Annals of Operations Research (2020) 293:193–212
https://doi.org/10.1007/s10479-019-03339-3

S.I.: QUEUEING THEORY AND NETWORK APPLICATIONS II

A clustered virtual machine allocation strategy based on a sleep-mode with wake-up threshold in a cloud environment

Shunfu Jin[1] · Xiuchen Qie[1] · Wenjuan Zhao[1] · Wuyi Yue[2] · Yutaka Takahashi[3]

Published online: 23 July 2019

Abstract
The massive amount of energy consumed by cloud data centers is detrimentally impacting on the environments. As such, to work towards "greener" computing, in this paper, we propose a clustered virtual machine (VM) allocation strategy based on a sleep-mode with a wake-up threshold. The VMs in a cloud data center are clustered into two pools, namely, Pool I and Pool II. The VMs in Pool I remain awake at all times, while the VMs in Pool II go to sleep under a light workload. After a sleep timer expires, the corresponding VM will resume processing tasks only if the number of waiting tasks reaches the wake-up threshold. Otherwise, the sleeping VM will remain asleep as a new sleep timer starts. By establishing a queue with an N-policy and asynchronous vacations of partial servers, we capture the stochastic behavior of tasks with the proposed strategy, and derive the performance measures in terms of the average latency of tasks and the energy saving rate of the system. Furthermore, we provide numerical results to demonstrate the impact of the system parameters on the system performance. Finally, we construct a system cost function to trade off different performance measures, and develop an intelligent searching algorithm to jointly optimize the number of the VMs in Pool II, the wake-up threshold and the sleeping parameter.

Keywords Cloud data center · Clustered VM allocation · Wake-up threshold · Sleep-mode · Average latency · Energy saving rate

1 Introduction

The energy consumption in cloud data centers has been steadily increasing over the last few years (Zhou et al. 2018). As a result, the minimization of power and energy consumption in a cloud environment has become an urgent problem that is now receiving significant attention (Marek and Hoon 2018).

This work was supported in part by National Natural Science Foundation (Nos. 61872311, 61472342) and Hebei Province Natural Science Foundation (No. F2017203141), China, and was supported in part by MEXT and JSPS KAKENHI Grant (No. JP17H01825), Japan.

✉ Shunfu Jin
jsf@ysu.edu.cn

Extended author information available on the last page of the article

Applying a sleep mode is an effective method for reducing energy consumption in cloud data centers (Khojandi et al. 2018). For the purpose of minimizing energy consumption during runtime, Duan et al. (2015) presented a dynamic idle interval prediction scheme to estimate the future idle interval length of a CPU and thereby choose the most cost-effective sleep state. Chou et al. (2016) proposed a fine-grain power management scheme for data center workloads. This proposed scheme could dynamically postpone the processing of some tasks, create longer idle periods and promote the use of a deeper sleep mode. Luo et al. (2017) based on flow preemption and power-aware routing, Luo et al. proposed a dynamic adaptive scheduling algorithm to reduce energy consumption by decreasing the ratio of low utilization devices and putting more devices into sleep mode. In the literature mentioned above, we note that a sleep mode was only applied to a physical machine (PM) rather than a virtual machine (VM). This weakens the application flexibility of sleep modes.

Queuing models with vacations are used as an important way to evaluate the performance measures of the cloud data centers with sleep modes (Do et al. 2018). Multiple-server queueing models with various types of vacations have been studied by many researchers. Jin et al. (2018) proposed a novel type of M/M/c queue with an N-policy and synchronous vacations. Tian et al. (2001) analyzed the equilibrium theory for an M/M/c queue with an N-policy and asynchronous vacations. Jin et al. (2017) proposed a queueing system with synchronous vacations of partial servers. The research mentioned above has contributed to enriching the queueing system with an N-policy and synchronous vacations, an N-policy and asynchronous vacations, as well as asynchronous vacations of partial servers. However, there has so far been no research on a queueing system with an N-policy and asynchronous vacations of partial servers.

Teaching–learning-based optimization (TLBO) is an effective tool for function optimization (Cao et al. 2016). Since the appearance of TLBO, many researchers have directed their efforts into improving the searching capability of TLBO algorithms. Yu et al. (2016) an effort to enhance the performance of TLBO algorithms, Yu et al. introduced the mutation crossover operation to increase population diversity, and applied the chaotic perturbation mechanism to make the TLBO algorithm avoid falling into the local optimum. Ji et al. (2017) inspired by the concept of historical population, Ji et al. added two new process, namely the self-feedback learning process as well as the mutation and crossover processes to the TLBO algorithm. The added processes improved the exploration ability of the new algorithm in comparison to the original TLBO algorithm. Li et al. (2017) in order to enhance the diversification of population, Li et al. increased the number of teachers, introduced new students, and performed local searches around the potentially optimal solutions. To the best of our knowledge, the application of the improved TLBO algorithm in the optimization of the VM allocation strategy has not been studied yet.

Inspired by the research mentioned above, in this paper, we propose a clustered VM allocation strategy based on a sleep-mode with a wake-up threshold. For one thing, allowing that all the VMs in a cloud data center go to sleep may degrade the quality of cloud service. Thus, we divide the VMs in a cloud data center into two pools: Pool I and Pool II. The VMs in Pool I always stay awake to provide a timely cloud service for processing tasks, whereas the VMs in Pool II independently go to sleep whenever possible to reduce energy consumption. Another, for a VM, frequently switching from a sleep state to a awake state may cause additional energy consumption. To get around this problem, in this paper we make the VMs in Pool II to wake up only if the number of tasks waiting in the system buffer reaches a wake-up threshold and the corresponding sleep timer expires. This wake-up mechanism improves the energy conservation by delaying the instant at which a VM wakes up. To analyze the proposed strategy, we first build a type of queue with an N-policy and asynchronous vacations of partial servers by using a matrix-geometric method, and investigate the system performance through analysis experiments and simulation experiments. Finally, for the purpose of optimizing the proposed strategy, we construct a cost function to trad off different system performance, and develop the TLBO algorithm to optimize the system parameters settings.

The rest of this paper is organized as follows. In Sect. 2, we propose a clustered VM allocation strategy based on a new type of sleep-mode in a cloud environment and build a system model accordingly. In Sect. 3, we analyze the system model using the method of a matrix-geometric solution. In Sect. 4, we derive the performance measures in terms of the average latency of tasks and the energy saving rate of the system with the proposed strategy. In Sect. 5, we demonstrate the influence of system parameters on the system performance by using numerical results. In Sect. 6, we develop an intelligent searching algorithm to jointly optimize the number of the VMs in Pool II, the wake-up threshold and the sleeping parameter. Finally, Sect. 7 is devoted to the conclusion.

2 VM allocation strategy and system model

In this section, in order to enhance the energy efficiency in a cloud environment, we firstly propose a clustered VM allocation strategy based on a sleep-mode with a wake-up threshold. And then, we establish a queue with an N-policy and asynchronous vacations of partial servers to capture the proposed strategy.

2.1 VM allocation strategy

It should be noted that additional energy will be consumed when a VM frequently switches from the sleep state to the awake state, while the system performance will be degraded when all the VMs are put into a sleep-mode. To get around these problems, based on a sleep-mode with a wake-up threshold, we propose a clustered VM allocation strategy.

The VMs in a cloud data center are clustered into two pools, namely, Pool I and Pool II. The VMs in Pool I remain awake all the time and operate on a higher speed. The VMs in Pool II will go to sleep independently when there are no tasks in the system buffer. For the purpose of improving the energy efficiency of cloud data centers, we introduce a wake-up threshold to a periodic sleep mode. At the end epoch of a sleep period, if the number of the tasks gathered in the system buffer reaches or exceeds a certain value, namely the wake-up threshold, the corresponding VM in Pool II will wake up independently and operate on a lower speed. Otherwise, the VM in Pool II will begin another sleep period with a new sleep timer.

Under the proposed strategy, all the VMs are dominated by a control server, where several sleep timers, a task counter, and a VM scheduler are deployed. When a VM in Pool II is scheduled to go to sleep from the awake state or begin another sleep period, a sleep timer with a random value will be started to control the time length of a sleep period. The task counter keeps working and counts the tasks waiting in the system buffer. At the moment that a sleep timer expires, the VM scheduler adjusts the state of the corresponding VM according to the value on the task counter.

Figure 1 illustrates the working flow of a VM with the clustered VM allocation strategy based on a sleep-mode with a wake-up threshold.

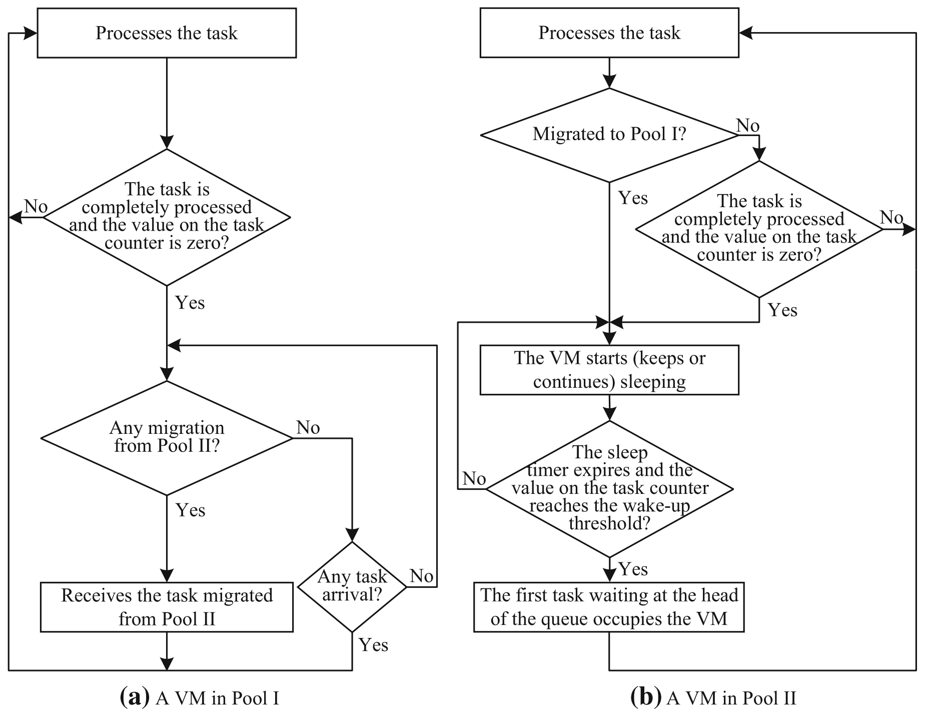

Processes the task
The task is completely processed and the value on the task counter is zero?
No
Yes
Any migration from Pool II?
No
Yes
Receives the task migrated from Pool II
Any task arrival?
No
Yes
(a) A VM in Pool I
Processes the task
Migrated to Pool I?
No
Yes
The task is completely processed and the value on the task counter is zero?
No
Yes
The VM starts (keeps or continues) sleeping
The sleep timer expires and the value on the task counter reaches the wake-up threshold?
No
Yes
The first task waiting at the head of the queue occupies the VM
(b) A VM in Pool II

Fig. 1 The working flow of a VM with the strategy proposed in this paper

As shown in Fig. 1, the VMs in Pool I remain awake all the time, while the VMs in Pool II switch between the sleep state and the awake state. Next, we discuss the state transition for the VMs in Pool II.

1. Awake state to sleep state

For a busy VM in Pool II, the state transition from awake state to sleep state occurs only at the instant when a task either in Pool I or Pool II is completely processed. When a task is completely processed in Pool I, if the VM scheduler monitors that the value on the task counter is zero and there is at least one task being processed in Pool II, one of the tasks being processed in Pool II will migrate Pool I, and then the evacuated VM in Pool II will go to sleep. When a task is completely processed in Pool II, if the VM scheduler monitors that the value on the task counter is zero, the evacuated VM in Pool II will go to sleep directly. We note that the task-migration considered in this paper is a kind of online VM-migration between different pools within a cloud data center, and this task-migration is to make the VMs in Pool II go to sleep earlier.

2. Sleep state to awake state

For a sleeping VM in Pool II, the state transition from sleep state to awake state occurs only at the end epoch of a sleep period. When a sleep timer expires, if the VM scheduler monitors that the value on the task counter is equal to or greater than the wake-up threshold, the corresponding VM in Pool II will wake up to process the first task waiting in the system buffer on a lower speed. Otherwise, a new sleep timer will be started and the VM in Pool II will begin another sleep period.

To summarize, at the moment that a sleep timer expires, the VMs in Pool II will wake up only if the number of the tasks waiting in the system buffer reaches the wake-up threshold. The wake-up mechanism in our proposed strategy effectively improves the energy conservation via extending sleep time by delaying the instant at which a VM wakes up. This is exactly the starting point for the sleep-mode with a wake-up threshold within our proposed strategy.

2.2 System model

According to the proposed VM allocation, we establish a queue with an N-policy and asynchronous vacations of partial servers. In this system model, the tasks are regarded as customers, each VM is regarded as an independent server, a sleep period is regarded as a vacation, and multiple sleep periods are regarded as multiple vacations.

In this system model, the buffer capacity is supposed to be infinite. Moreover, the number of the VMs in Pool I and Pool II are denoted as c and d, respectively. Let i_t be the total number of tasks in the system at instant t. i_t is called the system level, $i_t \geqslant 0$. Let j_t be the number of busy VMs in Pool II at instant t. j_t is called the system stage, $j_t = \overline{0, d}$, where $\overline{0, d}$ represents the range of integers from 0 to d.

Thus, the behavior of the system under consideration can be described in terms of the two-dimensional continuous-time stochastic process as follows:

$$\xi_t = \{i_t, j_t\}, \quad t \geqslant 0. \tag{1}$$

The state-space $\boldsymbol{\Omega}$ of the two-dimensional continuous-time stochastic process ξ_t is given as follows:

$$\begin{aligned}\boldsymbol{\Omega} = &\{(i, 0) : i = \overline{0, c}\} \cup \{(i, j) : i = \overline{c+1, c+d}, \quad j = \overline{0, i-c}\} \\ &\cup \{(i, j) : i \geqslant c+d+1, \quad j = \overline{0, d}\}.\end{aligned} \tag{2}$$

We assume that all the tasks in the system are homogenous and all the VMs in a pool are identical, where each task is for one VM. The arrival intervals of tasks are assumed to have an exponential distribution with parameter λ ($\lambda > 0$). In addition, the service times of tasks processed in Pool I and in Pool II are assumed to be exponentially distributed with parameters μ_1 ($\mu_1 > 0$) and μ_2 ($0 < \mu_2 < \mu_1$), respectively. Furthermore, the time length of a sleep timer is assumed to follow an exponential distribution with parameter θ ($\theta > 0$). Here, the parameter θ is called the sleeping parameter.

Based on the assumptions above, the two-dimensional continuous-time stochastic process ξ_t can be seen as a two-dimensional continuous-time Markov chain (CTMC).

Taking $c = 2$, $d = 3$ and $N = 2$ as an example, we show the state transition of the two-dimensional CTMC ξ_t in Fig. 2, where $N = 2$ shows this CTMC with a 2-policy.

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:

$$\pi_{i,j} = \lim_{t \to \infty} P\{i_t = i, j_t = j\}, \quad i \geqslant 0, \quad j = \overline{0, d}. \tag{3}$$

And then, we form the row vectors $\boldsymbol{\pi}_i$ as follows:

$$\boldsymbol{\pi}_i = \begin{cases} \pi_{i,0}, & i = \overline{0, c} \\ (\pi_{i,0}, \pi_{i,1}, \ldots, \pi_{i,i-c}), & i = \overline{c+1, c+d} \\ (\pi_{i,0}, \pi_{i,1}, \ldots, \pi_{i,d}), & i \geqslant c+d+1. \end{cases} \tag{4}$$

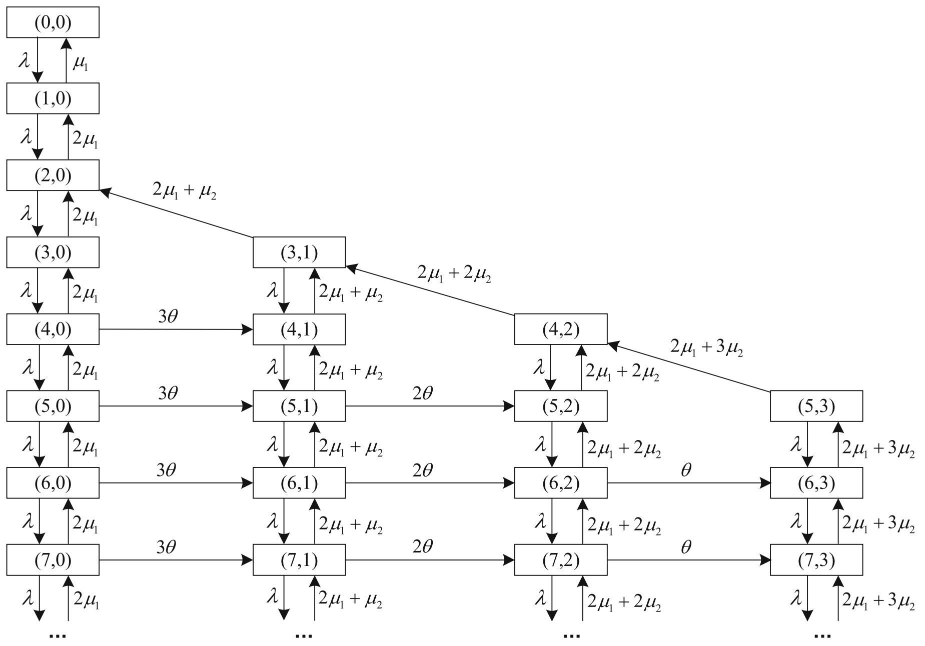


Fig. 2 The state transition of the the two-dimensional CTMC ξ_t with $c = 2,\ d = 3$ and $N = 2$

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

$$\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots). \tag{5}$$

3 Model analysis

In this section, we first investigate the transition rate matrix of the two-dimensional CTMC. And then, we derive the steady-state probability distribution of the system model.

3.1 Transition rate matrix

We denote $\boldsymbol{Q}$ as the one-step state transition rate matrix of the two-dimensional CTMC ξ_t. According to the system level, we separate the one-step state transition rate matrix $\boldsymbol{Q}$ of the two-dimensional CTMC ξ_t into some sub-matrices with different orders. To clearly represent the sub-matrices, we denote $\boldsymbol{Q}_{k,k'}$ as the one-step state transition rate sub-matrix for the system level changing from k $(k \geqslant 0)$ to k' $(k' \geqslant 0)$. Based on the initial system level k, we discuss the one-step state transition rate sub-matrix $\boldsymbol{Q}_{k,k'}$ in relation to three cases.

1. Initial system level is $k = \overline{0, c}$: the number of busy VMs in Pool I is k, while the number of busy VMs in Pool II is 0.

$k = 0$ means no tasks exist in the system at all. Therefore, the possible state transitions are from state (0, 0) to state (1, 0) and from state (0, 0) to state (0, 0). If there is a task arrival at the system, the system level will be increased by one, while the system stage will not change, and the transition rate from state $(0, 0)$ to state $(1, 0)$ is λ. If no task arrival occurs in the system, the system state will be fixed at state $(0, 0)$, and the transition rate is $-\lambda$. Consequently, the sub-matrices $\boldsymbol{Q}_{0,1}$ and $\boldsymbol{Q}_{0,0}$ are actually quantities given as follows:

$$\boldsymbol{Q}_{0,1} = \lambda, \quad \boldsymbol{Q}_{0,0} = -\lambda.$$

$k = \overline{1, c}$ means all the tasks in the system are receiving service from the VMs in Pool I. If one of the tasks is completely processed and departs the system, the system level will drop by one, while the system stage will not change, and the transition rate from state $(k, 0)$ to state $(k-1, 0)$ is $k\mu_1$. If there is a task arrival at the system, the system level will increase by one, while the system stage will not change, and the transition rate from state $(k, 0)$ to state $(k+1, 0)$ is λ. If neither an arrival nor a departure occurs, the system state will be fixed at state $(k, 0)$, the transition rate is $-(\lambda + k\mu_1)$. Consequently, the sub-matrices $\boldsymbol{Q}_{k,k-1}$, $\boldsymbol{Q}_{k,k+1}$ and $\boldsymbol{Q}_{k,k}$ are also actual quantities given as follows:

$$\boldsymbol{Q}_{k,k-1} = k\mu_1, \quad \boldsymbol{Q}_{k,k+1} = \lambda, \quad \boldsymbol{Q}_{k,k} = -(\lambda + k\mu_1), \quad k = \overline{1, c}.$$

2. Initial system level is $k = c + x,\ x = \overline{1, d-1}$: the number of busy VMs in Pool I is c, while the number of busy VMs in Pool II is less than or equal to x.

If there is at least one task queueing in the system buffer when a task is completely processed on a VM, whether the VM belongs to Pool I or Pool II, the task waiting at the head of the system buffer will be scheduled to the evacuated VM. In this case, the system level will drop by one, while the system stage will not change. Therefore, the transition rate from state (k, n) to state $(k-1, n)$ will be $(c\mu_1 + n\mu_2)$, where n $(n = \overline{0, x-1})$ is the number of busy VMs in Pool II.

If there are no tasks queueing in the system buffer when a task is completely processed on a VM, whether the VM belongs to Pool I or Pool II, the number of sleeping VMs will increase by one. If the evacuated VM belongs to Pool I, one of the tasks receiving service in Pool II will be migrated to the evacuated VM in Pool I, and the just-evacuated VM in Pool II will go to sleep. If the evacuated VM belongs to Pool II, the evacuated VM in Pool II will start sleeping directly. In both of these two cases, both the system level and the system stage will drop by one. Therefore, the transition rate from state (k, x) to state $(k-1, x-1)$ will be $(c\mu_1 + x\mu_2)$.

Consequently, the sub-matrix $\boldsymbol{Q}_{k,k-1}$ is a rectangular $(x+1) \times x$ matrix given as follows:

$$\boldsymbol{Q}_{k,k-1} = \begin{pmatrix} c\mu_1 & & & \\ & c\mu_1 + \mu_2 & & \\ & & \ddots & \\ & & & c\mu_1 + (x-1)\mu_2 \\ & & & c\mu_1 + x\mu_2 \end{pmatrix}, \quad k = c + x, \quad x = \overline{1, d-1}.$$

Before any of the sleep timers in Pool II expire, if there is a task arrival at the system, the newly arriving task has to queue in the system buffer. In this case, the system level will increase by one, while the system stage will not change. Therefore, the transition rate from state (k, n) to state $(k+1, n)$ will be λ. Consequently, the sub-matrix $\boldsymbol{Q}_{k,k+1}$ is a rectangular $(x+1) \times (x+2)$ matrix given as follows:

$$\boldsymbol{Q}_{k,k+1}=\begin{pmatrix}\lambda & & & & 0\\ & \lambda & & & 0\\ & & \ddots & & \vdots\\ & & & \lambda & 0\\ & & & & \lambda\ \ 0\end{pmatrix},\quad k=c+x,\quad x=\overline{1,d-1}.$$

At the moment that one of the sleep timers expires, the corresponding VM in Pool II will decide whether to wake up or continue sleeping according to the number of tasks queueing in the system buffer. If this number is equal to or greater than the wake-up threshold N, the corresponding VM in Pool II will wake up and prepare to process the task waiting at the head of the queue. In this case, the system level will not change, while the system stage will increase by one. Therefore, the transition rate from state (k,n) to state $(k,n+1)$ will be $(d-n)\theta$, where n $(n=\overline{0,x})$ is the number of busy VMs in Pool II. Otherwise, the corresponding VM in Pool II will continue sleeping and there will be no state transitions at all.

Before any of the sleep timers in Pool II expire, if neither an arrival nor a departure occurs, the system state will not change. When the number of tasks queueing in the system buffer is equal to or greater than the wake-up threshold N, the transition rate will be $(-h_n-(d-n)\theta)$, where $h_n=\lambda+c\mu_1+n\mu_2$. When the number of tasks queueing in the system buffer is less than the wake-up threshold N, the transition rate will be $-h_n$.

Consequently, the sub-matrix $\boldsymbol{Q}_{k,k}$ is a rectangular $(x+1)\times(x+1)$ matrix given as follows:

$$\boldsymbol{Q}_{k,k}=diag\,(-h_0,-h_1,\ldots,-h_x)\,,\quad k=c+x,\quad x=\overline{1,min\{N,d\}-1},$$

$$\boldsymbol{Q}_{k,k}=\begin{pmatrix}-h_0-d\theta & d\theta & & & & \\ \ddots & \ddots & & & & \\ & -h_{x-N}-(d-(x-N))\theta & (d-(x-N))\theta & & & \\ & & -h_{x-N+1} & 0 & & \\ & & \ddots & \ddots & & \\ & & & -h_{x-1} & 0 \\ & & & & -h_x\end{pmatrix},$$

$$k=c+x,\quad x=\overline{N,d-1}\ (N\leqslant d).$$

3. Initial system level is $k=c+x,\ x\geqslant d$: the number of busy VMs in Pool I is c, while the number of busy VMs in Pool II is less than or equal to d.

In the case of $k=c+d$, $\boldsymbol{Q}_{k,k-1}$ is a rectangular matrix of order $(d+1)\times d$. In the case of $k=c+x,\ x\geqslant d+1$, $\boldsymbol{Q}_{k,k-1}$ is square matrices of the order $(d+1)\times(d+1)$. In the case of $k=c+x,\ x\geqslant d$, $\boldsymbol{Q}_{k,k+1}$ and $\boldsymbol{Q}_{k,k}$ are all square matrices of the order $(d+1)\times(d+1)$. Referencing the discussions in Item (2), the sub-matrices $\boldsymbol{Q}_{k,k-1}$, $\boldsymbol{Q}_{k,k+1}$ and $\boldsymbol{Q}_{k,k}$ are given as follows:

$$\boldsymbol{Q}_{k,k-1} = \begin{pmatrix} c\mu_1 & & & & \\ & c\mu_1+\mu_2 & & & \\ & & \ddots & & \\ & & & c\mu_1+(d-1)\mu_2 & \\ & & & & c\mu_1+d\mu_2 \end{pmatrix}, \quad k = c+d,$$

$$\boldsymbol{Q}_{k,k-1} = \operatorname{diag}(c\mu_1, c\mu_1+\mu_2, \ldots, c\mu_1+d\mu_2), \quad k = c+x,\ x \geqslant d+1,$$

$$\boldsymbol{Q}_{k,k+1} = \operatorname{diag}(\lambda, \lambda, \ldots, \lambda), \quad k = c+x, \quad x \geqslant d,$$

$$\boldsymbol{Q}_{k,k} = \operatorname{diag}(-h_0, -h_1, \ldots, -h_d), \quad k = c+x, \quad x = \overline{d, N-1}\ (N > d),$$

$$\boldsymbol{Q}_{k,k} = \begin{pmatrix} -h_0-d\theta & & d\theta & & & & \\ & \ddots & & \ddots & & & \\ & & -h_{x-N}-(d-(x-N))\theta & (d-(x-N))\theta & & & \\ & & & -h_{x-N+1} & 0 & & \\ & & & & \ddots & \ddots & \\ & & & & & -h_{d-1} & 0 \\ & & & & & & -h_d \end{pmatrix},$$

$$k = c+x, x = \overline{\max\{N,d\}, N+d-1},$$

$$\boldsymbol{Q}_{k,k} = \begin{pmatrix} -h_0-d\theta & d\theta & & & \\ & -h_1-(d-1)\theta & (d-1)\theta & & \\ & & \ddots & \ddots & \\ & & & -h_{d-1}-\theta & \theta \\ & & & & -h_d \end{pmatrix},$$

$$k = c+x, \quad x \geqslant d+N.$$

So far, all the sub-matrices in the one step state transition rate matrix $\boldsymbol{Q}$ have been addressed. In the one-step state transition rate matrix $\boldsymbol{Q}$, the sub-matrices $\boldsymbol{Q}_{k,k-1}$ are repeated forever starting from the system level $(c+d+1)$, the sub-matrices $\boldsymbol{Q}_{k,k+1}$ are repeated forever starting from the system level $(c+d)$, and the sub-matrices $\boldsymbol{Q}_{k,k}$ are repeated forever starting from the system level $(c+d+N)$. For presentation purposes, the repetitive sub-matrices $\boldsymbol{Q}_{k,k-1}$, $\boldsymbol{Q}_{k,k+1}$ and $\boldsymbol{Q}_k$ are represented by $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{A}$, respectively. Then, the one step state transition rate matrix $\boldsymbol{Q}$ is written as follows:

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{Q}_{0,0} & \boldsymbol{Q}_{0,1} & & & & & & \\ \boldsymbol{Q}_{1,0} & \boldsymbol{Q}_{1,1} & \boldsymbol{Q}_{1,2} & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & \boldsymbol{Q}_{c+d,c+d-1} & \boldsymbol{Q}_{c+d,c+d} & \boldsymbol{C} & & & \\ & & & \boldsymbol{B} & \boldsymbol{Q}_{c+d+1,c+d+2} & \boldsymbol{C} & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \boldsymbol{B} & \boldsymbol{Q}_{c+d+N-1,c+d+N-1} & \boldsymbol{C} & \\ & & & & & \boldsymbol{B} & \boldsymbol{A} & \boldsymbol{C} \\ & & & & & \ddots & \ddots & \ddots \end{pmatrix}.$$

Obviously, the obtained form of the one step state transition rate matrix $\boldsymbol{Q}$ is a quasi-birth-and-death (QBD) matrix, so the CTMC ξ_t is also called a QBD CTMC (Tian and Zhang 2006).

Different methods can be used to resolve the steady-state transition probabilities. In our study, we consider a matrix-geometric solution method which is commonly used to analyze the QBD CTMC in the steady state.

3.2 Steady-state probability distribution

To analyze the QBD CTMC ξ_t by using the matrix-geometric solution method, we need to solve for the minimal non-negative solution of the matrix quadratic equation

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

and this solution, called the rate matrix and denoted by $\boldsymbol{R}$, can be explicitly determined.

In Sect. 3.1, the sub-matrices $\boldsymbol{B}$, $\boldsymbol{A}$ and $\boldsymbol{C}$ are deduced to be upper-triangular matrices. Thus, the rate matrix $\boldsymbol{R}$ is an upper-triangular matrix. We denote the unknown element of the rate matrix $\boldsymbol{R}$ in line u $(u = \overline{0,d})$ column v $(v = \overline{0,d})$ as $r_{u,v}$, and $r_u = r_{u,u}$ $(u = \overline{0,d})$. The rate matrix $\boldsymbol{R}$ can be written as follows:

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

And then, using the elements in $\boldsymbol{R}$, we calculate the elements of $\boldsymbol{R}^2$ as follows:

$$(\boldsymbol{R}^2)_{u,u} = r_u^2, \quad u = \overline{0,d}, \quad (\boldsymbol{R}^2)_{u,v} = \sum_{i=u}^{v} r_{u,i} r_{i,v}, \quad u = \overline{0,d-1}, \quad v = \overline{u+1,d}.$$

Next, by substituting the $\boldsymbol{R}^2$, $\boldsymbol{R}$, $\boldsymbol{B}$, $\boldsymbol{A}$ and $\boldsymbol{C}$ into Eq. (6), we obtain a set of equations:

$$\begin{cases} (c\mu_1 + u\mu_2) r_u^2 - (h_u + (d-u)\theta) r_u + \lambda = 0, \quad u = \overline{0,d} \\ (c\mu_1 + v\mu_2) \displaystyle\sum_{i=u}^{v} r_{u,i} r_{i,v} - (h_v + (d-v)\theta) r_{u,v} + (d-v+1)\theta r_{u,v-1} = 0, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad u = \overline{0,d-1}, \quad v = \overline{u+1,d}. \end{cases} \tag{8}$$

With the constraint that the traffic load $\rho = \lambda(c\mu_1 + d\mu_2)^{-1} < 1$, we prove that the first equation of Eq. (8) has two real roots r_u $(0 < r_u < 1)$ and r_u^* $(r_u^* \geqslant 1)$. It should be noted that the diagonal elements of $\boldsymbol{R}$ are r_u $(u = \overline{0,d})$.

It is too arduous to deduce a general expression of $r_{u,v}$ $(u = \overline{0,d-1}, v = \overline{u+1,d})$ in closed-form. So, based on the last equation of Eq. (8), we recursively compute the off-diagonal elements starting from the diagonal elements obtained in the first equation of Eq. (8).

By using the matrix-geometric solution method (Neuts 1981), we have

$$\boldsymbol{\pi}_i = \boldsymbol{\pi}_{c+d+N} \boldsymbol{R}^{i-(c+d+N)}, \; i \geqslant c+d+N, \tag{9}$$

and the unknown steady-state probability distribution $\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{c+d+N}$ can be obtained by solving the augmented matrix equation as follows:

$$(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{c+d+N}) \left(B[\boldsymbol{R}] \,\vdots\, \begin{matrix} \boldsymbol{e}_1 \\ (\boldsymbol{I}-\boldsymbol{R})^{-1}\boldsymbol{e}_2 \end{matrix} \right) = \left(\boldsymbol{0} \,\vdots\, 1 \right) \tag{10}$$

where $\boldsymbol{e}_1$ is a $\left(c + \left(\frac{d}{2} + N\right)(d+1)\right) \times 1$ vector, all of elements equal 1, $\boldsymbol{e}_2$ is a $(d+1) \times 1$ vector, all of elements equal 1, $\boldsymbol{0}$ is a $1\times \left(c + \left(\frac{d+2}{2} + N\right)(d+1)\right)$ vector, all of elements equal 0, and

$$B[\boldsymbol{R}] = \begin{pmatrix} \boldsymbol{Q}_{0,0} & \boldsymbol{Q}_{0,1} & & & & & & \\ \boldsymbol{Q}_{1,0} & \boldsymbol{Q}_{1,1} & \boldsymbol{Q}_{1,2} & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & \boldsymbol{Q}_{c+d,c+d-1} & \boldsymbol{Q}_{c+d,c+d} & \boldsymbol{C} & & & \\ & & & \boldsymbol{B} & \boldsymbol{Q}_{c+d+1,c+d+2} & \boldsymbol{C} & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \boldsymbol{B} & \boldsymbol{Q}_{c+d+N-1,c+d+N-1} & \boldsymbol{C} \\ & & & & & & \boldsymbol{B} & \boldsymbol{RB+A} \end{pmatrix}.$$

Finally, we obtain $\boldsymbol{\pi}_i$ $(i = \overline{0, c+d+N})$ by applying the Gauss–Seidel method (Chen et al. 2017) to solve Eq. (10), and obtain $\boldsymbol{\pi}_i$ $(i \geqslant c+d+N+1)$ by substituting $\boldsymbol{\pi}_{c+d+N}$ obtained in Eq. (10) into Eq. (9). So far, the steady-state probability distribution $\boldsymbol{\Pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots)$ of the system are given mathematically.

4 Performance measures

In this section, we mathematically evaluate the performance measures in terms of the average latency of tasks and the energy saving rate of the system.

We define the average latency of tasks as the average duration from the instant a task arrives at the system to the instant this task is completely processed.

Based on Little's law and the steady-state probability distribution of the system model given in Sect. 3.2, the average latency W of tasks is given as follows:

$$W = \frac{1}{\lambda}\left(\sum_{i=0}^{c} i\pi_{i0} + \sum_{i=c+1}^{c+d}\sum_{j=0}^{i-c} i\pi_{i,j} + \sum_{i=c+d+1}^{m_w}\sum_{j=0}^{d} i\pi_{i,j}\right). \tag{11}$$

m_w shown in Eq. (11) is a sufficiently large number that is set with the following condition:

$$1 - \sum_{i=0}^{m_w}\sum_{j=0}^{d} \pi_{i,j} < \varepsilon_1 \tag{12}$$

where ε_1 is an arbitrarily small number, such as $\varepsilon_1 = 10^{-13}$. The smaller the value of ε_1 is, the more precise the average latency of tasks will be.

We define the energy saving rate of the system as the energy conservation per unit time.

With the proposed strategy, the energy consumption of a VM in the awake state is higher than that in the sleep state. Indeed, additional energy will be consumed when a task is migrated from Pool II to Pool I, when a VM in Pool II listens to the system buffer, as well as when a VM in Pool II wakes up from a sleep state.

Based on the discussions above and the steady-state probability distribution of the system model given in Sect. 3.2, the energy saving rate E of the system with our proposed strategy is given as follows:

$$E = (\omega - \omega_s) \sum_{i=0}^{m_e} \sum_{j=0}^{d} (d-j)\pi_{ij} - \left(\omega_m \sum_{i=c+1}^{c+d} \sum_{j=1}^{d} c\mu_1 \pi_{ij} \right.$$
$$\left. + \omega_l \sum_{i=0}^{m_e} \sum_{j=0}^{d} \theta(d-j)\pi_{ij} + \omega_u \sum_{i=c+j+N}^{m_e} \sum_{j=0}^{d-1} \theta(d-j)\pi_{ij} \right) \tag{13}$$

where ω $(\omega > 0)$ is the energy consumption per unit time for a busy VM in Pool II, ω_s $(\omega_s > 0)$ is the energy consumption per unit time for a sleeping VM in Pool II, ω_m $(\omega_m > 0)$ is the energy consumption for each task-migration, ω_l $(\omega_l > 0)$ is the energy consumption for each listening, and ω_u $(\omega_u > 0)$ is the energy consumption for each wake-up from a sleep state to an awake state. Moreover, m_e shown in Eq. (13) is a sufficiently large number that is set with the following condition:

$$1 - \sum_{i=0}^{m_e} \sum_{j=0}^{d} \pi_{i,j} < \varepsilon_2 \tag{14}$$

where ε_2 is also an arbitrarily small number, such as $\varepsilon_2 = 10^{-13}$. The smaller the value of ε_2 is, the more precise the energy saving rate of the system will be.

5 Numerical experiments

In this section, we provide numerical experiments with analysis and simulation to further analyze the impacts of the system parameters on the system performance in terms of the average latency of tasks and the energy saving rate of the system.

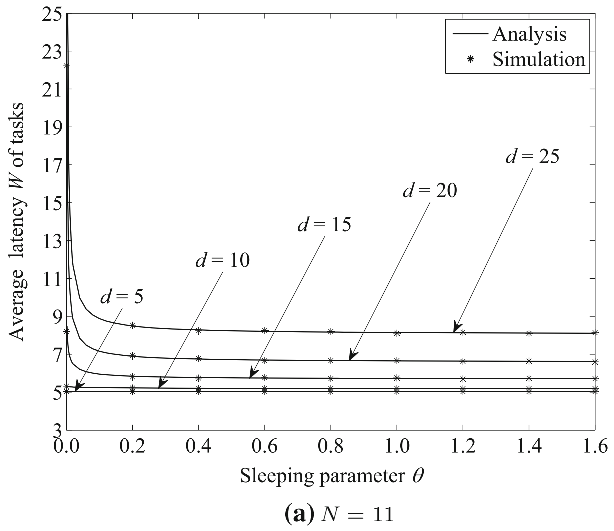
Based on Eqs. (11) and (13), the analysis results are carried out in Matlab 2016a on Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz 3.60 GHz, 8.00 GB RAM. Java language has the object-oriented features supporting the representation of multiple attributes through the definition of a class. Simulation experiments are implemented in the MyEclipse 2014 environment using Java language. In the simulation experiment, a TASK class is defined to include the attributes of UNARRIVE, WAIT, RUNHIGH, RUNLOW and FINISH. A VM class is defined to include the attributes of SLEEP, IDLE, BUSYLOW and BUSYHIGH.

Referencing to Jin et al. (2018), we set the experimental parameters as follows: $c+d = 50$, $\lambda = 7.00\ \mathrm{s}^{-1}$, $\mu_1 = 0.20\ \mathrm{s}^{-1}$, $\mu_2 = 0.10\ \mathrm{s}^{-1}$, $\omega = 0.50$ mW, $\omega_s = 0.10$ mW, $\omega_m = 0.20$ mJ, $\omega_l = 0.15$ mJ and $\omega_u = 0.20$ mJ.

Figure 3 depicts the resulting average latency W of tasks vs. the sleeping parameter θ for different numbers d of the VMs in Pool II and different wake-up thresholds N.

Figure 3a indicates that the average latency W of tasks increases with an increase of the number d of the VMs in Pool II. For a given wake-up threshold and a given sleeping parameter, the more VMs there are deployed in Pool II, the weaker the system capability becomes, and the longer the tasks will sojourn in the system. This results in a larger average latency of tasks.

Figure 3b indicates that the average latency W of tasks increases with an increase in the value for threshold N. This is due to the fact that, when the number of the VMs in Pool



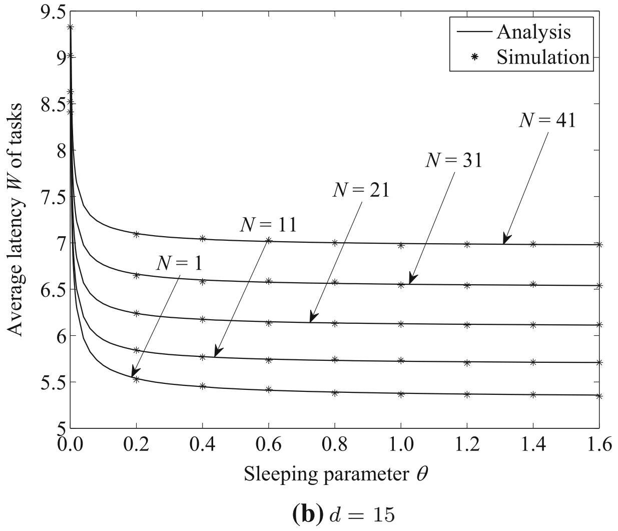


Fig. 3 The average latency L of tasks versus the sleeping parameter θ for different numbers d of the VMs in Pool II and different wake-up thresholds N

II and the sleeping parameter are given, as the value for the wake-up threshold increases, the more likely it is that the VMs in Poiol II will continue sleeping, even when their sleep timers expire. That is to say, more tasks will accumulate in the system buffer before being processed. Consequently, the average latency of tasks will increase.

From Fig. 2, we also observe that for any number d of VMs in Pool II and any value for the wake-up threshold N, as the sleeping parameter θ increases, the average latency W of tasks will initially decrease sharply and then decrease gradually. When the sleeping parameter θ is smaller (such as $0 < \theta < 0.4$ for $N = 11$ and $d = 15$), the time length of a sleep period is relatively long, so the tasks arriving during the sleep period have to wait longer in the system buffer. This leads to a larger average latency of tasks. In this case, the sleeping parameter has a greater impact on the average latency of tasks than any of the other factors, such as the arrival rate of tasks and the service rate of a task on a VM. Thus, the average latency of tasks will decrease sharply as the sleeping parameter increases.

When the sleeping parameter θ is larger (such as $0.4 < \theta < 1.6$ for $N = 11$ and $d = 15$), the time length of a sleep period is relatively short, so the tasks arriving during the sleep period will be processed earlier. This leads to a lower average latency of tasks. In this case, the arrival rate of tasks and the service rate of VMs play a dominate role in influencing the average latency of tasks. Thus, the average latency of tasks will decrease gradually as the sleeping parameter increases.

Figure 4 depicts the resulting energy saving rate E of the system vs. the sleeping parameter θ for different numbers d of the VMs in Pool II and different wake-up thresholds N.

Figure 4a indicates that deploying either relatively too few or too many VMs in Pool II will result in a lower energy saving rate E of the system. When the number of VMs in Pool II is smaller (such as $d = 5$ for $N = 11$), even though all the VMs in Pool II are asleep, less energy will be saved. This leads to a lower energy saving rate of the system. When the number of VMs in Pool II is bigger (such as $d = 25$ for $N = 11$), the system capability is relatively weaker. This is to say, once a VM in Pool II wakes up, that VM has little opportunity to go to sleep again. This also leads to a lower energy saving rate of the system.

Figure 4b indicates that a larger value for the wake-up threshold N will result in a higher energy saving rate E of the system. This is due to the fact that when the number of the VMs in Pool II and the sleeping parameter are given, the higher the value for the wake-up threshold is, the later a VM in Pool II will wake up from sleep state, and the longer the VMs in Pool II will remain asleep. This leads to a higher energy saving rate of the system.

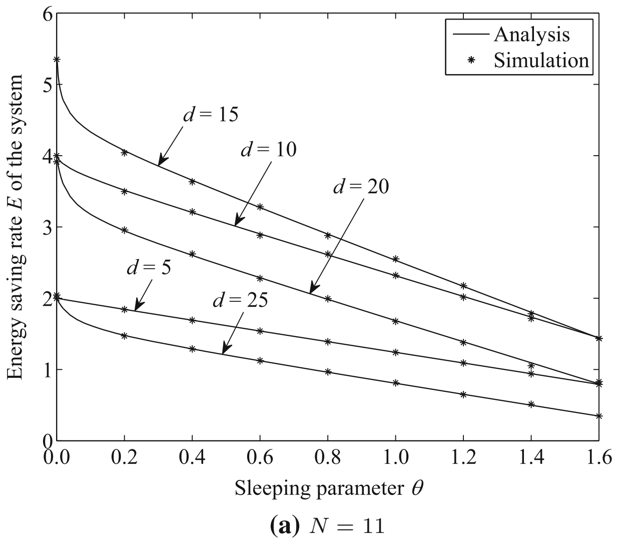
From Fig. 4, we also notice that for any number d of VMs in Pool II and for any value for the wake-up threshold N, the energy saving rate E of the system decreases as the sleeping parameter θ increases. The larger the sleeping parameter is, the shorter the time length of a sleep period is, and the earlier a VM in Pool II will wake up from a sleep state. That is to say, less energy will be saved. Additionally, the larger the sleeping parameter is, the more frequently the VM in Pool II listens to the system buffer or wakes up from sleep state. That is to say, additional energy will be consumed. Consequently, the energy saving rate of the system will decrease.

Combining the results shown in Figs. 2 and 3, we find that a lower average latency of tasks can be obtained with a smaller number of VMs in Pool II, a smaller value for the wake-up threshold and a larger sleeping parameter, whereas a higher energy saving rate of the system can be obtained with a moderate number of VMs in Pool II, a larger value for the wake-up threshold, and a smaller sleeping parameter. In the actual application of the proposed strategy, both the response performance and the energy saving effect should be taken into consideration. So, the number of the VMs in Pool II, the wake-up threshold and the sleeping parameter should be jointly optimized to balance the system performance.

6 Performance optimization

In this section, we improve the TLBO algorithm to optimize the proposed strategy.

For the optimization of the proposed strategy, we need to minimize the average latency of tasks while at the same time maximize the energy saving rate of the system. The criteria of such optimization problem are a function for the average latency of tasks to be minimized and another function for the energy saving rate of the system to be maximized. From the



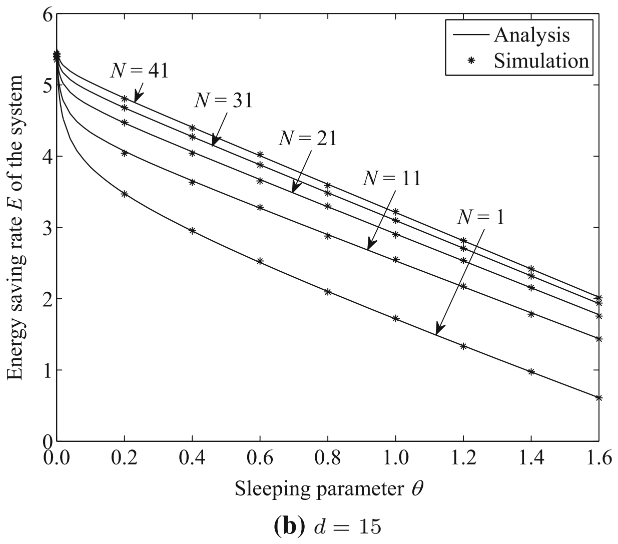


Fig. 4 The energy saving rate E of the system versus the sleeping parameter θ for different numbers d of the VMs in Pool II and different wake-up thresholds N

numerical results illustrated in Sect. 5, we find that the criteria are in conflict: a shorter average latency of tasks will certainly involve a lower energy saving rate of the system.

Considering the trade-off between the average latency of tasks and the energy saving rate of the system, we establish a system cost function $F(d, N, \theta)$ as follows:

$$F(d, N, \theta) = f_w W - f_e E \tag{15}$$

where f_w and f_e are the impact factors for the average latency W of tasks and the energy saving rate E of the system to the system cost function.

Table 1 Improved TLBO algorithm to obtain the optimal combination (d^*, N^*, θ^*)

```
Step 1: Initialize the population size $Num$, the maximum number $iter_{max}$ of iterations for the teaching–learning process, the maximal weight $\omega_{max}$ for teacher's teaching process, the minimal weight $\omega_{min}$ for teacher's teaching process, the variation sections of the sleeping parameter $[0, \theta_{max}]$, the number $X$ of total VMs
Step 2: Set the initial number $d$ of VMs in Pool II, the initial value $N$ for the wake-up threshold, the current optimal combination $(d^*, N^*, \theta^*)$, and calculate the corresponding fitness $F^*$
    $d = 0, \quad N = 1, \ (d^*, N^*, \theta^*) = (d, N, \theta_{max}), \quad F^* = F(d^*, N^*, \theta^*)$
Step 3: Initialize grade $\theta_i$ for the $i$th$(i = 1, 2, \ldots, Num)$ student by using cube chaotic equation
    $\theta_1 = rand_1$
    % $rand_1$ returns a sample in the interval $(-1, 1)$ with the uniform distribution %
    for $i = 2 : Num$
        $\theta_i = 4 * (\theta_{i-1})^3 - 3 * \theta_{i-1}$
    endfor
    for $i = 1 : Num$
        $\theta_i = (\theta_i + 1) * \theta_{max}/2$
    endfor
Step 4: Calculate the average grade $\theta_{mean}$ of current students, select the best grade $\theta_{teacher}$ as the teacher's grade, and calculate the best fitness $F_{best}$:
    $\theta_{mean} = \underset{i \in \{1,\ldots,Num\}}{mean} \theta_i,$
    $\theta_{teacher} = \underset{i \in \{1,\ldots,Num\}}{argmin} F(d, N, \theta_i), \quad F_{best} = \underset{i \in \{1,\ldots,Num\}}{min} F(d, N, \theta_i)$
Step 5: Set the initial number of iterations as $iter = 1$.
Step 6: Update each student's grade $\theta_i$ $(i = 1, 2, \ldots, Num)$ via the teacher's teaching process:
    $G = round(1 + rand), \quad \omega = \omega_{max} - (\omega_{max} - \omega_{min})(iter/iter_{max})^{1/iter}$
    % $round$ represents the rounding operation. %
    % $rand$ returns a sample in the interval $(0, 1)$ with the uniform distribution. %
    $\theta_i' = \omega * \theta_i + rand * (\theta_{teacher} - G * \theta_{mean})$
    if $F(d, N, \theta_i') < F(d, N, \theta_i)$ then $\theta_i = \theta_i'$
Step 7: Update each student's grade $\theta_i$ $(i = 1, 2, \ldots, Num)$ via the students' learning process among each others:
    Randomly select the $j$th $(j \neq i)$ student,
    if $F(d, N, \theta_j) < F(d, N, \theta_i)$ then $\theta_i = \theta_i + rand * (\theta_j - \theta_i)$
    else $\theta_i = \theta_i + rand * (\theta_i - \theta_j)$
    endif
Step 8: For the current iteration, select the best fitness $F_{gbest}$ and the best grade $\theta_{gbest}$:
    $F' = \underset{i \in \{1,\ldots,Num\}}{min} F(d, N, \theta_i)$
    if $F' < F_{best}$ then $F_{gbest} = F', \quad \theta_{gbest} = \underset{i \in \{1,\ldots,Num\}}{argmin} F(d, N, \theta_i)$
    else $F_{gbest} = F', \quad \theta_{gbest} = \underset{i \in \{1,\ldots,Num\}}{argmin} F(d, N, \theta_i)$
    endif
Step 9: Check the number of iterations:
    if $iter < iter_{max}$ then $iter = iter + 1$, go to Step 4
```

Table 1 continued

Step 10:	Update the current optimal combination (d^*, N^*, θ^*):
	if $F_{gbest} < F^*$ **then** $F^* = F_{gbest}$, $(d^*, N^*, \theta^*) = (d, N, \theta_{gbest})$
Step 11:	Check the number of VMs in Pool II:
	if $d < X$ **then** $d = d + 1$, go to **Step 3**
Step 12:	Check the value for the wake-up threshold:
	if $N < X$ **then** $N = N + 1$, go to **Step 3**
Step 13:	Output the optimal combination (d^*, N^*, θ^*).

The optimization of the proposed strategy turns to minimize the system cost function and obtaining the optimal combination (d^*, N^*, θ^*). That is as follows:

$$(d^*, N^*, \theta^*) = argmin\ F(d, N, \theta). \tag{16}$$

For a synchronous sleep-mode, in order to make sure that all the tasks waiting in the system buffer can be processed immediately once all the sleeping VMs wake up, the maximum wake-up threshold N is set as the number of the sleeping VMs. Obviously, the maximum wake-up threshold N in an asynchronous sleep-mode should not be higher than that in a synchronous sleep-mode. In the clustered VM allocation strategy based on an asynchronous sleep-mode proposed in this paper, we set the maximum wake-up threshold N as the number d of the VMs in Pool II. Therefore, when solving the optimization problem in Eq. (16), the value range for the wake-up threshold N is $[1, d]$.

We note that it is difficult to express a system cost function in a closed-form, and also it is difficult to determine the monotonicity of the system cost function. Moreover, we note that the system cost function is not differentiable and there are possible more than one extreme point in the system cost function. In a result, the optimum point of the system cost function can not be searched by using classical numerical methods. In order to jointly optimize the number of VMs in Pool II, the wake-up threshold and the sleeping parameter with the minimum system cost function, we turn to an intelligent optimization algorithm, TLBO algorithm.

TLBO is one of the most efficient algorithms for providing good quality solutions in a reasonable computation time (Buddala and Mahapatra 2018). However, there are still some drawbacks with the TLBO algorithm, such as low population diversity and poor searching ability (Wang et al. 2018; Liu et al. 2018). For the purpose of improving the TLBO algorithm, we introduce a cube chaotic mapping mechanism to disperse the initial grades of students, and use an exponentially decreasing strategy for the teaching process to enhance the searching ability of the algorithm. The main steps for the improved TLBO algorithm are given in Table 1.

For the given system parameters, such as the total number $(c + d)$ of VMs in a cloud data center, the arrival rate λ and the service rate μ_1 of a task on the VM in Pool I, etc., the optimal combination (d^*, N^*, θ^*) for the number of VMs in Pool II, the wake-up threshold and the sleeping parameter can be obtained with the proposed strategy by performing the improved TLBO algorithm in Table 1.

Using the system parameters given in Sect. 5, and setting $Num = 100$, $iter_{max} = 200$, $\omega_{max} = 0.8$, $\omega_{min} = 0.1$, $\theta_{max} = 2$, $X = 50$, $f_w = 4$ and $f_e = 1$ as an example, we execute the improved TLBO algorithm given in Table 1. With a different service rates μ_2 of a task on the VM in Pool II, we obtain the optimal combination (d^*, N^*, θ^*) for the number of VMs

Table 2 Optimization outcomes: $(d^{*}, N^{*}, \theta^{*})$ and F^{*}

Service rate μ_2 of a task on the VM in Pool II	Optimal combination $(d^{*}, N^{*}, \theta^{*})$	Minimum system cost F^{*}
0.05	(9, 2, 0.0001)	17.2012
0.10	(10, 10, 0.0403)	17.1864
0.15	(11, 4, 0.0945)	17.0491
0.20	(49, 2, 0.2846)	16.1823

in Pool II, the wake-up threshold and the sleeping parameter with the minimum system cost function F^{*} in Table 2.

The cloud capacity, the arrival intensity of tasks and the serving capability of VMs greatly influence the optimal outcomes in Table 2. With the optimization outcomes shown in Table 2, we can trade off the average latency of tasks and the energy saving rate of the system in the proposed strategy.

7 Conclusions

In this paper, we proposed a clustered VM allocation strategy with a novel sleep mode. Considering the sleep-mode with wake-up threshold in the proposed strategy, we established a queue with an N-policy and asynchronous vacations of partial servers. Based on the stochastic behavior of tasks with the proposed strategy, we built the QBD matrix and resolved the steady-state transition probabilities using the matrix-geometric solution method. Moreover, we evaluated the system performance in terms of the average latency of tasks and the energy saving rate of the system. Numerical results with analysis and simulation show that the average latency of tasks is lower with a smaller number of the VMs in Pool II, a smaller value for the wake-up threshold and a larger sleeping parameter, while the energy saving rate of the system is higher with a moderate number of VMs in Pool II, a larger value for the wake-up threshold and a smaller sleeping parameter. For this, we constructed a system cost function to balance different performance measures. By introducing a cube chaotic mapping mechanism for the grade initialization and an exponentially decreasing strategy for the teaching process, we developed an improved TLBO algorithm, and optimized the proposed strategy with the minimum system cost function.

References

Buddala, R., & Mahapatra, S. (2018). An integrated approach for scheduling flexible job-shop using teaching–learning-based optimization method. *Journal of Industrial Engineering International*. https://doi.org/10.1007/s40092-018-0280-8.

Cao, H., Xu, J., Ke, D., Jin, C., Deng, S., Tang, C., et al. (2016). Economic dispatch of micro-grid based on improved particle-swarm optimization algorithm. In *Proceedings of North American power symposium, NAPS, 2016*. https://doi.org/10.1109/NAPS.2016.7747875.

Chen, L., Sun, D., & Toh, K. (2017). An efficient inexact symmetric Gauss–Seidel based majorized ADMM for high-dimensional convex composite conic programming. *Mathematical Programming*, *161*(1–2), 237–270.

Chou, C., Wong, D., & Bhuyan, L. (2016). DynSleep: Fine-grained power management for a latency-critical data center application. In *Proceedings of the ACM/IEEE international symposium on low power electronics and design, ISLPED 2016* (pp. 212–217).

Do, N., Van, D., & Melikov, A. (2018). Equilibrium customer behavior in the M/M/1 retrial queue with working vacations and a constant retrial rate. *Operational Research*. https://doi.org/10.1007/s12351-017-0369-7.

Duan, L., Zhan, D., & Hohnerlein, J. (2015). 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).

Jin, S., Hao, S., & Yue, W. (2017). 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).

Jin, S., Wang, X., & Yue, W. (2018). A task scheduling strategy with a sleep-delay timer and a waking-up threshold in cloud computing. In *Proceedings of the 13th international conference on queueing theory and network applications, QTNA2018* (pp. 115–123).

Jin, S., Wu, H., & Yue, W. (2018). Pricing policy for a cloud registration service with a novel cloud architecture. *Cluster Computing*. https://doi.org/10.1007/s10586-018-2854-z.

Ji, X., Ye, H., & Zhou, J. (2017). An improved teaching–learning-based optimization algorithm and its application to a combinatorial optimization problem in foundry industry. *Applied Soft Computing*, *2017*, 504–516.

Khojandi, A., Shylo, O., & Zokaeinikoo, M. (2018). Automatic EEG classification: A path to smart and connected sleep interventions. *Annals of Operations Research*. https://doi.org/10.1007/s10479-018-2823-1.

Li, L., Weng, W., & Fujimura, S. (2017). An improved teaching–learning-based optimization algorithm to solve job shop scheduling problems. In *Processding of the 3rd international conference on computer and information sciences, ICCIS 2017* (pp. 797–801).

Liu, J., Jin, S., & Yue, W. (2018). Performance evaluation and system optimization of Green cognitive radio networks with a multiple-sleep mode. *Annals of Operations Research*. https://doi.org/10.1007/s10479-018-3086-6.

Luo, J., Zhang, S., Yin, L., & Guo, Y. (2017). 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).

Marek, R., & Hoon, K. (2018). Cognitive systems and operations research in big data and cloud computing. *Annals of Operations Research*, *265*(2), 183–186.

Neuts, M. (1981). *Matrix-geometric solutions in stochastic models*. Baltimore: Johns Hopkins University Press.

Tian, N., Gao, Z., & Zhang, Z. (2001). The equilibrium theory for queueing system M/M/C with asynchronous vacations. *Acta Mathematicae Application Sinica*, *24*(2), 185–194 (**in Chinese**).

Tian, N., & Zhang, Z. (2006). *Vacation queueing models: Theory and applications*. New York: Springer.

Wang, B., Li, H., & Feng, Y. (2018). An improved teaching–learning-based optimization for constrained evolutionary optimization. *Information Sciences*, *456*, 131–144.

Yu, K., Wang, X., & Wang, Z. (2016). An improved teaching–learning-based optimization algorithm for numerical and engineering optimization problems. *Journal of Intelligent Manufacturing*, *27*(4), 831–843.

Zhou, Z., Abawajy, J., & Li, F. (2018). Fine-grained energy Zhou data center. *IEEE Access*, *6*, 27080–27090.

Affiliations

Shunfu Jin[1] · Xiuchen Qie[1] · Wenjuan Zhao[1] · Wuyi Yue[2] · Yutaka Takahashi[3]

Xiuchen Qie
qiexiuchen@126.com

Wenjuan Zhao
zwj8569@163.com

Wuyi Yue
yue@konan-u.ac.jp

Yutaka Takahashi
takahashi@i.kyoto-u.ac.jp

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

[2] Department of Intelligence and Informatics, Konan University, Kobe, Japan

[3] Graduate School of Informatics, Kyoto University, Kyoto, Japan