Journal of Network and Systems Management (2019) 27:860–882
https://doi.org/10.1007/s10922-019-09489-w

An Energy-Efficient Strategy for Virtual Machine Allocation over Cloud Data Centers

Xiuchen Qie[1,2] · Shunfu Jin[1,2] · Wuyi Yue[3]

Received: 21 December 2017 / Revised: 23 December 2018 / Accepted: 8 January 2019 /
Published online: 17 January 2019

Abstract

With the increase in the scale of cloud data centers, more attention is being focused on the issue of energy conservation. In order to achieve greener, more efficient computing in cloud data centers, in this paper, we propose an energy-efficient Virtual Machine (VM) allocation strategy with an asynchronous multi-sleep mode and an adaptive task-migration scheme. The VMs hosted in a virtual cluster are divided into two modules, namely, Module I and Module II. The VMs in Module I are always awake, whereas the VMs in Module II will go to sleep independently, if possible. Accordingly, a queuing model with a partial asynchronous multiple vacations is established to capture the working principle of the proposed strategy. Using the method of a matrix-geometric solution, performance measures in terms of the average response time of tasks and the energy saving rate of the system are mathematically derived. Numerical experiments with analysis and simulation are provided to validate the proposed VM allocation strategy and to estimate the influence of system parameters on performance measures. Finally, a system cost function is constructed to trade off different performance measures, and an intelligent searching algorithm is employed to optimize the number of VMs in Module II and the sleeping parameter in the same time.

✉ Shunfu Jin
jsf@ysu.edu.cn

Xiuchen Qie
qiexiuchen@126.com

Wuyi Yue
yue@konan-u.ac.jp

[1] School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China

[2] Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Qinhuangdao 066004, China

[3] Department of Intelligence and Informatics, Konan University, Kobe 658-8501, Japan

Keywords Cloud data center · Energy-efficient strategy · VM allocation · Multi-sleep · Task-migration · Intelligent searching algorithm

1 Introduction

Cloud data centers are growing exponentially in number and size to accommodate an escalating number of users and an expansion in applications. In the current "Cisco Global Cloud Index", IT manufacturer Cisco predicts that by 2019, more than four-fifths of the workload in data centers will be handled in cloud data centers [1]. As a result, the tremendous amount of energy consumption and carbon dioxide emissions from cloud data centers are becoming a great concern worldwide. According to a report from Natural Resources Defense Council (NRDC), cloud data center energy consumption is estimated to reach 140 billion kW h by 2020, which will be responsible for the emission of nearly 150 million tons of carbon pollution [2]. Therefore, producing energy-efficient systems has become a focus for the development and operation of cloud data centers.

The main contributions of this paper are summarized as follows:

1. For reducing energy consumption and achieving greener cloud computing, we propose an energy-efficient Virtual Machine (VM) allocation strategy with an asynchronous multi-sleep mode and an adaptive task-migration scheme.
2. We present a method to model the proposed VM allocation strategy and to evaluate the system performance in terms of the average response time of tasks and the energy saving rate of the system.
3. With the help of an intelligent searching algorithm, we optimize the proposed VM allocation strategy to trade off different performance measures, such as the average response time of tasks and the energy saving rate of the system.

2 Review of Related Literature

In this section, we review the research studies into energy saving strategies in cloud data centers, sleep mode based energy saving strategies and enhanced particle swarm optimization algorithms. And then, we outline the motivation for our research.

2.1 Energy Saving Strategies in Cloud Data Centers

In cloud data centers, an enormous amount of energy can be wasted due to excessive provisioning [3, 4], while Service Level Agreements (SLA) violations can be risked by insufficient provisioning [5, 6]. In [7], by introducing dynamic voltage and frequency scaling (DVFS) methods as part of a consolidation approach, Arianyan et al. proposed a novel fuzzy multi-criteria and multi-objective resource management solution to reduce energy consumption and alleviate SLA violation. In [8], Jungmin et al. proposed a dynamic overbooking strategy, allocating a more precise amount of resources to VMs and traffic with a dynamically changing workload. In this strategy, both of the energy consumption and the SLA violations were considered. In [9], for the purpose of minimizing the energy consumption, Hosseinimotlagh et al. introduced an optimal utilization level of a host to execute a certain number of instructions. Furthermore, they proposed a VM scheduling algorithm based on unsurpassed utilization level in order to derive the optimal energy consumption while satisfying a given Quality of Service (QoS) requirement. The literature mentioned above has contributed to reducing energy consumption while guaranteeing response performance in cloud data centers. However, the energy consumption generated by idle hosts in cloud data centers has been ignored.

2.2 Sleep Mode Based Energy Saving Strategies

The use of a sleep mode is an efficient approach for reducing the energy consumption in data centers [10]. In [11], Duan et al. proposed a dynamic idle interval prediction scheme that can estimate the future idle interval length of a CPU and thereby choose the most cost-effective sleep state to minimize the power consumption during runtime. In [12], Sarji et al. proposed two energy models based on the statistical analysis of a server's operational behavior. With these models, the Energy Savings Engine (ESE) in the cloud provider decided either to migrate the VMs from a lightly-loaded server and then put the machine into a sleep mode, or to keep the current server running and ready for receiving any new tasks. In [13], Liu et al. proposed a sleep state management model to balance the system's energy consumption and the response performance. In this model, idle nodes were classified into different groups according to their sleep states. In the resource allocation process, nodes with the highest level of readiness were preferentially provided to the application. This research emphasized applying a sleep mode to a Physical Machine (PM).

To improve the energy efficiency of cloud data centers, Jin et al. proposed an energy-efficient strategy with a speed switch on PMs and a synchronous multi-sleep mode on partial VMs [14]. In [15], by applying dynamic power management (DPM) technology to PMs and introducing synchronous semi-sleep mode to partial VMs, Jin et al. proposed a novel VM scheduling strategy for reducing energy consumption in cloud data centers. Both of the studies mentioned above applied a synchronous sleep mode to the VMs. However, there has so far been no research into the effect of asynchronous sleep modes on the level of VMs in cloud data centers.

2.3 Enhanced Particle Swarm Optimization Algorithms

In 1995, particle swarm optimization (PSO) was developed as an effective tool for function optimization. Since then, numerous research studies on improving the searching ability of PSO algorithms have appeared. In [16], to enhance the performance of PSO algorithms, Cao et al. improved PSO algorithms by introducing a nonlinear dynamic inertia weight and two dynamic learning factors. In [17], Zhang et al. proposed a novel

PSO algorithm based on an adaptive inertia weight and chaos optimization, which enhanced the local optimization ability of the PSO algorithm and helped objective functions easily jump out of local optimum. In [18], Tian presented a new PSO algorithm by introducing chaotic maps (Tent and Logistic), a Gaussian mutation mechanism, and a local re-initialization strategy into the standard PSO algorithm. The chaotic map is utilized to generate uniformly distributed particles for the purpose of improving the quality of the initial population. From the research mentioned above, we note that the searching ability of PSO algorithms are greatly influenced by the inertia weight and the initial positions of particles.

2.4 Motivation for Our Research

Inspired by the literature mentioned above, in this paper, we propose an energy-efficient strategy for VM allocation over cloud data centers. We note that letting all the VMs in a virtual cluster go to sleep may degrade the quality of cloud service. Taking both the response performance and the energy conservation level into consideration, we divide the VMs in a virtual cluster into two parts: Module I and Module II. The VMs in Module I stay awake all the time to provide an instant cloud service for accomplishing tasks, while the VMs in Module II may go to sleep whenever possible to reduce energy consumption. The energy consumption of a VM is related to the processing speed of the VM. Generally speaking, the higher the processing speed is, the more energy will be consumed. In our proposed strategy, the VMs in Module I process tasks at a higher speed to guarantee the response performance, while the VMs in Module II process tasks at a lower speed to save more energy. In order to further enhance the energy efficiency of the proposed strategy, we introduce an adaptive task-migration scheme which shifts an unfinished task in Module II to an idle VM in Module II. When an idle VM appears in Module I, a task being processed on a VM in Module II will migrate to the idle VM in Module I, and then the just evacuated VM in Module II will go to sleep independently. To analyze the proposed strategy, we build a queueing model with partial asynchronous multiple vacations by using a matrix geometric solution, and investigate the system performance through theoretical analysis and simulation experiments. Finally, in order to optimize the proposed strategy, we construct a cost function to balance different system performance levels, such as the average response time of tasks and the energy saving rate of the system, and apply the PSO algorithm to optimize the system parameter settings.

The rest of this paper is organized as follows. In Sect. 3, a novel energy-efficient VM allocation strategy is proposed and a queueing model is built accordingly. In Sect. 4, the queueing model is analyzed by using a matrix geometric solution. In Sect. 5, the expressions of the average response time of tasks and the energy saving rate of the system are derived. With numerical experiments, the system performance is evaluated in Sect. 6. In Sect. 7, an intelligent searching algorithm is used to optimize the number of the VMs in Module II and the sleeping parameter together. Finally, Sect. 8 outlines conclusions from the research.

3 Energy-Efficient VM Allocation Strategy and System Model

In this section, an energy-efficient VM allocation strategy with an asynchronous multi-sleep mode and an adaptive task-migration scheme is proposed. Accordingly, a type of continuous-time multi-server queueing model with partial asynchronous multiple vacations is established.

3.1 Energy-Efficient VM Allocation Strategy

In conventional cloud data centers, all the VMs remain open waiting for the arrival of tasks regardless of current traffic. This may result in a great deal of energy waste. To get around this problem, a novel VM allocation strategy with an asynchronous multi-sleep mode and an adaptive task-migration scheme is proposed in this paper. It should be emphasized that the asynchronous multi-sleep mode considered in this paper is at the level of VMs rather than that of PMs.

Given the processing capability and the energy conservation level, all the VMs hosted in a virtual cluster are divided into two modules, namely, Module I and Module II. The VMs in Module I stay awake all the time and run on a high speed when tasks arrive. Whereas, the VMs in Module II switch between the sleep state and the busy state.

For a busy VM in Module II, state transition only happens at the instant when a task is completely processed. Given that a task is completely processed in Module II, if the system buffer is empty, the evacuated VM in Module II will go to sleep. Once a VM in Module II switches to the sleep state, a sleep timer will be started, the data in the memory will be saved to a hibernation file on the hard disk, and then the power of the other accessories, except for the memory, will be cut off, so that the VM will no longer be available for processing tasks in the system. Given that a task is completely processed in Module I, if the system buffer is empty and there is at least one task being processed in Module II, one of the tasks being processed in Module II will be migrated to Module I, and then the evacuated VM in Module II will go to sleep. We note that the task-migration considered in this paper is a kind of online VM-migration between different modules within a virtual cluster.

For a sleeping VM in Module II, state transition only happens at the instant when a sleep timer expires. At the moment that a sleep timer expires, the sleeping VM in Module II will listen to the system and decide whether to keep sleeping or to wake up. If the system buffer is empty, another sleep timer will be started and the sleeping VM in Module II will begin another sleep period, so that multiple sleep periods are formed. Otherwise, the sleeping VM in Module II will wake up to process the first task waiting in the system buffer on a lower speed. Once a VM in Module II switches to the awake state, the corresponding sleep timer will be turned off, the data of the hibernation file on the hard disk will be read into the memory, and then the power of all accessories will be turned on, so that the VM will be available for processing tasks in the system.

With our proposed sleep mode, energy consumption could be saved, but the incoming tasks may not receive timely service. We speculate that the average response time of tasks is lower with a smaller number of VMs in Module II, while the energy saving rate of the system is higher with a suitable number of VMs in Module II. We also speculate that the average response time of tasks is lower with a shorter sleep period, while the energy saving rate of the system is higher with a longer sleep period. Given this, we should optimize the proposed strategy by trading off the average response time and the energy saving rate of the system. (The optimization approach will be given in Sect. 7.)

We show the state transition of a virtual cluster in the cloud data center under the proposed VM allocation strategy in Fig. 1.

As shown in Fig. 1, in the proposed strategy, the numbers of the VMs in Module I and Module II are denoted as c and d, respectively. All the VMs hosted in one virtual cluster are dominated by a control server, in which several sleep timers and a VM scheduler are deployed. Each sleep timer is responsible for controlling the sleep time of a VM in Module II. The numbers of tasks in the system, busy VMs in Module I and sleeping VMs in Module II are denoted as M, b and s, respectively. Given these parameters, the VM scheduler adjusts the VM state.

According to the state of VMs both in Module I and in Module II, we consider three cases as follows:

Case 1 There is at least one idle VM in Module I, and all the VMs in Module II are sleeping.

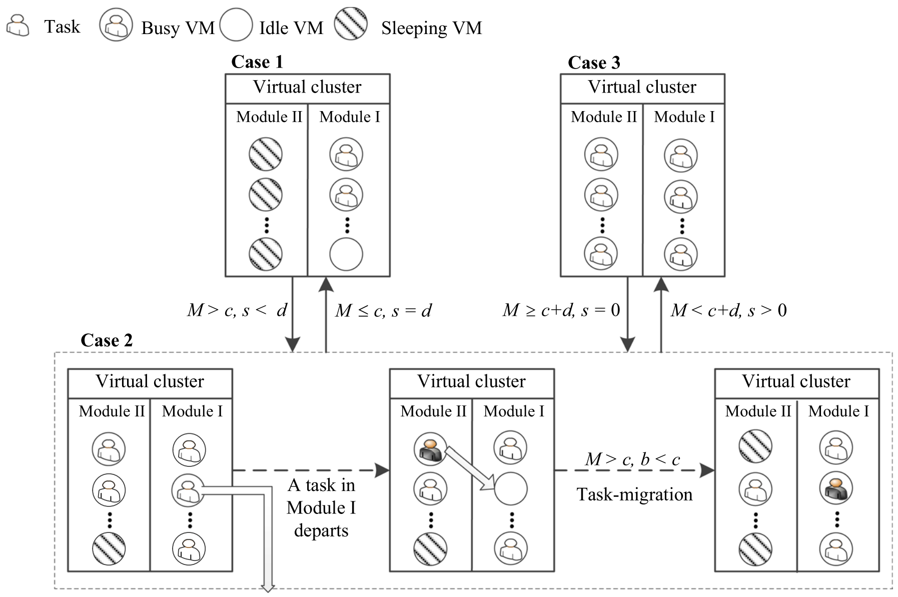


Fig. 1 State transition of a virtual cluster considered in three cases in this paper

In **Case 1**, each arriving task could be processed immediately on a high speed in Module I. However, as more tasks arrive at the system, more VMs in Module I will be occupied. If there are no VMs available, a newly incoming task has to wait in the system buffer. Once a sleep timer expires, the corresponding VM in Module II will wake up to process the first task queueing in the system buffer on a low speed, and then the system will be converted to **Case 2**.

Case 2 All the VMs in Module I are busy, and there is at least one sleeping VM in Module II.

In **Case 2**, with the departures of the tasks, more VMs in Module II will go to sleep. At the moment a task process is completed in Module I and there are no tasks waiting in the system buffer, i.e., $M > c$ and $b < c$, one of the tasks being processed in Module II will be migrated to Module I, then the just evacuated VM in Module II will go to sleep. When all the VMs in Module II are asleep, i.e., $M \leq c$ and $s = d$, the system will be converted back to **Case 1**.

We note that for **Case 2**, there are no VMs available in the system, so a newly incoming task will queue in the system buffer. When a task is completely processed on one of the VMs in Module I, the just evacuated VM in Module I will process the first task queueing in the system buffer on a high speed. Also, when one of the sleep timers expires, the corresponding VM in Module II will wake up and process the first task queueing in the system buffer on a low speed. Once all the VMs in Module II wake up, i.e., $M \geq c + d$ and $s = 0$, the system will be converted to **Case 3**.

Case 3 All the VMs in both Module I and Module II are busy.

In **Case 3**, a newly incoming task has to wait in the system buffer since all the VMs hosted in the virtual cluster are occupied. With the departures of the tasks, more tasks in the system buffer will be processed on the evacuated VMs. Once the system buffer is empty and there exists at least one sleeping VM in Module II, i.e., $M < c + d$ and $s > 0$, the system will be converted back to **Case 2**.

3.2 System Model

In cloud data centers, there are many available task scheduling schemes, such as event-driven scheduling schemes, preemptive scheduling schemes and random scheduling schemes. In our paper, we assume that an available VM can be assigned to the first task queueing in the system buffer. Regarding a task as a customer, a VM as an independent server, a sleep period as a vacation and multiple sleep periods as multiple vacations, we model the proposed strategy as a type of novel queueing model with partial asynchronous multiple vacations.

The system model is described as being in an infinite state. Let random variable $N(t) = i, i \in \{0, 1, \dots\}$ be the total number of tasks in system at instant t. $N(t)$ is also called the system level. Let random variable $J(t) = j, j \in \{0, 1, \dots, d\}$ be the number of busy VMs in Module II at instant t. $J(t)$ is also called the system stage. $\{N(t), J(t), t \geq 0\}$ constitutes a two-dimensional continuous-time stochastic process with the state-space $\boldsymbol{\Omega}$ as follows:

$$\begin{aligned}\boldsymbol{\Omega} = \{(i,0) : 0 \leq i \leq c\} &\cup \{(i,j) : c < i \leq c+d,\ 0 \leq j \leq i-c\} \\ &\cup \{(i,j) : i > c+d, 0 \leq j \leq d\}.\end{aligned} \tag{1}$$

In our research, we focus on user initiated tasks [19], and we make the following assumptions. We suppose that the arrival intervals of tasks, the service times of a task processed in Module I and in Module II, and the time lengths of a sleep timer are independent, identically distributed (i.i.d) random variables. Task arrivals are assumed to follow a Poisson process with parameter λ ($\lambda > 0$), the service times of a task processed in Module I and in Module II are assumed to follow exponential distributions with parameters μ_1 ($\mu_1 > 0$) and μ_2 ($0 < \mu_2 < \mu_1$), respectively. In addition, the time length of a sleep timer is assumed to follow an exponential distribution with parameter θ, called the sleeping parameter. It should be noted that in the system model, we assume that no time is taken for a task to migrate or for a sleeping VM to wake up.

Based on the assumptions above, $\{N(t), J(t), t \geq 0\}$ can be regarded as a two-dimensional continuous time Markov chain (CTMC).

We define $\pi_{i,j}$ as the steady-state probability distribution of the system model for the system level being equal to i and the system stage being equal to j. $\pi_{i,j}$ is then given as follows:

$$\pi_{i,j} = \lim_{t\to\infty} P\{N(t) = i, J(t) = j\},\ (i,j) \in \boldsymbol{\Omega}. \tag{2}$$

We define $\boldsymbol{\pi}_i$ as the steady-state probability distribution when the system level is i. $\boldsymbol{\pi}_i$ can be given as follows:

$$\boldsymbol{\pi}_i = \begin{cases} \pi_{i,0},\ 0 \leqslant i \leqslant c \\ (\pi_{i,0}, \pi_{i,1}, \dots, \pi_{i,i-c}),\ c < i \leqslant c+d \\ (\pi_{i,0}, \pi_{i,1}, \dots, \pi_{i,d}),\ i > c+d. \end{cases} \tag{3}$$

The steady-state probability distribution $\boldsymbol{\Pi}$ of the two-dimensional CTMC is composed of $\boldsymbol{\pi}_i$ ($i \geq 0$). $\boldsymbol{\Pi}$ is given as follows:

$$\boldsymbol{\Pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \dots). \tag{4}$$

4 Model Analysis

In this section, the transition rate matrix of the two-dimensional CTMC is firstly investigated. Then, the steady-state probability distribution of system model is derived.

4.1 Transition Rate Matrix

Let $\boldsymbol{Q}$ be the one step state transition rate matrix of the two-dimensional CTMC $\{(N(t), J(t)), t \geq 0\}$. Based on the system level, $\boldsymbol{Q}$ is separated into several sub-matrices. Let $\boldsymbol{Q}_{k,l}$ be the one step state transition rate sub-matrix for the system level changing from k $(k = 0, 1, \ldots)$ to l $(l = 0, 1, \ldots)$. For convenience of presentation, we denote $\boldsymbol{Q}_{k,k-1}$, $\boldsymbol{Q}_{k,k+1}$ and $\boldsymbol{Q}_{k,k}$ as $\boldsymbol{B}_k$, $\boldsymbol{C}_k$ and $\boldsymbol{A}_k$, respectively. $\boldsymbol{B}_k$, $\boldsymbol{C}_k$ and $\boldsymbol{A}_k$ are discussed in the following cases.

1. When the initial system level k ranges from 0 to c, k VMs in Module I are busy and all the VMs in Module II are sleeping.

For the case of $k = 0$, there are no tasks at all in the system. This means that the possible state transitions are from (0, 0) to (1, 0) and from (0, 0) to (0, 0). If a task arrives at the system, the system level will increase by one but the system stage will remain unchanged, i.e., the system state will transform to (1, 0) from (0, 0) with the transition rate λ. Otherwise, the system state will remain fixed at (0, 0) with the transition rate $-\lambda$. Thus, C_0 and A_0 are given as follows:

$$C_0 = \lambda, \ A_0 = -\lambda.$$

For the case of $0 < k \leqslant c$, all the tasks in the system are being processed on the VMs in Module I. If a task is completely processed, the system level will decrease by one but the system stage will remain unchanged, i.e., the system state will transform to $(k-1, 0)$ from $(k, 0)$ with the transition rate $k\mu_1$. If a task arrives at the system, the system level will increase by one but the system stage will remain unchanged, i.e., the system state will transfer to $(k+1, 0)$ from $(k, 0)$ with the transition rate λ. Otherwise, the system state will remain fixed at $(k, 0)$ with the transition rate $-(\lambda + k\mu_1)$. Thus, B_k, C_k and A_k are given as follows:

$$B_k = k\mu_1, \ C_k = \lambda, \ A_k = -(\lambda + k\mu_1).$$

2. When the initial system level k ranges from $(c+1)$ to $(c+d)$, all the VMs in Module I are busy, while at most $(k-c)$ VMs in Module II are busy.

For the case of $k = c + x, \ x = 1, 2, \ldots, d-1$, the number of busy VMs in Module I is c, while in Module II, there are at most x busy VMs.

If a task is completely processed and there is at least one task in the system buffer, the first task queueing in the system buffer will occupy the evacuated VM to receive service. Consequently, the system level will decrease by one, but the system stage will remain fixed, i.e., the system state will transform to $(k-1, n)$ from (k, n) with the transition rate $(c\mu_1 + n\mu_2)$, where n $(0 \leq n \leq x)$ is the number of busy VMs in Module II. If a task is completely processed on the VM in Module I and there are no tasks in the system buffer, one of the tasks being processed in Module II will migrate to the evacuated VM in Module I and the just-evacuated VM in Module II will start sleeping. If a task is completely processed on the VM in Module II and there are no tasks in the system buffer, the evacuated VM in Module II will start sleeping directly. Consequently, both the system level and the system stage will decrease by one, i.e., the system state will transform to $(k-1, x-1)$ from (k, x) with the transition rate $(c\mu_1 + x\mu_2)$. Thus, $\boldsymbol{B}_k$ is a rectangular $(x+1) \times x$ matrix and is given as follows:

$$\boldsymbol{B}_k = \begin{pmatrix} c\mu_1 & & & \\ & c\mu_1 + \mu_2 & & \\ & & \ddots & \\ & & & c\mu_1 + (x-1)\mu_2 \\ & & & c\mu_1 + x\mu_2 \end{pmatrix}.$$

None of VMs in Module II will wake up before their corresponding sleep timers expire, even though the system buffer is not empty. If a task arrives at the system before one of the sleep timers expires, the system level will increase by one but the system stage will remain fixed, i.e., the system state will transform to $(k+1, n)$ from (k, n) with the transition rate λ. Thus, $\boldsymbol{C}_k$ is a rectangular $(x+1) \times (x+2)$ matrix and is given as follows:

$$\boldsymbol{C}_k = \begin{pmatrix} \lambda & & & & 0 \\ & \lambda & & & 0 \\ & & \ddots & & \vdots \\ & & & \lambda & 0 \\ & & & & \lambda \; 0 \end{pmatrix}.$$

If one of the sleep timers expires, the corresponding VM in Module II will wake up and process the first task queueing in the buffer. Consequently, the system level k will remain fixed but the system stage n will increase by one, i.e., the system state will transform to $(k, n+1)$ from (k, n) with the transition rate $(d-n)\theta$. Otherwise, the system state will remain fixed: when the system buffer is not empty, the transition rate is $-h_n$, where $h_n = \lambda + c\mu_1 + n\mu_2 + (d-n)\theta$; when the system buffer is empty, the transition rate is $-(\lambda + c\mu_1 + x\mu_2)$. Thus, $\boldsymbol{A}_k$ is a rectangular $(x+1) \times (x+1)$ matrix and is given as follows:

$$\boldsymbol{A}_k = \begin{pmatrix} -h_0 & d\theta & & & \\ & -h_1 & (d-1)\theta & & \\ & & \ddots & \ddots & \\ & & & -h_{x-1} & (d-x+1)\theta \\ & & & & -(\lambda + c\mu_1 + x\mu_2) \end{pmatrix}.$$

For the case of $k = c + d$, the number of tasks in the system is equal to the total number of VMs. This is really just a specialized case discussed previously. $\boldsymbol{B}_k$ is a rectangular $(d+1) \times d$ matrix, $\boldsymbol{C}_k$ and $\boldsymbol{A}_k$ are square matrices of the order $(d+1) \times (d+1)$. $\boldsymbol{B}_k$, $\boldsymbol{C}_k$ and $\boldsymbol{A}_k$ are given as follows:

$$\boldsymbol{B}_k = \begin{pmatrix} c\mu_1 & & & & \\ & c\mu_1 + \mu_2 & & & \\ & & \ddots & & \\ & & & c\mu_1 + (d-1)\mu_2 & \\ & & & & c\mu_1 + d\mu_2 \end{pmatrix},$$

$$\boldsymbol{C}_k = \begin{pmatrix} \lambda & & & & \\ & \lambda & & & \\ & & \ddots & & \\ & & & \lambda & \\ & & & & \lambda \end{pmatrix},$$

$$\boldsymbol{A}_k = \begin{pmatrix} -h_0 & d\theta & & & \\ & -h_1 & (d-1)\theta & & \\ & & \ddots & \ddots & \\ & & & -h_{d-1} & \theta \\ & & & & -h_d \end{pmatrix}.$$

3. When the initial system level is greater than the total number of VMs, i.e., $k > c + d$, all the VMs in Module I are busy, while the VMs in Module II are either busy or sleeping. $\boldsymbol{B}_k$, $\boldsymbol{C}_k$ and $\boldsymbol{A}_k$ are square matrices of the order $(d+1) \times (d+1)$. Similar to the discussion in item (2), the sub-matrices $\boldsymbol{B}_k$, $\boldsymbol{C}_k$ and $\boldsymbol{A}_k$ are given as follows:

$$\boldsymbol{B}_k = \begin{pmatrix} c\mu_1 & & & & \\ & c\mu_1 + \mu_2 & & & \\ & & \ddots & & \\ & & & c\mu_1 + (d-1)\mu_2 & \\ & & & & c\mu_1 + d\mu_2 \end{pmatrix},$$

$$\boldsymbol{C}_k = \begin{pmatrix} \lambda & & & & \\ & \lambda & & & \\ & & \ddots & & \\ & & & \lambda & \\ & & & & \lambda \end{pmatrix},$$

$$\boldsymbol{A}_k = \begin{pmatrix} -h_0 & d\theta & & & \\ & -h_1 & (d-1)\theta & & \\ & & \ddots & \ddots & \\ & & & -h_{d-1} & \theta \\ & & & & -h_d \end{pmatrix}.$$

Now, all the sub-matrices in the one step state transition rate matrix $\boldsymbol{Q}$ have been addressed. Starting from the system level $(c+d)$, the sub-matrices $\boldsymbol{A}_k$ and $\boldsymbol{C}_k$ in $\boldsymbol{Q}$ are repeated forever. Starting from the system level $(c+d+1)$, the sub-matrices $\boldsymbol{B}_k$ in $\boldsymbol{Q}$ are repeated forever. The repetitive sub-matrices $\boldsymbol{B}_k$, $\boldsymbol{A}_k$ and $\boldsymbol{C}_k$ are represented by $\boldsymbol{B}$, $\boldsymbol{A}$ and $\boldsymbol{C}$, respectively. For this, $\boldsymbol{Q}$ is written as follows:

$$\boldsymbol{Q} = \begin{pmatrix} A_0 & C_0 & & & & & & & \\ B_1 & A_1 & C_1 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & B_c & A_c & C_c & & & & \\ & & & \boldsymbol{B}_{c+1} & \boldsymbol{A}_{c+1} & \boldsymbol{C}_{c+1} & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \boldsymbol{B}_{c+d} & \boldsymbol{A} & \boldsymbol{C} & \\ & & & & & & \boldsymbol{B} & \boldsymbol{A} & \boldsymbol{C} \\ & & & & & & & \ddots & \ddots & \ddots \end{pmatrix}. \tag{5}$$

The block-tridiagonal structure of the one step state transition rate matrix $\boldsymbol{Q}$ shows that the state transitions occur only between adjacent system levels. Referring to [20], we know that the two-dimensional CTMC $\{N(t), J(t), t \geq 0\}$ can be seen as a type of Quasi Birth-and-Death (QBD) process.

4.2 Steady-State Probability Distribution

For the QBD process $\{N(t), J(t), t \geq 0\}$ with the one step state transition rate matrix $\boldsymbol{Q}$, the necessary and sufficient condition for positive recurrence is that the matrix quadratic equation

$$\boldsymbol{R}^2\boldsymbol{B} + \boldsymbol{R}\boldsymbol{A} + \boldsymbol{C} = \boldsymbol{0} \tag{6}$$

has the minimal non-negative solution $\boldsymbol{R}$ with the spectral radius $SP(\boldsymbol{R}) < 1$. This solution, called the rate matrix and denoted by $\boldsymbol{R}$, can be explicitly determined.

From Sect. 4.1, we find that the sub-matrices $\boldsymbol{B}$, $\boldsymbol{A}$ and $\boldsymbol{C}$ are upper-triangular matrices. So, the rate matrix $\boldsymbol{R}$ must be an upper-triangular matrix and can be expressed as follows:

$$\boldsymbol{R} = \begin{pmatrix} r_0 & r_{01} & r_{02} & \cdots & r_{0d-1} & r_{0d} \\ & r_1 & r_{12} & \cdots & r_{1d-1} & r_{1d} \\ & & r_2 & \cdots & r_{2d-1} & r_{2d} \\ & & & \ddots & \vdots & \vdots \\ & & & & r_{d-1} & r_{d-1d} \\ & & & & & r_d \end{pmatrix}. \tag{7}$$

Then, the elements of $\boldsymbol{R}^2$ are

$$(\boldsymbol{R}^2)_{kk} = r_k^2, \; 0 \leqslant k \leqslant d,$$

$$(\boldsymbol{R}^2)_{jk} = \sum_{i=j}^{k} r_{ji}r_{ik}, \; 0 \leqslant j \leqslant d-1, j+1 \leqslant k \leqslant d.$$

Substituting $\boldsymbol{R}^2$, $\boldsymbol{R}$, $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ into Eq. (6) yields a set of equations:

$$\begin{cases} (c\mu_1 + k\mu_2)r_k^2 - h_k r_k + \lambda = 0,\ 0 \leqslant k \leqslant d \\ (c\mu_1 + k\mu_2)\sum_{i=j}^{k} r_{ji}r_{ik} - h_k r_{jk} + (d-k+1)\theta r_{j,k-1} \\ \quad = 0,\ 0 \leqslant j \leqslant d-1,\ j+1 \leqslant k \leqslant d. \end{cases} \tag{8}$$

If the traffic load $\rho = \lambda(c\mu_1 + d\mu_2)^{-1} < 1$, it can be proven that the first equation of Eq. (8) has two real roots $0 < r_k < 1$ and $r_k^* \geq 1$. Note that the diagonal elements of $\boldsymbol{R}$ are r_k $(0 \leq k \leq d)$ and the spectral radius of $\boldsymbol{R}$ satisfies:

$$SP(\boldsymbol{R}) = max\{r_0, r_0, \dots, r_d\} < 1. \tag{9}$$

The off-diagonal elements of $\boldsymbol{R}$ satisfy the last equation of Eq. (8). It is an arduous task to give a general expression for r_{jk} $(0 \leq j \leq d-1, j+1 \leq k \leq d)$ in closed-form, so we recursively compute the off-diagonal elements based on the diagonal elements.

Since the QBD process with the one step state transition rate matrix $\boldsymbol{Q}$ is positive recurrent, the stationary distribution is easily expressed in the matrix geometric form with the rate matrix $\boldsymbol{R}$ as follows:

$$\boldsymbol{\pi}_i = \boldsymbol{\pi}_{c+d}\boldsymbol{R}^{i-(c+d)},\ i \geq c+d. \tag{10}$$

In order to obtain the unknown stationary distribution $\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \cdots, \boldsymbol{\pi}_{c+d}$, we construct a square matrix $B[\boldsymbol{R}]$ of the order $\left[c + \frac{(d+1)(d+2)}{2}\right] \times \left[c + \frac{(d+1)(d+2)}{2}\right]$ as follows:

$$B[\boldsymbol{R}] = \begin{pmatrix} A_0 & C_0 & & & & & & & \\ B_1 & A_1 & C_1 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & B_c & A_c & C_c & & & & \\ & & & \boldsymbol{B}_{c+1} & \boldsymbol{A}_{c+1} & \boldsymbol{C}_{c+1} & & & \\ & & & & \ddots & \ddots & \ddots & & \\ & & & & & \boldsymbol{B}_{c+d-1} & \boldsymbol{A}_{c+d-1} & \boldsymbol{C}_{c+d-1} \\ & & & & & & \boldsymbol{B}_{c+d} & \boldsymbol{RB}+\boldsymbol{A} \end{pmatrix}. \tag{11}$$

Using the method of a matrix geometric solution, we can construct an augmented matrix equation as

$$(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \dots, \boldsymbol{\pi}_{c+d}) \left(B[\boldsymbol{R}] \begin{array}{:c} \boldsymbol{e}_1 \\ (\boldsymbol{I}-\boldsymbol{R})^{-1}\boldsymbol{e}_2 \end{array} \right) = (\underbrace{0, 0, \dots, 0}_{c+\frac{(d+1)(d+2)}{2}}, 1) \tag{12}$$

where $\boldsymbol{e}_1$ is a $\left[c + \frac{d(d+1)}{2}\right] \times 1$ vector with ones, and $\boldsymbol{e}_2$ is a $(d+1) \times 1$ vector with ones.

Applying the Gauss–Seidel method [21] to solve Eq. (12), we can obtain $\boldsymbol{\pi}_0$, $\boldsymbol{\pi}_1$, ..., $\boldsymbol{\pi}_{c+d}$. Substituting $\boldsymbol{\pi}_{c+d}$ obtained in Eq. (12) into Eq. (10), we can obtain

$\boldsymbol{\pi}_i$ $(i = c + d + 1, c + d + 2, \ldots)$. Then the steady-state probability distribution $\boldsymbol{\Pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots)$ of the system can be given mathematically.

5 Performance Measures

In this section, the performance measures in terms of the average response time of tasks and the energy saving rate of the system are mathematically evaluated.

We define the response time of a task as the duration from the instant a task arrives at the system to the instant this task is completely processed.

Based on the steady-state probability distribution of the system model given in Sect. 4.2, the average response time $E[T]$ of tasks is given as follows:

$$E[T] = \frac{1}{\lambda}\left(\sum_{i=0}^{c} i\pi_{i0} + \sum_{i=c+1}^{c+d}\sum_{j=0}^{i-c} i\pi_{i,j} + \sum_{i=c+d+1}^{\infty}\sum_{j=0}^{d} i\pi_{i,j}\right). \tag{13}$$

In our proposed VM allocation strategy, energy consumption can be reduced during the sleep period. We let ω $(\omega > 0)$ be the energy consumption per unit time for a busy VM in Module II, and ω_s $(\omega_s > 0)$ be the energy consumption per unit time for a sleeping VM in Module II. Obviously, $\omega > \omega_s$. We note that additional energy will be consumed when a task migrates from Module II to Module I, when a VM in Module II listens to the system buffer, as well as when a VM in Module II wakes up from sleep state. Let ω_m $(\omega_m > 0)$, ω_l $(\omega_l > 0)$ and ω_u $(\omega_u > 0)$ be the energy consumption for each migration, listening and wakeup, respectively.

We define the energy saving rate of the system as the energy conservation per unit time with our proposed strategy. Based on the discussions above and the steady-state probability distribution of the system model given in Sect. 4.2, the energy saving rate ψ of the system is given as follows:

$$\begin{aligned}\psi = (\omega - \omega_s)\sum_{i=0}^{\infty}\sum_{j=0}^{d}(d-j)\pi_{ij} - \Bigg(\omega_m \sum_{i=c+1}^{c+d}\sum_{j=1}^{d} c\mu_1\pi_{ij} \\ + \omega_l \sum_{i=0}^{\infty}\sum_{j=0}^{d}\theta(d-j)\pi_{ij} + \omega_u \sum_{i=c+j+1}^{\infty}\sum_{j=0}^{d-1}\theta(d-j)\pi_{ij}\Bigg)\end{aligned} \tag{14}$$

6 Numerical Experiments

In order to evaluate the average response time of tasks and the energy saving rate of the system with the proposed VM allocation strategy, we provide numerical experiments with analysis and simulation. The analysis results are obtained based on Eqs. (13) and (14) using Matlab 2011a. The simulation results are obtained by using MyEclipse 2014. We create a JOB class with attributes in terms of UNARRIVE, WAIT, RUNHIGH, RUNLOW and FINISH to record the task state. We also create a SERVER class with attributes in terms of SLEEP, IDLE,

Table 1 Parameter settings in numerical experiments

Parameters	Value
Total number of VMs in the system	$c + d = 50$
Arrival rate of tasks	$\lambda = 4.50\ \text{ms}^{-1}$
Service rate of a task on the VM in Module I	$\mu_1 = 0.20\ \text{ms}^{-1}$
Energy consumption per unit time of a busy VM in Module II	$\omega = 0.50$ mJ
Energy consumption per unit time of a sleeping VM in Module II	$\omega_s = 0.10$ mJ
Additional energy consumption for each migration	$\omega_m = 0.20$ mJ
Additional energy consumption for each listening	$\omega_l = 0.15$ mJ
Additional energy consumption for each wakeup	$\omega_u = 0.20$ mJ

BUSYLOW and BUSYHIGH to record the state of a VM. The necessary and sufficient condition for the system being stable is $\rho < 1$. We analyze the system model and evaluate the system performance under the condition that $\rho < 1$. To compare our proposed strategy with the existing VM allocation strategies, we set parameters in numerical experiments by referencing [14]. The parameter settings in the numerical experiments are shown in Table 1.

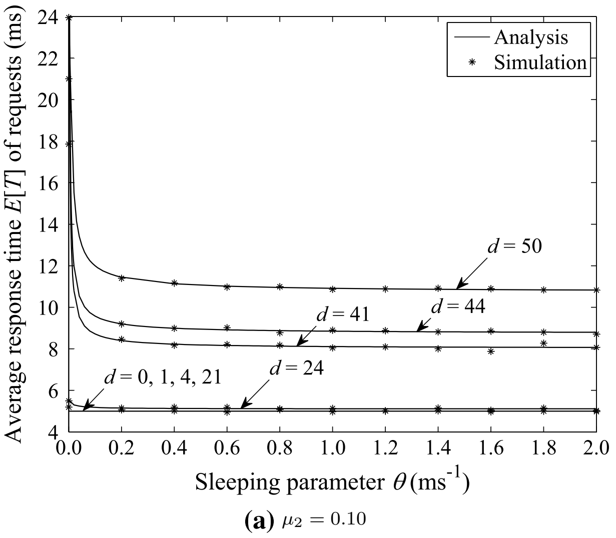
We note that, with different parameter settings, as long as the system is stable, the trend for all the performance measures will not change much.

Figure 2 examines the influence of the sleeping parameter θ on the average response time $E[T]$ of the tasks for the different number d of VMs in Module II.

From Fig. 2, we observe that if there are less VMs in Module II (such as $d < 24$), the average response time $E[T]$ of tasks remains nearly constant across all the values of the sleeping parameter θ. For this case, the capability of the VMs in Module I is strong enough to process all the arriving tasks, and there are no tasks waiting in the system buffer. As a result, it is likely that the VMs in Module II keep sleeping. So, the average response time of tasks is approximately the average service time (μ_1^{-1}) of tasks processed in Module I.

From Fig. 2, we also observe that if there are more VMs in Module II (such as $d = 24, 41, 44, 50$), the average response time $E[T]$ of tasks initially decreases sharply from a high value, then decreases slightly before finally converging to a certain value as the sleeping parameter θ increases. For this case, the processing capability of the VMs in Module I is insufficient to cope with the existing traffic load, so some arriving tasks have to wait in the system buffer. As a result, the VMs in Module II are more likely to be awake after a sleep period and process the tasks waiting in the system buffer. The influence of the sleeping parameter on the average response time of tasks is discussed as follows.

When the sleeping parameter θ is relatively small (such as $0 < \theta < 0.4$ for $d = 41$), the tasks arriving in the sleep period will have to wait longer in the system buffer. This results in a higher average response time of tasks. For this case, the influence on the average response time of tasks exerted by the sleeping parameter is greater than that exerted by the arrival rate of tasks and the service rate of tasks.



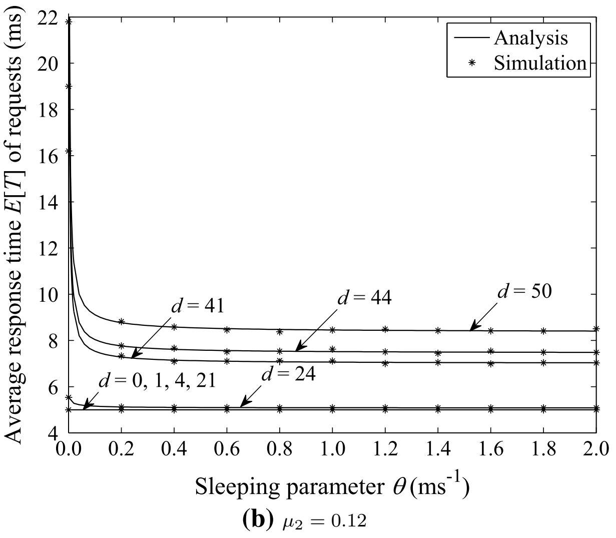
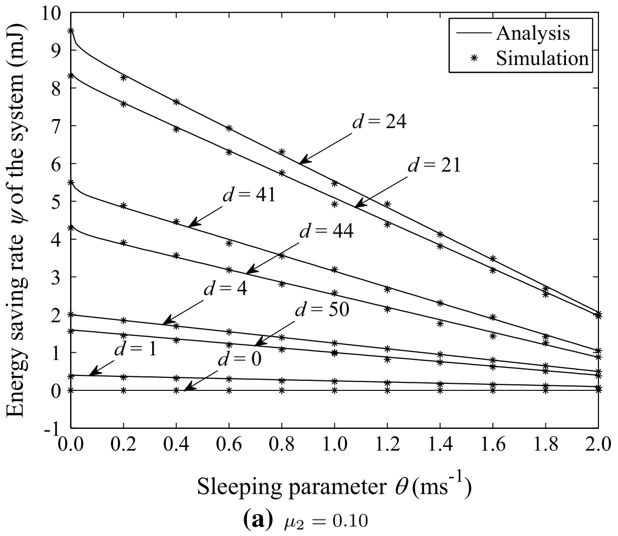


Fig. 2 Average response time $E[T]$ of tasks versus sleeping parameter θ

Consequently, the average response time of tasks will decrease sharply as the sleeping parameter increases.

When the sleeping parameter θ becomes larger (such as $0.4 < \theta < 2.0$ for $d = 41$), the tasks arriving during a sleep period will be processed earlier. This results in a lower average response time of tasks. For this case, the arrival rate of tasks and the service rate of tasks are the dominate factors influencing the average response time of tasks. Consequently, there is only a slight decreasing trend in the average response time of tasks in respect to the sleeping parameter.



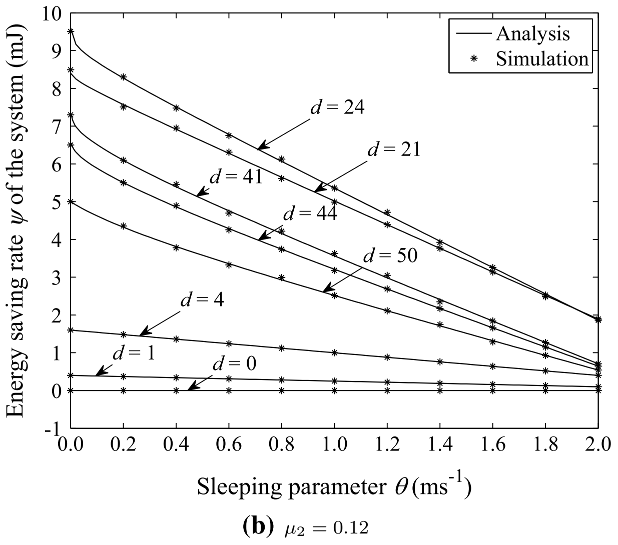


Fig. 3 Energy saving rate ψ of the system versus sleeping parameter θ

From Fig. 2, we also notice that for the same sleeping parameter θ, the average response time $E[T]$ of tasks will increase as the number d of VMs in Module II increases. As the number of VMs in Module II increases, and the system capability becomes weaker, and tasks will sojourn longer in the system. This will inevitably increase the average response time of tasks.

Comparing the results in Fig. 2a, b, we find that for the same number d of VMs in Module II and the same sleeping parameter θ, a larger service rate μ_2 of a task on the VM in Module II leads a lower average response time $E[T]$ of tasks. This is because the fact that the larger the service rate of a task on the VM in Module II is, the more quickly the VMs in Module II will process the tasks, and the fewer tasks will wait in the buffer. Therefore, the average response time of potential users will be lower.

Figure 3 examines the influence of the sleeping parameter θ on the energy saving rate ψ of the system for the different number d of VMs in Module II.

From Fig. 3, we observe that for the same number d of VMs in Module II, the energy saving rate ψ of the system decreases as the sleeping parameter θ increases. The larger the sleeping parameter is, the more frequently the VM in Module II listens to the system buffer and consumes additional energy. Therefore, the energy saving rate of the system will decrease.

From Fig. 3, we also notice that for the same sleeping parameter θ, either too few or too many VMs being deployed in Module II will lead to a lower energy saving rate ψ of the system. When the number of VMs in Module II is very small (such as $d = 0, 1, 4$), less energy can be saved even though all the VMs in Module II are sleeping. This results in a lower energy saving rate of the system. When the number of VMs in Module II is very large (such as $d = 41, 44, 50$), the system capability gets weaker. There is hardly any chance for the VMs in Module II to go to sleep. This results in a lower energy saving rate of the system.

Comparing the results shown in Fig. 3a, b, we find that the different numbers d of the VMs in Module II and the different service rates μ_2 of a task on the VM in Module II have different influence on the energy saving rate ψ of the system.

When less VMs are deployed in Module II (such as $d = 4$ for $\theta = 0.2$), the service capability of Module I is strong enough to process most of the arriving tasks, therefore only a few VMs in Module II will wake up and process other remaining tasks. In this case, the energy saving rate of the system mainly depends on the service rate of a task on the awake VM in Module II. As the service rate of a task on the VM in Module II increases, the VMs in Module II will consume more energy. This results in a lower energy saving rate of the system.

When more VMs are deployed in Module II (such as $d = 41, 44, 50$ for $\theta = 0.2$), the service capability of Module I is weaker, therefore more VMs in Module II have to wake up and process the arriving tasks. In this case, the number of sleeping VMs in Module II is the dominant factor to influence the energy saving rate of the system. The larger the service rate of a task on the VM in Module II is, the more quickly the VMs in Module II will process the tasks, and the more VMs in Module II will go to sleep, therefore more energy will be saved. This results the energy saving rate of the system to be higher.

From the discussions above, we foresee that when deploying the VMs in Module II and setting the sleeping parameter, we need to consider the service rate of a task on the VM in Module II.

In Figs. 2 and 3, the experiment results with $d = 0$ are for the conventional strategy where all the VMs always stay awake. The experiment results with $d = 50$ are for the conventional strategy where all the VMs are under an asynchronous multi-sleep mode. Compared to the conventional strategy where all the VMs always stay awake, our proposed strategy results in greater energy consumption without significantly affecting the response performance. Compared to the conventional strategy where all the VMs are under an asynchronous multi-sleep mode, our proposed strategy performs better in guaranteeing the response performance at the cost of occasional degradation in energy saving effect.

Comparing the results shown in Figs. 2 and 3, we find that a larger sleeping parameter leads to not only a shorter average response time of tasks but also a lower energy saving rate of the system, while a smaller sleeping parameter leads to not only a higher energy saving rate of the system but also a longer average response time of tasks. We also find that the energy saving rate of the system is higher with a moderate number of VMs in Module II, while the average response time of tasks is lower with a smaller number of VMs in Module II. Therefore, a trade-off between the average response time of tasks and the energy saving rate of the system should be aimed for when setting the number of VMs in Module II and the sleeping parameter in our proposed VM allocation strategy.

7 Performance Optimization

By trading off the average response time of tasks against the energy saving rate of the system, we establish a system cost function $F(d, \theta)$ as follows:

$$F(d, \theta) = f_1 E[T] - f_2 \psi \tag{15}$$

where f_1 and f_2 are treated as the impact factors for the average response time $E[T]$ of tasks and the energy saving rate ψ of the system on the system cost function.

We note that the mathematical expressions for the average response time $E[T]$ of tasks and the energy saving rate ψ of the system are difficult to express in closed-forms. The monotonicity of the system cost function is uncertain. In order to jointly optimize the number of the VMs in Module II and the sleeping parameter with the minimum system cost function, we turn to the Particle Swarm Optimization (PSO) intelligent searching algorithm.

Compared with other intelligent optimization algorithms, the PSO algorithm is simple to implement, and there are not many parameters to be adjusted [22, 23]. However, the traditional PSO algorithm has the disadvantage of premature convergence and easily falling into a local extreme. For this, in this paper, we turn to a PSO algorithm with a chaotic mapping mechanism and a nonlinear decreasing inertia weight to optimize the number of the VMs in Module II and the sleeping parameter together.

The main steps to jointly optimize the number of the VMs in Module II and the sleeping parameter are given in Table 2.

In Table 2, we use the system parameters given in Table 1, and set $f_1 = 4$, $f_2 = 1$, $N = 100$, $iter_{max} = 200$, $c_1 = 1.4962$, $c_2 = 1.4962$, $w_{max} = 0.95$, $w_{min} = 0.40$, $Ub = 2$, $Lb = 0$ and $X = 50$. For different service rates μ_2, we obtain the optimal combination (d^*, θ^*) for the number of VMs in Module II and the sleeping parameter with the minimum system cost function F^* in Table 3.

The optimization results in Table 3 depend on the arrival intensity of tasks, the serving capability of VMs and the cloud capacity. By substituting the arrival rate λ, the service rate μ_2 of a task on the VM in Module II, and the total number $(c + d)$

Table 2 Main steps to obtain optimal combination (d^*, θ^*)

```
Step 1: Initialize the number $N$ of particles, the maximum number of iterations $iter_{max}$
        for each particle's position, the cognitive acceleration coefficients $c_1$, the social
        acceleration coefficients $c_2$, the maximal inertia weight $w_{max}$, the minimal inertia
        weight $w_{min}$, the upper search boundary $Ub$, the lower search boundary $Lb$, the
        number $X$ of total VMs, and set the initial number of VMs in Module II as $d = 0$.
Step 2: Set the optimal combination $(d^*, \theta^*)$ for the number of VMs in Module II and
        the sleeping parameter, and calculate the corresponding fitness $F^*$:
            $(d^*, \theta^*) = (0, Ub),\ F^* = F(d^*, \theta^*) = f_1 E[T] - f_2 \psi$
Step 3: Initialize position $\theta_i$ for the $i$th$(i = 1, 2, \dots, N)$ particle by using a chaotic equation:
            $\theta_1 = rand$
            % $rand$ returns a sample in the interval $(0, 1)$ from the uniform distribution. %
            for $i = 2 : N$
              if $\theta_{i-1} < 0.50$
                $\theta_i = 1.50 * \theta_{i-1} + 0.25$
              else
                $\theta_i = 0.50 * \theta_{i-1} - 0.25$
              endif
            endfor
            for $i = 1 : N$
              $\theta_i = Lb + ((\ln(\theta_i + 0.50) + \ln 2)/\ln 3) * (Ub - Lb)$
            endfor
Step 4: For each particle $(i = 1, 2, \dots, N)$, initialize the personal best position $pb_i$ and
        the velocity $v_i$, and calculate the fitness $F_i$:
            $pb_i = \theta_i,\ v_i = randn,\ F_i = F(d, pb_i) = f_1 E[T] - f_2 \psi$
            % $randn$ returns a sample from the standard normal distribution. %
Step 5: Select the global best position $gb$ among all the personal best positions with
        the number $d$ of VMs in module II:
            $gb = \underset{i \in \{1, \dots, N\}}{argmin}\ F_i$
Step 6: Set the initial number of iterations as $iter = 1$.
Step 7: Update the inertia weight with the strategy of nonlinear decreasing inertia weight:
            $w = (w_{max} - w_{min})(iter/iter_{max})^2 + (w_{min} - w_{max})(2 * iter/iter_{max}) + w_{max}$
Step 8: For each particle $(i = 1, 2, \dots, N)$, update the position $\theta_i$, the velocity $v_i$, the
        fitness $F_i$ and the personal best position $pb_i$:
            $v_i = w * v_i + c_1 * rand * (pb_i - \theta_i) + c_2 * rand * (gb - \theta_i),\ \theta_i = \theta_i + v_i$
            $F'_i = F(d, \theta_i) = f_1 E[T] - f_2 \psi$
            if $F'_i < F_i$
              $F_i = F'_i,\ pb_i = \theta_i$
            endif
Step 9: Select the global best position $gb$ among all the personal best positions with
        the number $d$ of VMs in module II:
            $gb = \underset{i \in \{1, \dots, N\}}{argmin}\ F_i$
Step 10: Check the number of iterations:
            if $iter < iter_{max}$
              $iter = iter + 1$, go to Step 7
            endif
Step 11: Select the optimal combination $(d^*, \theta^*)$ for the number of VMs in Module II
         and the sleeping parameter:
            $F' = F(d, gb) = f_1 E[T] - f_2 \psi$
            if $F' < F^*$
              $F^* = F',\ (d^*, \theta^*) = (d, gb)$
            endif
Step 12: Check the number of VMs in Module II:
            if $d < X$
              $d = d + 1$, go to Step 3
            endif
Step 13: Output optimal combination $(d^*, \theta^*)$ and the minimum system cost function $F^*$.
```

Table 3 Optimization results: (d^*, θ^*) and F^*

Service rate μ_2	Optimal combination (d^*, θ^*)	Minimum system cost function F^*
0.05	(22, 0.0001)	11.9046
0.10	(23, 0.0330)	11.8698
0.15	(25, 0.0816)	11.6391
0.20	(49, 0.1593)	11.0410

of VMs in a virtual cluster, etc. into the algorithm in Table 2, the optimal parameter combination (d^*, θ^*) for the number of VMs in Module II and the sleeping parameter can be obtained for the proposed strategy.

8 Conclusions

In this paper, with the aim of reducing energy consumption and achieving greener computing, we proposed a novel energy-efficient Virtual Machine (VM) allocation strategy. Considering an asynchronous multi-sleep mode and an adaptive task-migration scheme with the proposed strategy, we established a type of queueing model with partial asynchronous multiple vacations, and derived the steady-state distribution of the system model. The queueing model quantified the effects of the number of VMs in Module II and the sleeping parameter. These effects were measured by two performance measures: the average response time of tasks and the energy saving rate of the system. Experimental results showed that the energy saving rate of the system is higher with a moderate number of VMs in Module II, while the average response time of tasks is lower with a smaller number of VMs in Module II. Accordingly, we built a system cost function to investigate a trade-off between different performance measures. By using a PSO algorithm with a chaotic mapping mechanism and a nonlinear decreasing inertia weight, we jointly optimized the number of VMs in Module II and the sleeping parameter with the minimum system cost function. In our future work, we would investigate a VM allocation strategy with (N, T) policy to trade off the average response time of tasks and the energy saving rate of the system, and build a four-dimensional Markov chain to have insight into the proposed strategy by considering the migration time and the wakeup time. Moreover, we would introduce a more versatile stochastic process, such as Markov Modulated Poisson Process (MMPP) or Interrupted Poisson Process (IPP), to model the task arrivals, and use a real-world dataset to enhance the contribution of our research.

Acknowledgements This work was supported in part by National Science Foundations (Nos. 61872311, 61472342) and Natural Science Foundation of Hebei Province (F2017203141), China, and was supported in part by MEXT, Japan.

References

1. Hintemann, R., Clausen, J.: Green cloud? The current and future development of energy consumption by data centers, networks and end-user devices. In: Proceedings of the 4th International Conference on ICT for Sustainability (ICT4S 2016), pp. 109–115 (2016)
2. Jin, X., Zhang, F., Vasilakos, A., Liu, Z.: Green data centers: A survey, perspectives, and future directions (2016). https://arxiv.org/pdf/1608.00687v1.pdf. Accessed 9 Dec 2017
3. Singh, S., Chana, I.: Resource provisioning and scheduling in clouds: QoS perspective. J. Supercomput. **72**(3), 926–960 (2016)
4. Haddar, I., Raouyane, B., Bellafkih, M.: Generating a service broker framework for service selection and SLA-based provisioning within network environments. In: Proceedings of the 9th International Conference on Ubiquitous and Future Networks (ICUFN 2017), pp. 630–635 (2017)
5. Nakamura, L., Azevedo, L., Batista, B., Meneguette, R., Toledo, C., Estrella, J.: An analysis of optimization algorithms designed to fully comply with SLA in cloud computing. IEEE Latin Am. Trans. **15**(8), 1497–1505 (2017)
6. Hasan, S., Kouki, Y., Ledoux, T., Pazat, J.: Exploiting renewable sources: when green SLA becomes a possible reality in cloud computing. IEEE Trans. Cloud Comput. **5**(2), 249–262 (2017)
7. Arianyan, E., Taheri, H., Khoshdel, V.: Novel fuzzy multi objective DVFS-aware consolidation heuristics for energy and SLA efficient resource management in cloud data centers. J. Netw. Comput. Appl. **78**, 43–61 (2017)
8. Son, J., Dastjerdi, A., Calheiros, R., Buyya, R.: SLA-aware and energy-efficient dynamic overbooking in SDN-based cloud data centers. IEEE Trans. Sustain. Comput. **2**(2), 76–89 (2017)
9. Hosseinimotlagh, S., Khunjush, F., Samadzadeh, R.: SEATS: smart energy-aware task scheduling in real-time cloud computing. J. Supercomput. **71**(1), 45–66 (2015)
10. Luo, J., Zhang, S., Yin, L., Guo, Y.: Dynamic flow scheduling for power optimization of data center networks. In: Proceedings of the 5th International Conference on Advanced Cloud and Big Data (CBD 2017), pp. 57–62 (2017)
11. Duan, L., Zhan, D., Hohnerlein, J.: Optimizing cloud data center energy efficiency via dynamic prediction of CPU idle intervals. In: Proceedings of the 8th IEEE International Conference on Cloud Computing (IEEE CLOUD 2015), pp. 985–988 (2015)
12. Sarji, I., Ghali, C., Chehab, A., Kayssi, A.: CloudESE: Energy efficiency model for cloud computing environments. In: Proceedings of the 2011 International Conference on Energy Aware Computing (ICEAC 2011), pp. 1–6 (2011)
13. Liu, Y., Zhu, H., Lu, K., Wang, X.: Self-adaptive management of the sleep depths of idle nodes in large scale systems to balance between energy consumption and response times. In: Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2012), pp. 633–639 (2012)
14. Jin, S., Hao, S., Yue, W.: Energy-efficient strategy with a speed switch and a multiple-sleep mode in cloud data centers. In: Proceedings of the 12th International Conference on Queueing Theory and Network Applications (QTNA2017), pp. 143–154 (2017)
15. Jin, S., Hao, S., Wang, B.: Virtual machine scheduling strategy based on dual-speed and work vacation mode and its parameter optimization. J. Commun. **38**(12), 10–20 (2017). (in Chinese)
16. Cao, H., Xu, J., Ke, D., Jin, C., Deng, S., Tang, C., Cui, M., Liu, J.: Economic dispatch of micro-grid based on improved particle-swarm optimization algorithm (2016). https://doi.org/10.1109/NAPS.2016.7747875
17. Zhang, Y., Zhao, Y., Fu, X., Xu, J.: A feature extraction method of the particle swarm optimization algorithm based on adaptive inertia weight and chaos optimization for Brillouin scattering spectra. Opt. Commun. **376**, 56–66 (2016)
18. Tian, D.: Particle swarm optimization with chaos-based initialization for numerical optimization (2016). https://doi.org/10.1080/10798587.2017.1293881
19. Paxson, V., Floyd, S.: Wide-area traffic: the failure of Poisson modeling. IEEE/ACM Trans. Netw. **3**(3), 226–244 (1995)
20. Tian, N., Zhang, Z.: Vacation Queueing Models: Theory and Applications. Springer, New York (2006)
21. Jiang, M., Hu, J., Zhao, R., Wei, X., Nie, Z.: Hybrid IE-DDM-MLFMA with Gauss–Seidel iterative technique for scattering from conducting body of translation. Appl. Comput. Electromagn. Soc. J. **30**(2), 148–156 (2015)

22. Rahmat-Samii, Y., Gies, D., Robinson, J.: Particle swarm optimization (PSO): a novel paradigm for antenna designs. Ursi Radio Sci. Bull. **76**(3), 14–22 (2017)
23. Guedria, N.: Improved accelerated PSO algorithm for mechanical engineering optimization problems. Appl. Soft Comput. **40**, 455–467 (2016)

Xiuchen Qie received the B.Eng. degree in Computer Science and Technology from Harbin Normal University, Harbin, China. Now Miss. Qie is a postgraduate student at School of Information Science and Engineering, Yanshan University, Qinhuangdao, China. Xiuchen Qie's reserch interests include cloud computing, virtual machine management and stochastic modeling.

Shunfu Jin received the B.Eng. degree in Computer Science from North-East Heavier Machinery College, China, M.Eng. degree in Computer Science from Yanshan University, China, and Dr.Eng degree in Circuits and System from Yanshan University. Now she is a professor at School of Information Science and Engineering, Yanshan University, China. Dr. Jin's research interests include stochastic modeling for telecommunication, performance evaluation for computer system and network and application for queueing system.

Wuyi Yue received the B.Eng. degree in Electronic Engineering from Tsinghua University, China, the M.Eng. and Dr.Eng. degrees in Applied Mathematics and Physics from Kyoto University, Japan. Dr. Yue is currently a professor at the Faculty of Intelligence and Informatics, and Dean of the Graduate School of Natural Science, Konan University, Japan. Dr. Yue is a fellow of the ORSJ, a senior number of IEICE, Japan and a member of the IEEE. Dr. Yue's research interests include queueing theory, stochastic processes and optimal methods as well applied to system modeling, performance analysis and optimal resource allocation of communication networks, systems engineering, and operations research. Dr. Yue is an author (a co-author) and a co-editor of more than 10 monographs published by Springer and other Publishers, and more than 300 refereed papers published in journals such as IEEE Transactions and IEICE Transactions, and in proceedings of IEEE, ACM, LNCS, Springer.