta_2, \beta_2 > 0$. The service time of a request processed by the warm VM loaded with an SE is supposed to follow an exponential distribution with service rate $\mu_2, \mu_2 > 0$. A sleep timer is used to control the time length of a sleep period. The time length of the sleep timer is assumed to follow an exponential distribution with sleep parameter $\phi, \phi > 0$.

Based on these assumptions, the stochastic process $\{(N_2(t), J_2(t), S_2(t)), t \geq 0\}$ can be regarded as a CTMC.

We define $\pi^*_{i,j,k}$ as the steady-state probability distribution of the warm PM for the system level being equal to i, the system stage being equal to j, and the system phase being equal to k. $\pi^*_{i,j,k}$ is given by

$$\pi^*_{i,j,k} = \lim_{t \longrightarrow \infty} P\{N_2(t) = i, J_2(t) = j, S_2(t) = k\}, \tag{12}$$

where $(i, j, k) \in \Omega_2$.

We define $\boldsymbol{\pi}^*_i$ as the steady-state probability distribution vector of the warm PM for the system level being equal to i. $\boldsymbol{\pi}^*_i$ can be given as follows:

$$\boldsymbol{\pi}^*_i = \begin{cases} \left(\pi^*_{0,0,0}, \pi^*_{0,1,1}, \ldots, \pi^*_{0,1,m}, \pi^*_{0,2,0}, \ldots, \pi^*_{0,2,m-1}\right), & i = 0, \\ \left(\pi^*_{i,0,0}, \pi^*_{i,1,m}, \pi^*_{i,2,0}, \ldots, \pi^*_{i,2,m-1}\right), & i \geq 1. \end{cases} \tag{13}$$

The steady-state probability distribution $\boldsymbol{\Pi}_2$ of the CTMC $\{(N_2(t), J_2(t), S_2(t)), t \geq 0\}$ is composed of $\boldsymbol{\pi}^*_i, i \geq 0$ and given as follows:

$$\boldsymbol{\Pi}_2 = (\boldsymbol{\pi}^*_0, \boldsymbol{\pi}^*_1, \ldots). \tag{14}$$

5. Model Analysis

In this section, we construct the transition rate matrixes in the context of CTMC and derive the steady-state probability distributions of the hot PM and the warm PM, respectively.

5.1. Steady-State Probability Distributions of the Hot PM. Let $\mathbf{Q}_1$ be the one-step state transition rate matrix of the CTMC $\{(N_1(t), J_1(t), S_1(t)), t \geq 0\}$. Let $\mathbf{Q}_{u,v}$ be the one-step state transition rate submatrix of $\mathbf{Q}_1$ for the system level changing to $v, v = 0, 1, \ldots, L$, from $u, u = 0, 1, \ldots, L$. For the convenience of expression, we denote $\mathbf{Q}_{u,u-1}$ as $\mathbf{B}_u$, $\mathbf{Q}_{u,u}$ as $\mathbf{A}_u$, and $\mathbf{Q}_{u,u+1}$ as $\mathbf{C}_u$.

(1) For the case of $u = 0$, there are no requests in the hot PM buffer.

If a new request arrives at the hot PM, the system state changes in the following two cases:

(a) When the number of the hot VMs loaded with an SE is less than m and the hot RSE is idle, the newly arriving request accesses the hot RSE immediately. The system level and the system phase remain unchanged, but the system stage increases by one. The system state transfers to $(0, 1, k)$ from $(0, 0, k), 0 \leq k \leq m - 1$, with λ_1.

(b) When the number of the hot VMs loaded with an SE is less than m, but the hot RSE is busy, or the number of the hot VMs loaded with an SE is up to m, the newly arriving request has to wait in the hot PM buffer. The system level increases by one, but the system stage and the system phase remain unchanged. The system state transfers to $(1, 1, k)$ from $(0, 1, k), 0 \leq k \leq m - 1$, or to $(1, 0, m)$ from $(0, 0, m)$, with λ_1.

If a request is completely processed by the hot RSE, one of the deployed hot VMs loads the SE and processes this request. The system level remains unchanged, the system stage decreases by one, and the system phase increases by one. The system state transfers to $(0, 0, k + 1)$ from $(0, 1, k), 0 \leq k \leq m - 1$, with β_1.

If a request is completely processed by a hot VM and departs the system, the SE is removed. The system level and the system stage remain unchanged, but the system phase decreases by one. The system state transfers to $(0, 0, k - 1)$ from $(0, 0, k), 1 \leq k \leq m$, or to $(0, 1, k - 1)$ from $(0, 1, k), 1 \leq k \leq m - 1$, with $k\mu_1$.

Otherwise, the system state remains fixed at $(0, 0, k), 0 \leq k \leq m$, with $-(\lambda_1 + k\mu_1)$, or at $(0, 1, k), 0 \leq k \leq m - 1$, with $-(\lambda_1 + k\mu_1 + \beta_1)$.

In summary, $\mathbf{A}_0$ is a $(2m + 1) \times (2m + 1)$ matrix given as follows:

$$\mathbf{A}_0 = \left(\begin{array}{cccccccccc}
-\lambda_1 & & & & & \lambda_1 & & & & \\
\mu_1 & -(\lambda_1 + \mu_1) & & & & & \lambda_1 & & & \\
 & 2\mu_1 & -(\lambda_1 + 2\mu_1) & & & & & \lambda_1 & & \\
 & & \ddots & \ddots & & & & & \ddots & \\
 & & & (m-1)\mu_1 & -(\lambda_1 + (m-1)\mu_1) & & & & & \lambda_1 \\
 & & & & m\mu_1 & -(\lambda_1 + m\mu_1) & & & & \\
 & \beta_1 & & & & & -(\lambda_1 + \beta_1) & & & \\
 & & \beta_1 & & & & \mu_1 & -(\lambda_1 + \beta_1 + \mu_1) & & \\
 & & & \ddots & & & & \ddots & \ddots & \\
 & & & & \beta_1 & & & & (m-1)\mu_1 & -(\lambda_1 + \beta_1 + (m-1)\mu_1)
\end{array}\right). \quad (15)$$

$\mathbf{C}_0$ is a $(2m + 1) \times (m + 1)$ matrix given as follows:

$$\mathbf{C}_0 = \begin{pmatrix}
0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 \\
\lambda_1 & 0 & \cdots & 0 \\
0 & \lambda_1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \lambda_1
\end{pmatrix}. \quad (16)$$

(2) For the case of $1 \leq u < L$, there is at least one request in the hot PM buffer, and the hot PM buffer is not full.

If a new request arrives at the hot PM, the newly arriving request has to wait at the hot PM buffer. The system level increases by one, but the system stage and the system phase remain unchanged. The system state transfers to $(u + 1, 0, m)$ from $(u, 0, m)$ or to $(u + 1, 1, k)$ from $(u, 1, k), 0 \leq k \leq m - 1$, with λ_1.

If a request is completely processed by the hot RSE, one of the deployed hot VMs loads the SE and provides service for this request. The system state changes in the following two cases:

(a) When the number of the hot VMs loaded with an SE is less than m, the hot RSE processes the first request waiting in the hot PM buffer immediately. The system level decreases by one, the system stage remains unchanged, and the system phase increases by one. The system state transfers to $(u-1,1,k+1)$ from $(u,1,k), 0 \le k \le m-2$, with β_1.

(b) When the number of the hot VMs loaded with an SE is up to m, the hot RSE becomes idle, and the requests in the hot PM buffer keep waiting. The system level remains unchanged, the system stage decreases by one, and the system phase increases by one. The system state transfers to $(u,0,m)$ from $(u,1,m-1)$ with β_1.

If a request is completely processed by a hot VM and departs the system, the SE is removed. The system state changes in the following two cases:

(a) When the hot RSE is idle, the first request waiting in the hot PM buffer accesses the hot RSE immediately. The system level and the system phase decrease by one, and the system stage increases by one. The system level transfers to $(u-1,1,m-1)$ from $(u,0,m)$ with $m\mu_1$.

(b) When the hot RSE is busy, the requests in the hot PM buffer keep waiting. The system level and the system stage remain unchanged, but the system phase decreases by one. The system state transfers to $(u,1,k-1)$ from $(u,1,k), 1 \le k \le m-1$, with $k\mu_1$.

Otherwise, the system state remains fixed at $(u,0,m)$ with $-(\lambda_1+m\mu_1)$, or at $(u,1,k), 0 \le k \le m-1$, with $-(\lambda_1+k\mu_1+\beta_1)$.

In summary, $\mathbf{B}_1$ is an $(m+1)\times(2m+1)$ matrix given as follows:

$$\mathbf{B}_1 = \begin{pmatrix} 0 & \cdots & 0 & 0 & \cdots & m\mu_1 \\ 0 & \cdots & 0 & \beta_1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & \beta_1 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix}. \tag{17}$$

Let $\mathbf{B}$ represent $\mathbf{B}_u, u = 2,3,\ldots,L-1$. $\mathbf{B}$ is an $(m+1)\times(m+1)$ matrix given as follows:

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & \cdots & m\mu_1 \\ 0 & 0 & \beta & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \beta_1 \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}. \tag{18}$$

Let $\mathbf{A}$ represent $\mathbf{A}_u, u = 1,2,\ldots,L-1$. $\mathbf{A}$ is an $(m+1)\times(m+1)$ lower triangular matrix given by

$$\mathbf{A} = \begin{pmatrix} -(\lambda_1+m\mu_1) & & & & \\ & -(\lambda_1+\beta_1) & & & \\ & \mu_1 & -(\lambda_1+\beta_1+\mu_1) & & \\ & & \ddots & \ddots & \\ \beta_1 & & & (m-1)\mu_1 & -(\lambda_1+\beta_1+(m-1)\mu_1) \end{pmatrix}. \tag{19}$$

Let $\mathbf{C}$ represent $\mathbf{C}_u, u = 1,2,\ldots,L-1$. $\mathbf{C}$ is an $(m+1)\times(m+1)$ diagonal matrix given as follows:

$$\mathbf{C} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_1 & & \\ & & \ddots & \\ & & & \lambda_1 \end{pmatrix}. \tag{20}$$

(3) For the case of $u = L$, the hot PM buffer is full. Therefore, no new requests can join the hot PM.

If a request is completely processed by the hot RSE, one of the deployed VMs loads the SE and processes this request. The system state changes in the following two cases:

(a) When the number of the hot VMs loaded with an SE is less than m, the hot RSE processes the first request waiting in the hot PM buffer immediately. The system level decreases by one, the system stage remains unchanged, and the system phase increases by one. The system state transfers to $(L-1,1,k+1)$ from $(L,1,k), 0 \le k \le m-2$, with β_1.

(b) When the number of the hot VMs loaded with an SE is up to m, no other hot VMs can be provisioned by the hot RSE, so the hot RSE becomes idle, and all the requests in the hot PM buffer keep waiting. The system level remains unchanged, the system stage decreases by one, and the system phase increases by one. The system state transfers to $(L,0,m)$ from $(L,1,m-1)$ with β_1.

If a request is completely processed by a hot VM and departs the system, the SE is removed. The system state changes in the following two cases:

(a) When the hot RSE is idle, the first request waiting in the hot PM buffer is processed by the hot RSE immediately. The system level and the system phase decrease by one, and the system stage increases by one. The system state transfers to $(L-1,1,m-1)$ from $(L,0,m)$ with $m\mu_1$.

(b) When the hot RSE is busy, all the requests in the hot PM buffer keep waiting. The system level and the system stage remain unchanged, but the system phase decreases by one. The system state transfers to $(L,1,k-1)$ from $(L,1,k), 1 \le k \le m-1$, with $k\mu_1$.

Otherwise, the system state remains fixed at $(L,0,m)$ with $-m\mu_1$, or at $(L,1,k), 0 \le k \le m-1$, with $-(k\mu_1+\beta_1)$.

Obviously, $\mathbf{B}_L$ is an $(m+1)\times(m+1)$ matrix, and $\mathbf{B}_L = \mathbf{B}$. $\mathbf{A}_L$ is an $(m+1)\times(m+1)$ lower triangular matrix given as follows:

$$\mathbf{A}_L = \begin{pmatrix} -m\mu_1 & & & & \\ & -\beta_1 & & & \\ & \mu & -(\beta_1+\mu_1) & & \\ & & \ddots & \ddots & \\ & \beta_1 & & (m-1)\mu_1 & -(\beta_1+(m-1)\mu_1) \end{pmatrix}. \tag{21}$$

At present, we have obtained all the submatrices in the one-step state transition rate matrix $\mathbf{Q}_1$. $\mathbf{Q}_1$ can be written as follows:

$$\mathbf{Q}_1 = \begin{pmatrix} \mathbf{A}_0 & \mathbf{C}_0 & & & & \\ \mathbf{B}_1 & \mathbf{A} & \mathbf{C} & & & \\ & \mathbf{B} & \mathbf{A} & \mathbf{C} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \mathbf{B} & \mathbf{A} & \mathbf{C} \\ & & & & \mathbf{B} & \mathbf{A}_L \end{pmatrix}. \tag{22}$$

The steady-state probability distribution $\mathbf{\Pi}_1$ of the CTMC $\{(N_1(t), J_1(t), S_1(t)), t \ge 0\}$ satisfies the following equilibrium equation and normalization condition:

$$\begin{cases} \mathbf{\Pi}_1\mathbf{Q}_1 = \mathbf{0}, \\ \mathbf{\Pi}_1\mathbf{e}_1 = 1, \end{cases} \tag{23}$$

where $\mathbf{e}_1$ is an $((L+2)m+L+1)\times 1$ vector with all elements being equal to 1.

By solving equation (23), we derive the steady-state probability distribution $\mathbf{\Pi}_1$ of the CTMC $\{(N_1(t), J_1(t), S_1(t)), t \ge 0\}$, where $\mathbf{\Pi}_1 = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_L)$.

5.2. Steady-State Probability Distribution of the Warm PM.

Let $\mathbf{Q}_2$ be the one-step state transition rate matrix of the CTMC $\{(N_2(t), J_2(t), S_2(t)), t \ge 0\}$. Let $\mathbf{Q}^*_{u,v}$ be the one-step state transition rate submatrix of $\mathbf{Q}_2$ for the system level changing to $v, v = 0, 1, \ldots$, from $u, u = 0, 1, \ldots$. We denote $\mathbf{Q}^*_{u,u-1}$ as $\mathbf{B}^*_u$, $\mathbf{Q}^*_{u,u}$ as $\mathbf{A}^*_u$, and $\mathbf{Q}^*_{u,u+1}$ as $\mathbf{C}^*_u$.

(1) For the case of $u = 0$, there are no requests in the warm PM buffer.

If a new request arrives at the warm PM, the system state changes in the following three cases:

(a) When the warm RSE and the warm VMs are asleep, the newly arriving request has to wait in the warm PM buffer until the sleep timer expires. The system level increases by one, but the system stage and the system phase remain unchanged. The system state transfers to $(1,0,0)$ from $(0,0,0)$ with λ_2.

(b) When the warm RSE and the warm VMs are awake, and the number of the warm VMs loaded with an SE is up to m; no other warm VMs can be provisioned by the hot RSE. The newly arriving request has to wait in the warm PM buffer. The system level increases by one, but the system stage and the system phase remain unchanged. The system state transfers to $(1,1,m)$ from $(0,1,m)$ with λ_2.

(c) When the warm RSE and the warm VMs are awake, and the number of the warm VMs loaded with an SE is less than m, at least one VM can be provisioned. If the warm RSE is busy, the newly arriving request has to wait in the warm PM buffer. The system level increases by one, but the system stage and the system phase remain unchanged. The system state transfers to $(1,2,k)$ from $(0,2,k), 0 \le k \le m-1$, with λ_2. If the warm RSE is idle, the newly arriving request accesses the warm RSE immediately. The system level and the system phase remain unchanged, but the system stage increases by one. The system state transfers to $(0,2,k)$ from $(0,1,k), 1 \le k \le m-1$, with λ_2.

If a request is completely processed by the warm RSE, one of the deployed warm VMs loads the SE and processes this request. The system level remains unchanged, the system stage decreases by one, and the system phase increases by one. The system state transfers to $(0,1,k+1)$ from $(0,2,k), 0 \le k \le m-1$, with β_2.

If a request is completely processed by a warm VM and departs the system, the system state changes in the following two cases:

(a) When the warm RSE is idle and there is only one warm VM loaded with an SE, the used SE is removed. The warm RSE and the warm VMs enter a sleep period immediately. The system level remains unchanged, but the system stage and the system phase decrease by one. The system state transfers to $(0,0,0)$ from $(0,1,1)$ with μ_2.

(b) When the warm RSE is idle and there are at least two warm VMs loaded with an SE or the warm RSE is busy and there is at least one warm VM loaded with an SE, the used SE is removed. The system level and the system stage remain unchanged, but the system phase decreases by one. The system state transfers to $(0,1,k-1)$ from $(0,1,k), 2 \le k \le m$, or to $(0,2,k-1)$ from $(0,2,k), 1 \le k \le m-1$, with $k\mu_2$.

Otherwise, the system state remains fixed at $(0,0,0)$ with $-\lambda_2$, at $(0,1,k), 1 \le k \le m$, with $-(\lambda_2 + k\mu_2)$, or at $(0,2,k), 0 \le k \le m-1$, with $-(\lambda_2 + k\mu_2 + \beta_2)$.

In summary, $\mathbf{A}_0^*$ is a $(2m+1) \times (2m+1)$ matrix given as follows:

$$\mathbf{A}_0^* = \begin{pmatrix}
-\lambda_2 & & & & & & & & & \\
\mu_2 & -(\lambda_2+\mu_2) & & & & & \lambda_2 & & & \\
 & 2\mu_2 & -(\lambda_2+2\mu_2) & & & & & \lambda_2 & & \\
 & & \ddots & \ddots & & & & & \ddots & \\
 & & & (m-1)\mu_2 & -(\lambda_2+(m-1)\mu_2) & & & & & \lambda_2 \\
 & & & & m\mu_2 & -(\lambda_2+m\mu_2) & & & & \\
 & \beta_2 & & & & & -(\lambda_2+\beta_2) & & & \\
 & & \beta_2 & & & & \mu_2 & -(\lambda_2+\beta_2+\mu_2) & & \\
 & & & \ddots & & & & \ddots & \ddots & \\
 & & & & \beta_2 & & & & (m-1)\mu_2 & -(\lambda_2+\beta_2+(m-1)\mu_2)
\end{pmatrix}. \tag{24}$$

$\mathbf{C}_0^*$ is a $(2m+1) \times (m+2)$ matrix given as follows:

$$\mathbf{C}_0^* = \begin{pmatrix}
\lambda_2 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 \\
0 & \lambda_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \lambda_2
\end{pmatrix}. \tag{25}$$

(2) For the case of $1 \le u < \infty$, there is at least one request in the warm PM buffer.

If there are no new request arrivals on the warm PM, while the warm RSE and the warm VMs are asleep, once the sleep timer expires, the warm RSE wakes up and processes the first request waiting in the warm PM buffer immediately. The system level decreases by one, the system stage increases by two, and the system phase remains unchanged. The system state transfers to $(u-1,2,0)$ from $(u,0,0)$ with ϕ.

If a new request arrives at the warm PM, the newly arriving request has to wait in the warm buffer. The system level increases by one, but the system stage and the system phase remain unchanged. The system state transfers to $(u+1,0,0)$ from $(u,0,0)$, to $(u+1,1,m)$ from $(u,1,m)$, or to $(u+1,2,k)$ from $(u,2,k), 0 \le k \le m-1$, with λ_2.

If a request is completely processed by the warm RSE, one of the deployed warm VMs loads the SE and provides service for this request. The system state changes in the following two cases:

(a) When the number of the warm VMs loaded with an SE is less than m, the warm RSE processes the first request waiting in the warm PM buffer immediately. The system level decreases by one, the system stage remains unchanged, and the system phase increases by one. The system state transfers to $(u-1,2,k+1)$ from $(u,2,k), 0 \le k \le m-2$, with β_2.

(b) When the number of the warm VMs loaded with an SE is up to m, no other warm VMs can be provisioned by the warm RSE. Therefore, the warm RSE becomes idle and none of the requests waiting in the warm PM buffer can access the warm RSE. The system level remains unchanged, the system stage decreases by one, and the system phase increases by one. The system state transfers to $(u,1,m)$ from $(u,2,m-1)$ with β_2.

If a request is completely processed by a warm VM and departs the system, the used SE is removed. The system state changes in the following two cases:

(a) When the warm RSE is idle, the first request waiting in the warm PM buffer accesses the warm RSE immediately. The system level and the system phase decrease by one, but the system stage increases by one. The system state transfers to $(u-1,2,m-1)$ from $(u,1,m)$ with $m\mu_2$.

(b) When the warm RSE is busy, none of the requests waiting in the warm PM buffer can access the warm RSE. The system level and the system stage remain unchanged, but the system phase decreases by one. The system state transfers to $(u,2,k-1)$ from $(u,2,k), 1 \le k \le m-1$, with $k\mu_2$.

Otherwise, the system state remains fixed at $(u,0,0)$ with $-(\lambda_2 + \phi)$, at $(u,1,m)$ with $-(\lambda_2 + m\mu_2)$, or at $(u,2,k), 0 \le k \le m-1$, with $-(\lambda_2 + k\mu_2 + \beta_2)$.

In summary, $\mathbf{B}_1^*$ is an $(m+2) \times (2m+1)$ matrix given as follows:

Table 1: Iteration algorithm to compute the rate matrix $\mathbf{R}$.

Step 1. Initialize the upper bound of error ε (for example, $\varepsilon = e^{-10}$). Initialize $m, L, \lambda_0, \mu_1, \mu_2, \beta_1, \beta_2$ and ϕ as needed. Initialize the matrix $\mathbf{R} = \mathbf{0}$ with the order of $(m+2) \times (m+2)$.

Step 2. Construct the matrices $\mathbf{B}^*$, $\mathbf{A}^*$, and $\mathbf{C}^*$ by

$$\mathbf{B}^* = \begin{pmatrix} 0 & 0 & \phi & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & m\mu_2 \\ 0 & 0 & 0 & \beta_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \beta_2 \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix},$$

$$\mathbf{A}^* = \begin{pmatrix} -(\lambda_2+\phi) & & & & & \\ & -(\lambda_2+m\mu_2) & & & & \\ & & -(\lambda_2+\beta_2) & & & \\ & & \mu_2 & -(\lambda_2+\beta_2+\mu_2) & & \\ & & & \ddots & \ddots & \\ & \beta_2 & & & (m-1)\mu_2 & -(\lambda_2+\beta_2+(m-1)\mu_2) \end{pmatrix},$$

$$\mathbf{C}^* = \begin{pmatrix} \lambda_2 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_2 \end{pmatrix}.$$

Step 3. Calculate $\mathbf{W}, \mathbf{V}$, and $\mathbf{R}_1$ by

$\mathbf{W} = \mathbf{B}^* (\mathbf{A}^*)^{-1}$,

$\mathbf{V} = \mathbf{C}^* (\mathbf{A}^*)^{-1}$,

$\mathbf{R}_1 = -\mathbf{R}^2\mathbf{W} - \mathbf{V}$.

Step 4. While $\{\|\mathbf{R} - \mathbf{R}_1\|_\infty > \varepsilon\}$,

% $\|\mathbf{R} - \mathbf{R}_1\|_\infty = \max\left\{\sum_{i=1}^{(m+2)} \sum_{j=1}^{(m+2)} |r_{i,j} - r_{i,j}^*|\right\}$, where $r_{i,j}$ and $r_{i,j}^*$ are elements in $\mathbf{R}$ and $\mathbf{R}_1$, Respectively.%

$\mathbf{R} = \mathbf{R}_1$.

$\mathbf{R}_1 = -\mathbf{R}^2\mathbf{W} - \mathbf{V}$.

Endwhile

$\mathbf{R} = \mathbf{R}_1$.

Step 5. Output $\mathbf{R}$.

$$\mathbf{B}_1^* = \begin{pmatrix} 0 & \cdots & \phi & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & m\mu_2 \\ 0 & \cdots & 0 & \beta_2 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & \beta_2 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix}. \quad (26)$$

Let $\mathbf{B}^*$ represent $\mathbf{B}_u^* (u = 2, 3, \ldots)$. $\mathbf{B}^*$ is an $(m+2) \times (m+2)$ upper triangular matrix given as follows:

$$\mathbf{B}^* = \begin{pmatrix} 0 & 0 & \phi & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & m\mu_2 \\ 0 & 0 & 0 & \beta_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \beta_2 \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}. \quad (27)$$

Let $\mathbf{A}^*$ represent $\mathbf{A}_u^* (u = 1, 2, \ldots)$. $\mathbf{A}^*$ is an $(m+2) \times (m+2)$ lower triangular matrix given by

$$\mathbf{A}^* = \begin{pmatrix} -(\lambda_2+\phi) & & & & & & \\ & -(\lambda_2+m\mu_2) & & & & & \\ & & -(\lambda_2+\beta_2) & & & & \\ & & \mu_2 & -(\lambda_2+\beta_2+\mu_2) & & & \\ & & & \ddots & \ddots & & \\ & \beta_2 & & & (m-1)\mu_2 & -(\lambda_2+\beta_2+(m-1)\mu_2) \end{pmatrix}. \quad (28)$$

Let $\mathbf{C}^*$ represent $\mathbf{C}_u^*\,(u = 1, 2, \ldots)$. $\mathbf{C}^*$ is an $(m+2)\times(m+2)$ diagonal matrix given as follows:

$$\mathbf{C}^* = \begin{pmatrix} \lambda_2 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_2 \end{pmatrix}. \quad (29)$$

At present, we have obtained all the submatrices in the one step state transition rate matrix $\mathbf{Q}_2$. $\mathbf{Q}_2$ can be written as follows:

$$\mathbf{Q}_2 = \begin{pmatrix} \mathbf{A}_0^* & \mathbf{C}_0^* & & & \\ \mathbf{B}_1^* & \mathbf{A}^* & \mathbf{C}^* & & \\ & \mathbf{B}^* & \mathbf{A}^* & \mathbf{C}^* & \\ & & \mathbf{B}^* & \mathbf{A}^* & \mathbf{C}^* \\ & & & \ddots & \ddots & \ddots \end{pmatrix}. \quad (30)$$

Based on the structure of the one-step state transition rate matrix $\mathbf{Q}_2$, the three-dimensional CTMC $\{(N_2(t), J_2(t), S_2(t)), t \geq 0\}$ of the warm PM can be regarded as a type of Quasi Birth-and-Death (QBD) process. Thus, we can apply the method of a matrix-geometric solution [20, 21] to derive the steady-state probability distribution $\mathbf{\Pi}_2$ of the CTMC $\{(N_2(t), J_2(t), S_2(t)), t \geq 0\}$, where $\mathbf{\Pi}_2 = (\boldsymbol{\pi}_0^*, \boldsymbol{\pi}_1^*, \ldots)$.

First, we set up a matrix quadratic equation as follows:

$$\mathbf{R}^2\mathbf{B}^* + \mathbf{R}\mathbf{A}^* + \mathbf{C}^* = \mathbf{0}. \quad (31)$$

Since $\mathbf{A}^*$ must be nonsingular, from equation (31), we have

$$\mathbf{R}^2\mathbf{B}^*(\mathbf{A}^*)^{-1} + \mathbf{R} + \mathbf{C}^*(\mathbf{A}^*)^{-1} = \mathbf{0}. \quad (32)$$

By deducing equation (32), we obtain

$$\mathbf{R} = -\mathbf{R}^2\mathbf{W} - \mathbf{V}, \quad (33)$$

where $\mathbf{W} = \mathbf{B}^*(\mathbf{A}^*)^{-1}$ and $\mathbf{V} = \mathbf{C}^*(\mathbf{A}^*)^{-1}$.

In order to compute the rate matrix $\mathbf{R}$, we present an iteration algorithm in Table 1.

Using the rate matrix $\mathbf{R}$ obtained in Table 1, we further construct a square matrix as follows:

$$B[\mathbf{R}] = \begin{pmatrix} \mathbf{A}_0^* & C_0^* \\ \mathbf{B}_1^* & \mathbf{R}\mathbf{B}^* + \mathbf{A}^* \end{pmatrix}. \quad (34)$$

The steady-state probability distribution vectors $\boldsymbol{\pi}_0^*$ and $\boldsymbol{\pi}_1^*$ satisfy the following equation:

$$\begin{cases} (\boldsymbol{\pi}_0^*, \boldsymbol{\pi}_1^*)B[\mathbf{R}] = 0, \\ \boldsymbol{\pi}_0^*\mathbf{e}_0^* + \boldsymbol{\pi}_1^*(\mathbf{I} - \mathbf{R})^{-1}\mathbf{e}_1^* = 1, \end{cases} \quad (35)$$

where $\mathbf{e}_0^*$ is a $(2m+1)\times 1$ vector and $\mathbf{e}_1^*$ is an $(m+2)\times 1$ vector, respectively, with all elements being equal to 1.

By using the Gauss–Seidel method, we solve equation (35) to obtain $\boldsymbol{\pi}_0^*$ and $\boldsymbol{\pi}_1^*$. Other steady-state probability distribution vectors $\boldsymbol{\pi}_i^*, i = 2, 3, \ldots$, satisfy the matrix-geometric solution form as follows:

$$\boldsymbol{\pi}_i^* = \boldsymbol{\pi}_1^*\mathbf{R}^{i-1}, \quad i \geq 2. \quad (36)$$

Up to this point, we can mathematically give the steady-state probability distribution $\mathbf{\Pi}_2 = (\boldsymbol{\pi}_0^*, \boldsymbol{\pi}_1^*, \ldots)$ of the CTMC $\{(N_2(t), J_2(t), S_2(t)), t \geq 0\}$.

6. Performance Measures

In this section, we present two performance measures of the cloud system: the average latency of requests and the energy saving rate of the system.

The service intensity ρ_2 of the warm PM is given as follows:

$$\rho_2 = \lambda_2\left(\frac{1}{\mu_2} + \frac{1}{\beta_2}\right), \quad (37)$$

where λ_2 is the arrival rate of requests at a warm PM, μ_2 is the service rate of a request on a warm VM, and β_2 is the service rate of a request on the warm RSE.

For the proposed scheme, the service intensity ρ of the system is given as follows:

$$\rho = \max\{\rho_0, \rho_2\}, \quad (38)$$

where ρ_0 is the service intensity of the TSDE.

The necessary and sufficient condition for the system to be stable is $\rho < 1$. We evaluate the average latency of requests and the energy-saving rate of the system under the condition that the service intensity $\rho < 1$.

We define the latency of a request as the time duration from the instant a request arrives at the cloud system to the instant the request is about to receive the service. In this paper, the average latency of requests in the cloud system includes the average latency of requests queueing in the TSDE buffer and the average latency of requests queueing in the hot PM buffer or the warm PM buffer.

In Section 4.1, the average latency $E[W_{\text{dec}}]$ of requests queueing in the TSDE buffer has already been obtained. Next, we need to compute the average latency $E[W_{\text{vm}}]$ of requests queueing in the hot PM buffer or the warm PM buffer.

Using the steady-state probability distribution $\mathbf{\Pi}_1$ of the CTMC $\{(N_1(t), J_1(t), S_1(t)), t \geq 0\}$ given in Section 5.1, the average number $E[N_{\text{hot}}]$ of requests queueing in the hot PM buffer can be given by

$$E[N_{\text{hot}}] = \sum_{i=1}^{L} i \left(\sum_{k=0}^{m-1} \pi_{i,1,k} + \pi_{i,0,m} \right). \tag{39}$$

For the convenience of technique, we tag one of the hot PMs. Based on Little's law, the average latency $E[W_{\text{hot}}]$ of requests queueing in the buffer of the tagged hot PM is obtained as follows:

$$E[W_{\text{hot}}] = \frac{1}{\lambda_1 \left(1 - \sum_{i=0}^{m-1} \pi_{L,1,i} - \pi_{L,0,m}\right)} E[N_{\text{hot}}], \tag{40}$$

where λ_1 is the arrival rate of requests at the tagged hot PM.

We also tag one of the warm PMs. Using the steady-state probability distribution $\mathbf{\Pi}_2$ of the CTMC $\{(N_2(t), J_2(t), S_2(t)), t \geq 0\}$ given in Section 5.2, the average number $E[N_{\text{warm}}]$ of requests queueing in the buffer of the tagged warm PM is given as follows:

$$E[N_{\text{warm}}] = \sum_{i=1}^{l_1} i \left(\pi^*_{i,0,0} + \pi^*_{i,1,m} + \sum_{k=0}^{m-1} \pi^*_{i,2,k} \right). \tag{41}$$

l_1 shown in equation (41) is a sufficiently large number satisfying the following equation:

$$1 - \sum_{i=1}^{l_1} i \left(\pi^*_{i,0,0} + \pi^*_{i,1,m} + \sum_{k=0}^{m-1} \pi^*_{i,2,k} \right) < \varepsilon_1, \tag{42}$$

where ε_1, called a precision factor of the average number of requests in the warm PM buffer, is a number related to the precision of the average number of requests in the warm PM buffer. The smaller the value of ε_1 is, the more precisely the average number of requests in the warm PM buffer will be given.

Accordingly, the average latency $E[W_{\text{warm}}]$ of requests queueing in the tagged warm PM buffer is obtained as follows:

$$E[W_{\text{warm}}] = \frac{1}{\lambda_2} E[N_{\text{warm}}], \tag{43}$$

where λ_2 is the arrival rate of requests at the tagged warm PM.

Combining equations (40) and (43), the average latency $E[W_{\text{vm}}]$ of requests queueing in the hot PM buffer or the warm PM buffer can be obtained as follows:

$$E[W_{\text{vm}}] = qE[W_{\text{hot}}] + (1-q)E[W_{\text{warm}}], \tag{44}$$

where q is the probability that a newly arriving request can be accepted by the hot pool.

In summary, the average latency $E[W]$ of requests queueing in the cloud system can be derived by

$$E[W] = E[W_{\text{vm}}] + E[W_{\text{dec}}]. \tag{45}$$

Since the TSDE and the hot PMs are always running, the energy consumption there is normally constant. The energy-saving rate of the system is therefore measured as the energy conservation per unit time in the warm pool.

When the warm PMs are in the active state, the energy is consumed normally just like in the TSDE and the hot PMs. Let $w, w > 0$, be the energy consumption per second for the warm pool in the active state. Let $w_1, w_1 > 0$, be the energy consumption per second for the warm pool in the sleep state. When the warm PMs are in the sleep state, less energy will be consumed. It is obvious that $w > w_1$.

For a sleeping warm PM, if a sleep period is about to expire, the RSE and the VMs need to monitor the PM buffer. Therefore, additional energy will be consumed. Let $w_2, w_2 > 0$, be the energy consumption for each monitoring. It is noted that additional energy is also consumed when the warm PM wakes up from a sleep state. Let $w_3, w_3 > 0$, be the energy consumption for each wake up.

Therefore, in this paper, energy-saving rate S of the system is given as follows:

$$S = \sum_{i=0}^{l_2} \pi^*_{i,0,0} (w - w_1 - \phi w_2) - \sum_{i=1}^{l_2} \pi^*_{i,0,0} \phi w_3, \tag{46}$$

where ϕ is the sleep parameter of the proposed dynamic sleep mechanism defined in Section 4.3.

l_2, shown in equation (46), is a sufficiently large number, which satisfies the following equation:

$$1 - \sum_{i=1}^{l_2} \pi^*_{i,0,0} < \varepsilon_2, \tag{47}$$

where ε_2, called a precision factor of the energy-saving rate of the system, is a number related to the precision of the energy-saving rate of the system. The smaller the value of ε_2 is, the more precisely the energy saving rate of the system will be given.

7. Numerical Results

To numerically analyze the average latency $E[W]$ of requests and the energy-saving rate S of the system with the proposed scheme, we carry out experiments to provide numerical results based on MATLAB. All the experiments are carried out on a PC configured with Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz, 8.00 GB RAM, and 500G disk. The parameters set in the experiments are listed in Table 2.

7.1. Numerical Results for the Average Latency of Requests.

Figure 2 illustrates the change trend for the average latency $E[W]$ of requests with the sleep parameter ϕ for a different number n_h of the hot PMs and a different number n_w of the warm PMs.

Table 2: Parameter settings in the experiments.

Parameters	Values
The maximum number m of VMs deployed on one PM	$m = 3$
Arrival rate λ_0 of requests at the TSDE	$\lambda_0 = 4.8\,\mathrm{s}^{-1}$
Service rate δ of a request on the TSDE	$\delta = 5\,\mathrm{s}^{-1}$
Service rate β_1 of a request on the hot RSE	$\beta_1 = 1\,\mathrm{s}^{-1}$
Energy consumption ω per second for the warm pool in the active state	$\omega = 2\,\mathrm{mJ}$
Energy consumption ω_1 per second for the warm pool in the sleep state	$\omega_1 = 0.3\,\mathrm{mJ}$
Additional energy consumption ω_2 for each monitoring	$\omega_2 = 0.1\,\mathrm{mJ}$
Additional energy consumption ω_3 for each wake up	$\omega_3 = 0.2\,\mathrm{mJ}$
Precision factor ε_1 of the average number of requests in the warm PM buffer	$\varepsilon_1 = 10^{-15}$
Precision factor ε_2 of the energy-saving rate of the system	$\varepsilon_2 = 10^{-15}$

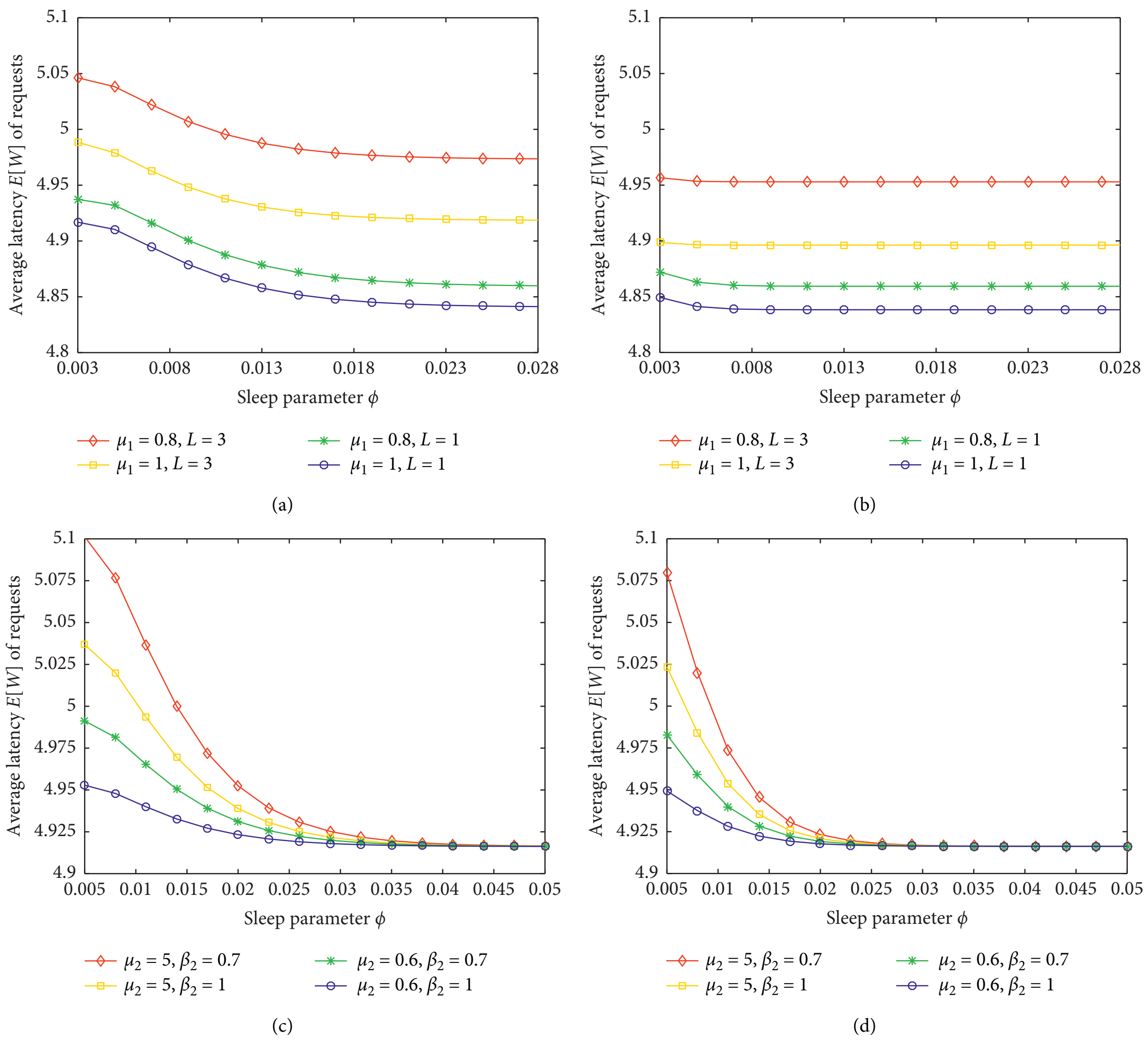

Figure 2: Average latency $E[W]$ of requests vs. sleep parameter ϕ. (a) $n_h = 2, n_w = 6$. (b) $n_h = 3, n_w = 6$. (c) $n_h = 2, n_w = 4$. (d) $n_h = 2, n_w = 6$.

In Figures 2(a) and 2(b), we show the average latency $E[W]$ of requests for the different service rates μ_1 of a request on a hot VM and the different capacities L of a hot PM buffer, respectively. In Figures 2(c) and 2(d), we show the average latency $E[W]$ of requests for the different service rates μ_2 of a request on a warm VM and the different service rates β_2 of a request on a warm RSE, respectively.

From Figure 2, we notice that, as the sleep parameter ϕ increases, the average latency $E[W]$ of requests firstly decreases accordingly and then tends to be fixed.

In the stage of the smaller sleep parameter ϕ, a newly arriving request has to wait for a longer time in the buffer of a sleeping warm PM. As the sleep parameter grows, the waiting time of a request in the warm PM buffer gets shorter. Therefore, the average latency $E[W]$ of requests shows a downtrend. This implies that the influence of the sleep mechanism on the response performance of the system is greater in the case of a smaller sleep parameter.

When the sleep parameter ϕ gets larger and grows to a certain value, the time length of a sleep period is close to zero. Therefore, a warm PM has little chance to go to sleep. As a result, the average latency $E[W]$ of requests tends to be fixed as the sleep parameter increases. This implies that the proposed sleep mechanism has little effect on the response performance of the system when the sleep parameter is large enough.

For the same sleep parameters ϕ in both Figures 2(a) and 2(b), we notice that the average latency $E[W]$ of requests goes up as the capacity L of a hot PM buffer increases. The larger a hot PM's buffer capacity is, the longer the requests wait in the hot PM buffer. This gives rise to an increase in the average latency of requests. We also notice that, as the service rate μ_1 of a request on a hot VM increases, the average latency of requests gets reduced. The higher the service rate is, the less time a request occupies the hot VM. Therefore, the average latency of requests shows a downtrend.

Comparing Figures 2(a) with 2(b), we find that, for the same capacity L of a hot PM buffer, the same service rate μ_1 of a request on a hot VM, and the same sleep parameter ϕ, as the number n_h of the hot PMs increases, the average latency $E[W]$ of requests becomes lower. The more the PMs are deployed in the hot pool, the earlier the requests arrive at the hot pool receive service. Therefore, the average latency of requests shows a downtrend. In addition, we also find that when the sleep parameter is smaller, the downtrend for the average latency of requests gets slighter as the number of the hot PMs increases. This implies that the more the PMs are deployed in the hot pool, the weaker the influence of the sleep mechanism on the response performance of the system becomes.

For the same sleep parameters ϕ in both Figures 2(c) and 2(d), we observe that the average latency $E[W]$ of requests rises up as the service rate μ_2 of a request on a warm VM increases. When the service rate of a request on a warm VM is higher, the probability of the warm RSE and the warm VMs being idle is greater. Therefore, the warm PM is more likely to be asleep, which causes the request to wait longer in the warm PM buffer. Accordingly, the average latency of requests gets larger. We also observe that, as the service rate β_2 of a request on a warm RSE increases, the average latency of requests is reduced. The higher the service rate of a request on a warm RSE is, the less time a request occupies the warm RSE. This leads to a lower average latency of requests.

Comparing Figures 2(c) with 2(d), we find that, for the same service rate μ_2 of a request on a warm VM, the same service rate β_2 of a request on a warm RSE, and the same sleep parameter ϕ, a greater number n_w of the warm PMs gives rise to a lower average latency $E[W]$ of requests. The more the PMs are deployed in the warm pool, the earlier the requests arrive at the warm pool receive service. Therefore, the average latency of requests gets reduced. In addition, we also find that the downtrend for the average latency of requests becomes sharper as the number of the warm PMs increases in the case of a smaller sleep parameter. This implies that the more the PMs are deployed in the warm pool, the stronger the influence of the sleep mechanism on the response performance of the system becomes.

7.2. Numerical Results for the Energy-Saving Rate of the System. Figure 3 shows the trends for the energy-saving rate S of the system with the sleep parameter ϕ for a different number n_h of the hot PMs and a different number n_w of the warm PMs.

In Figures 3(a) and 3(b), we show the energy-saving rate S of the system for the different service rates μ_1 of a request on a hot VM and the different capacities L of a hot PM buffer, respectively. In Figures 3(c) and 3(d), we show the energy-saving rate S of the system for the different service rates μ_2 of a request on a warm VM and the different service rates β_2 of a request on a warm RSE, respectively.

From Figure 3, we notice that, as the sleep parameter ϕ increases, the energy-saving rate S of the system shows a downward trend. In the stage of a smaller sleep parameter, the energy-saving rate of the system is initially higher. The smaller the sleep parameter is, the longer the time length of a sleep period is. For this case, frequent listening and waking up of the warm RSE and the warm VMs are avoided so that additional energy use is reduced.

As the sleep parameter ϕ gets larger, the energy-saving rate S of the system decreases. The larger the sleep parameter is, the shorter the time length of a sleep period is. For this case, the warm RSE and the warm VMs listen to the buffer and wake up from sleep frequently. This causes additional energy consumption.

For the same sleep parameters ϕ in both Figures 3(a) and 3(b), we notice that the energy-saving rate S of the system goes up as the capacity L of a hot PM buffer or the service rate μ_1 of a request on a hot VM increases. The larger the capacity of a hot PM buffer is, the more requests the hot PM can accept. The higher the service rate of a request on a hot VM is, the less time a request occupies the hot VM. Therefore, the processing capability of a hot PM becomes stronger. For this case, fewer requests are allocated to the warm pool so that the warm PMs are more likely to be in the sleep state. Accordingly, the energy-saving rate of the system is greater.

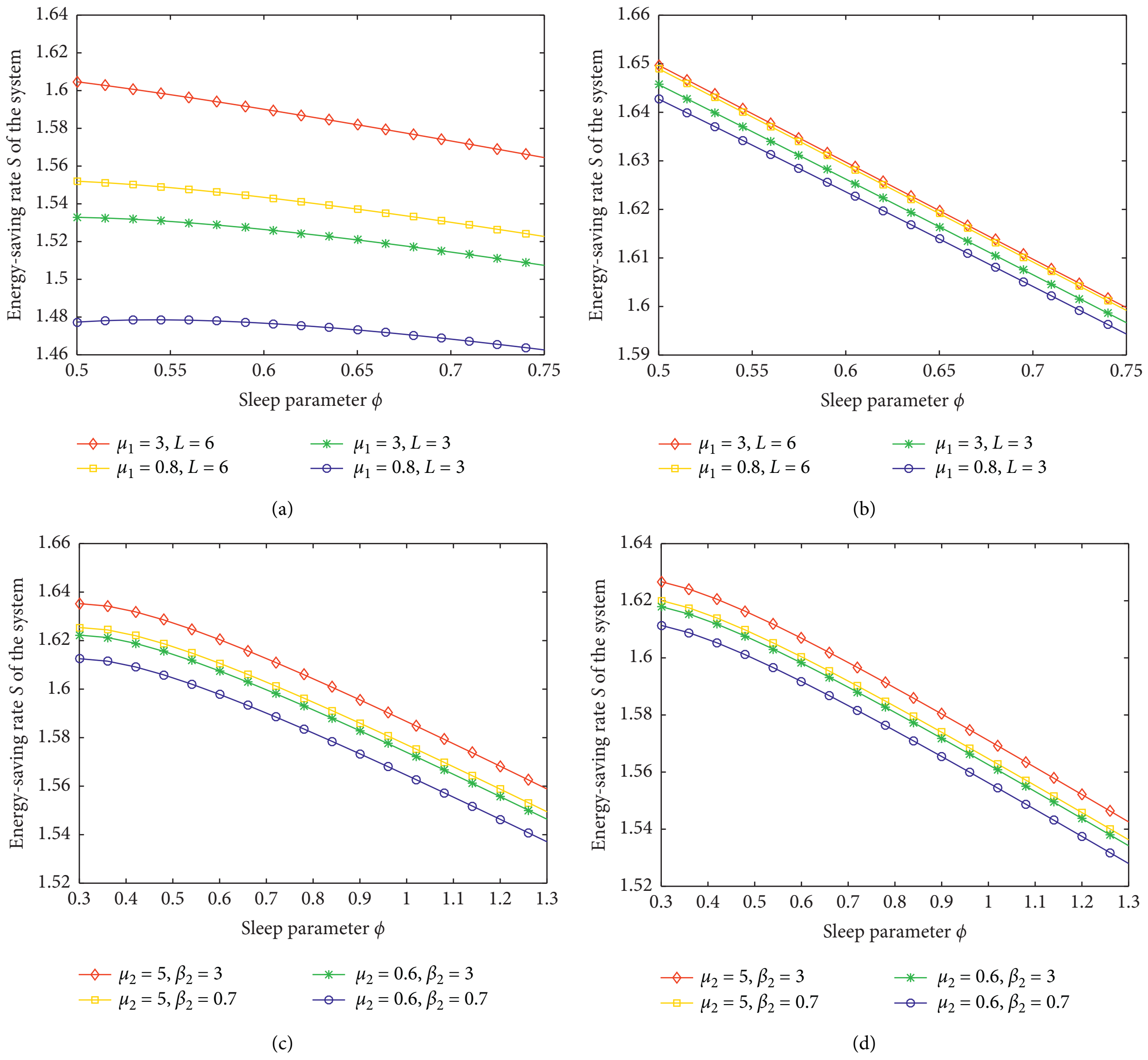

Figure 3: Energy-saving rate S of the system vs. sleep parameter ϕ. (a) $n_h = 4, n_w = 2$. (b) $n_h = 5, n_w = 2$. (c) $n_h = 4, n_w = 2$. (d) $n_h = 4, n_w = 3$.

Comparing Figures 3(a) with 3(b), we find that, for the same capacity L of a hot PM buffer, the same service rate μ_1 of a request on a hot VM, and the same sleep parameter ϕ, a larger number n_h of the hot PMs leads to a higher energy-saving rate S of the system. The more the PMs are deployed in the hot pool, the stronger the processing capability of the hot pool is. For this case, fewer requests are allocated to the warm pool so that the warm PM is more likely to be in the sleep state. This causes an increase in the energy-saving rate of the system. In addition, we also find that the more the PMs are deployed in the hot pool, the closer the energy-saving rates of the system with different capacities of a hot PM buffer and different service rates of a request on a hot VM are. This implies that the capacity of the hot PM buffer and the service rate of a request on a hot VM have less influence on the energy-saving rate of the system as the number of the hot PMs rises.

For the same sleep parameters ϕ in both Figures 3(c) and 3(d), we observe that the energy-saving rate S of the system rises as the service rate μ_2 of a request on a warm VM or service rate β_2 of a request on a warm RSE grows. The higher the service rate of a request on a warm RSE is, the less time a request occupies the warm RSE. The higher the service rate of a request on a warm VM is, the less time a request occupies the warm VM. For this case, the warm PM is more likely to become idle and enter a sleep period. Therefore, the energy-saving rate of the system shows a growth trend.

Comparing Figures 3(c) with 3(d), we find that, for the same service rate μ_2 of a request on a warm VM, the same service rate β_2 of a request on a warm RSE, and the same sleep parameter ϕ, a greater number n_w of the warm PMs leads to a higher energy-saving rate S of the system. The more the PMs are deployed in the warm pool, the stronger the processing capability of the warm pool is. For this case,

Table 3: Improved Salp Swarm Algorithm proposed to obtain the optimal sleep parameters.

```
Step 1. Initialize the number $N$ of salps, maximum iteration $T_{\max}$ for each salp's position, initial inertia weight $w_s$, inertia weight $w_m$ at the maximum iteration, upper search boundary $u_b$, and lower Search boundary $l_b$.
Step 2. Initialize the position $\phi_i (i = 1, 2, \ldots, N)$ for each salp by using a chaotic equation:
        $\phi_1$ = rand.
    % rand represents random numbers that obey uniform distribution between $(0, 1)$.%
        for $i = 2$: $N$
      $\phi_i = \xi\phi_{i-1}(1 - \phi_{i-1})$
    % $\xi$ is a given real parameter.%
    endfor
    for $i = 1$: $N$
      $\phi_i = l_b + \phi_i(u_b - l_b)$
    endfor
Step 3. Calculate the fitness $F_i (i = 1, 2, \ldots, N)$ for each salp:
    $F_i = F(\phi_i) = f_1 E[W] - f_2 S$.
Step 4. Select the best position $\phi^*$ among all the salps as the source food and calculate the fitness $F^*$ of the source food:
    $\phi^* = \operatorname{argmin}_{i\in\{1,\ldots,N\}} F_i$,
    $F^* = F(\phi^*) = f_1 E[W] - f_2 S$.
Step 5. Set the initial number of iterations as $t = 1$.
Step 6. Update the coefficient $c_1$ and inertia weight $w(t)$ with a nonlinear decreasing function:
    $c_1 = 2\exp(-4t/T_{\max})$,
    $w(t) = (w_s - w_m)((T_{\max} + t)/t) + w_s$.
Step 7. Update the position $\phi_i$ and calculate the fitness $F_i (i = 1, 2, \ldots, N - 1)$ for other salps.
    for $i = 1$: $N - 1$
      if $i \le \lfloor (N - 1)/2 \rfloor$
        $c_2$ = rand, $c_3$ = rand
        $\phi_i = \begin{cases} F + c_1((u_b - l_b)c_2 + l_b), & c_3 \ge 0 \\ F - c_1((u_b - l_b)c_2 + l_b), & c_3 < 0 \end{cases}$
      else
        $\phi_i = 1/2(\phi_i + w(t)\phi_{i-1})$
      end if
      $F_i = F(\phi_i) = f_1 E[W] - f_2 S$
    end for
Step 8. Update the source food $\phi^*$ and calculate the fitness $F^*$ of the source food:
      $\phi^* = \operatorname{argmin}_{i\in\{1,2,\ldots,N\}} F_i$,
      $F^* = \min_{i\in\{1,2,\ldots,N\}} F_i$.
Step 9. Check the number of iterations:
      if $t < T_{\max}$
        $t = t + 1$, go to Step 6
      endif
Step 10. Output the optimal sleep parameter $\phi^*$ and the minimum cost $F^*$.
```

Table 4: Optimal sleep parameters of the proposed scheme.

L	μ_2	ϕ^*	F^*
3	0.6	1.7660	25.5421
	1.1	1.9225	24.4361
	5.0	2.0739	24.3177
5	0.6	1.7771	25.1119
	1.1	1.9197	25.0055
	5.0	2.1480	24.8868
10	0.6	1.7640	26.5505
	1.1	1.9243	26.4440
	5.0	2.0710	26.3253

the probability of a warm PM being idle is higher, so the warm PM is more likely to be in the sleep state. This leads to an increase in the energy-saving rate of the system. In addition, we also find that the more the PMs are deployed in the warm pool, the closer the energy-saving rates of the system with different service rates of a request on a warm VM and different service rates of a request on a warm RSE are. This implies that when the number of warm PMs is greater, the energy-saving rate of the system is rarely affected by the service rate of a request on a warm VM and the service rate of a request on a warm RSE.

8. Performance Optimization

Based on the numerical results given in Section 7, we find that, with an increase in the sleep parameter ϕ, the average latency $E[W]$ of requests shows a downward trend, and the energy-saving rate S of the system also decrease. This indicates that when the sleep parameter tends to infinity, the average latency of requests will be minimized, and the energy-saving rate will be close to zero. Obviously, in this case, the energy-saving mechanism will not work at all. Conversely, when the sleep parameter tends to zero, the energy-saving rate will be maximized, and the average latency of requests will become too great to be accepted. In this case, the cloud system cannot provide service normally. How to optimally set the sleep parameter is an important issue in any energy-efficient resource management scheme. In this paper, the criterion for optimization is to balance different performance measures. To do this, we combine the average latency of requests and the energy-saving rate of the system and construct a cost function $F(\phi)$ as follows:

$$F(\phi) = f_1 E[W] - f_2 S, \tag{48}$$

where f_1 and f_2 are the influencing factors for the average latency $E[W]$ of requests and the energy-saving rate S of the system, respectively, in regards to the cost function in the system parameters. It is noted that the higher the cloud user's demand for the response performance is, the larger the parameter f_1 should be set; the higher the cloud provider's demand for the energy efficiency is, the larger the parameter f_2 should be set.

We note that it is difficult to express the average latency $E[W]$ of requests and the energy-saving rate S of the system in closed forms. Therefore, we cannot easily figure out the monotonicity of the cost function. For minimizing the system cost $F(\phi)$ and optimizing the sleep parameter ϕ, we introduce a swarm-based algorithm: SSA.

SSA is an intelligent searching optimization algorithm inspired by the swarming behaviour of salps. In 2017, Seyedali et al. first established a mathematical model of salp chains and presented the SSA to settle many optimization problems [22]. SSA has only one main controlling parameter, so it is simple and easy to implement. However, like other swarm-based algorithms, SSA has the insufficiencies of low convergence precision and slow convergence speed when dealing with high-dimensional complex optimization problems [23]. In the classical SSA optimization process, global exploration and local exploitation are a pair of contradictions. If this process is out of balance, the algorithm easily falls into local optimization and leads to convergence stagnation. Consequently, in this paper, we present an improved SSA by introducing logistic chaotic initialization and adaptive inertia weight [24]. We call this improved SSA LA-SSA.

In this LA-SSA, we firstly adopt a logistic chaotic mapping method to generate the initial salp population. This enhances the diversity of the initial individuals and improves the convergence speed of the algorithm in the early stage. Secondly, we introduce an adaptive inertia weight to update the follower position. The inertia weight reflects the ability of the follower to inherit the salp position from the previous one. If the position of the follower is the locally optimal solution, it is easy to fall into the local optimum and result in convergence stagnation for SSA. Moreover, to improve the convergence precision and help SSA break out of the local optimum, in this paper, the inertia weight of linear decline is introduced, which determines the degree of influence of the previous individual on the current individual. This means salp individuals have strong global convergence capacity and relatively accurate results can be obtained in the later stage.

Table 3 shows the main steps of the LA-SSA.

In addition to utilize the parameters in Table 2, we set $f_1 = 5$, $f_2 = 1$, $N = 50$, $T_{\max} = 100$, $\xi = 4$, $w_s = 0.9$, $w_e = 0.4$, $ub = 5$, and $lb = 0$ as an example in the LA-SSA to optimize the dynamic energy-efficient resource management scheme proposed in this paper. For different capacities L of a hot PM buffer and different service rates μ_2 of a request on a warm VM, we produce the optimal sleep parameter ϕ^* and the minimum cost F^* in Table 4.

From Table 4, we observe that, for the same capacity L of a hot PM buffer, the optimal sleep parameter ϕ^* maintains an upward trend as the service rate μ_2 of a request on a warm VM increases. In contrast, the minimum cost F^* shows a downward trend when the service rate μ_2 of a request on a warm VM goes up.

9. Summary

Considering large amounts of energy consumption generated by cloud data centers, we proposed a dynamic energy-efficient resource management scheme under a multitier cloud architecture. In order to improve the energy efficiency while maintaining the quality of experience for cloud users, we grouped the PMs into different resource pools and introduced a synchronous sleep mechanism to the warm pool. By establishing a Markov chain, we obtained the average latency of requests and the energy-saving rate of the system. In addition, we provided numerical results to study the influence of the sleep mechanism on the system performance. To balance different performance measures, we constructed a system cost function. Moreover, we presented an improved SSA to obtain the optimal sleep parameters and the minimum costs.

In subsequent work, we consider to study energy conservation in cloud systems with heterogeneous cloud users and PMs. Furthermore, we consider to analyze the system models by considering any general stochastic processes, such as Markovian Arrival Process (MAP) and Markovian Service Process (MSP).

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by National Natural Science Foundation (nos. 61872311 and 61973261), China, and was supported in part by MEXT and JSPS KAKENHI , Grant JP17H01825 Japan.

References

[1] Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan, and L. Liu, "Service level agreement based energy-efficient resource management in cloud data centers," *Computers and Electrical Engineering*, vol. 40, no. 5, pp. 1621–1633, 2014.

[2] Y. Hao, J. Cao, T. Ma, and S. Ji, "Adaptive energy-aware scheduling method in a meteorological cloud," *Future Generation Computer Systems*, vol. 101, pp. 1142–1157, 2019.

[3] J. Luo, X. Li, and M. Chen, "Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5804–5816, 2014.

[4] K. Karthiban and J. Raj, "An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm," *Soft Computing*, vol. 24, no. 3, pp. 14933–14942, 2020.

[5] Y. Hao, J. Cao, Q. Wang, and J. Du, "Energy-aware scheduling in edge computing with a clustering method," *Future Generation Computer Systems*, vol. 117, pp. 259–272, 2021.

[6] A. Auday, W. Itani, R. Zantout, and A. Zekri, "Type-aware virtual machine management for energy efficient cloud data centers," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 185–203, 2018.

[7] M. Zakarya and L. Gillam, "An energy aware cost recovery approach for virtual machine migration," *in Proc. International Conference on Economics of Grids, Clouds, Systems, and Services*, pp. 175–190, 2016.

[8] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient VM scheduling for cloud data centers: exact allocation and migration algorithms," in *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pp. 671–678, Delft, Netherlands, 2013.

[9] N. Sharma and R. Guddeti, "Multi-objective energy efficient virtual machines allocation at the cloud data center," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 158–171, 2016.

[10] S. Jin, X. Qie, W. Zhao, W. Yue, and Y. Takahashi, "A clustered virtual machine allocation strategy based on a sleep-mode with wake-up threshold in a cloud environment," *Annals of Operations Research*, vol. 293, no. 1, pp. 193–212, 2019.

[11] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *Proceedings of the 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pp. 500–507, Turin, Italy, 2014.

[12] Sridharshini and V. Sivagami, "Energy-aware scheduling using workload consolidation techniques in cloud environment," *International Journal of Computer Science and Engineering Communications*, vol. 3, no. 3, pp. 1141–1148, 2015.

[13] H. Mora, F. J. Mora Gimeno, M. T. Signes-Pont, and B. Volckaert, "Multilayer architecture model for mobile cloud computing paradigm," *Complexity*, vol. 2019, no. 2, 13 pages, Article ID 3951495, 2019.

[14] M. Usman, A. Samad, Ismail, H. Chizari, and A. Aliyu, "Energy-efficient virtual machine allocation technique using interior search algorithm for cloud data center," in *Proceedings of the 6th ICT International Student Project Conference*, pp. 1–4, Johor, Malaysia, 2017.

[15] A. Beloglazov, "Energy-efficient management of virtual machines in data centers for cloud computing," Ph.D. dissertation, The University of Melbourne, Melbourne, Australia, 2013.

[16] H. Zhu, W. Hai, and X. Liao, "Task scheduling model and virtual machine deployment algorithm for energy consumption optimization in cloud computing," *Systems Engineering-Theory and Practice*, vol. 36, no. 3, pp. 768–778, 2016.

[17] R. Ghosh, F. Longo, V. K. Naik, and K. S. Trivedi, "Modeling and performance analysis of large scale IaaS Clouds," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1216–1234, 2013.

[18] V. Paxson and S. Floyd, "Wide area traffic: the failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1995.

[19] W. Stewart, *Probability, Markov Chains, Queues, and Simulation*, Princeton University Press, Princeton, NJ, USA, 2009.

[20] G. Latouche and V. Ramaswami, "Introduction to matrix analytic methods in stochastic modeling," Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[21] M. Neuts, *Matrix-Geometric Solutions in Stochastic Models*, Johns Hopkins University Press, Baltimore, MD, USA, 1984.

[22] M. Seyedali, H. Amir, Z. Seyedeh, S. Shahrzad, F. Hossams, and M. Seyed, "Salp swarm algorithm: a bio-inspired optimizer for engineering design problems," *Advances in Engineering Software*, vol. 114, pp. 163–191, 2017.

[23] J. Wu, R. Nan, and L. Chen, "Improved salp swarm algorithm based on weight factor and adaptive mutation," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 31, no. 3, pp. 1–23, 2019.

[24] P. An, "Particle swarm optimization algorithm based on chaotic theory and adaptive inertia weight," *Journal of Nanoelectronics and Optoelectronics*, vol. 12, no. 4, pp. 404–408, 2017.