Journal of the Operations Research Society of China (2020) 8:561–580
https://doi.org/10.1007/s40305-019-00272-x

Performance Evaluation and Social Optimization of an Energy-Saving Virtual Machine Allocation Scheme Within a Cloud Environment

Xiushuang Wang[1] · Jing Zhu[1] · Shunfu Jin[1] · Wuyi Yue[2] · Yutaka Takahashi[3]

Received: 27 November 2018 / Revised: 8 September 2019 / Accepted: 3 October 2019 /
Published online: 11 November 2019

Abstract

Achieving greener cloud computing is non-negligible for the open-source cloud platform. In this paper, we propose a novel virtual machine allocation scheme with a sleep-delay and establish a corresponding mathematical model. Taking into account the number of tasks and the state of the physical machine, we construct a two-dimensional Markov chain and derive the average latency of tasks and the energy-saving degree of the system in the steady state. Moreover, we provide numerical experiments to show the effectiveness of the proposed scheme. Furthermore, we study the Nash equilibrium behavior and the socially optimal behavior of tasks and carry out an improved adaptive genetic algorithm to obtain the socially optimal arrival rate of tasks. Finally,

This work was supported in part by the National Natural Science Foundation of China (Nos. 61872311, 61973261, 61472342) and Hebei Provincial Natural Science Foundation (No. F2017203141), China, and was supported in part by MEXT and JSPS KAKENHI (Nos. JP17H01825 and JP26280113), Japan.

✉ Shunfu Jin
jsf@ysu.edu.cn

Xiushuang Wang
ysuwxs@163.com

Jing Zhu
zhuj6886@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 066004, Shandong, China

2 Department of Intelligence and Informatics, Konan University, Kobe 658-8501, Japan

3 Graduate School of Informatics, Kyoto University, Kyoto 606-8225, Japan

we present a pricing policy for tasks to maximize the social profit when managing the network resource within the cloud environment.

Keywords Cloud computing · Resource allocation scheme · Mathematical analysis · Markov chain · Socially optimization · Genetic algorithm

Mathematics Subject Classification 68M20 · 60K20 · 60K25 · 91A15

1 Introduction

As a commercial infrastructure paradigm, cloud computing has revolutionized the IT industry [1,2]. However, the energy consumption of cloud computing shows a rising trend, while the resources themselves are highly underutilized [3,4]. This presents a bottleneck that restricts the improvement of cloud computing and reveals the great importance of greening the networks.

Consolidation of virtual machines (VMs) is an effective technique to minimize the excess energy consumption resulting from the diversity of workload. Many scholars have targeted solving the consolidation problem to improve resource utilization and reduce energy consumption over the cloud environment. In [5], Fard et al. presented a dynamic VM consolidation technique, in which the detections of server overload and server underload were supported. By calculating the deviation between the utilization and the threshold of the overload server, the VMs were consolidated until the number of VMs reached an upper threshold. In [6], Khoshkholghi et al. proposed a dynamic and adaptive energy-efficient VM consolidation mechanism by developing a two-phase VM allocation algorithm for the placement of new VMs and the consolidation of selected VMs. Using the consolidation methodologies mentioned above, energy conservation is achieved to a certain degree. However, the situation of all the VMs stays awake even though no tasks are to be processed remains.

In fact, VMs in cloud computing are usually underutilized to guarantee the quality of experience (QoE) of users. Extensive studies have consequently been conducted on how to reduce energy consumption during lower workload periods. In [7], Cheng et al. presented an energy-saving task scheduling algorithm for a heterogeneous cloud computing system based on a type of vacation queueing model. In this model, the idle VMs were on vacation, and the vacation was similar to being in a sleep mode. In [8], Guo et al. conducted a theoretical study into the impact of a base station (BS) sleeping on both energy-efficiency and user-perceived delay, and presented three typical wake-up schemes in terms of single sleep (SS), multiple sleep (MS) and N-limited schemes. In [9], Juwo et al. proposed a two-stage BS sleeping scheme in cellular networks. The sleeping mode was divided into a light sleeping stage and a deep sleeping stage according to whether there were tasks in the coverage of the BS. With the two-stage sleeping scheme, the BS frequently switched between the on state, the doze state and the shut-down state.

As discussed above, putting idle VMs into sleep mode can reduce the energy consumption. However, additional energy will be consumed and the user performance

will be degraded due to the frequent state switches of physical machines (PMs) using a conventional sleep mode.

Inspired by these observations, the trade-off between providing higher QoE to users and reducing energy consumption should be addressed. In this paper, we propose an energy-saving VM allocation scheme with synchronous multi-sleep and sleep-delay. By building a type of vacation queuing model, we evaluate and optimize the system performance of the proposed scheme.

The main contributions of this paper can be listed as follows:

(1) We propose a novel energy-saving VM allocation scheme with the constraint of response performance within a cloud environment. When the system is empty, all the VMs hosted on a PM, and the PM itself, keep awake for a period, rather than instantly switching into the sleep state, so the newly arriving tasks can receive timely service. In this way, the QoE of users can be guaranteed, while the additional energy consumption can be effectively reduced.
(2) We present a method to model the proposed scheme and evaluate the system performance mathematically. We establish a multi-server queueing model to capture the stochastic behavior of tasks in the cloud data center with the proposed scheme. By constructing a two-dimensional Markov chain (MC), we evaluate the system performance in terms of the average latency of tasks and the energy-saving degree of the system.
(3) We give a pricing policy charging for tasks to optimize the social profit. Based on the reward for a processed task and the cost for a task waiting in the system buffer, we investigate the Nash equilibrium behavior. Considering also the saved income derived by a cloud service provider due to the energy conservation, we build a revenue function to investigate the socially optimal behavior of tasks. In order to maximize the social profit, we present an enhanced intelligent searching algorithm to obtain the socially optimal arrival rate of tasks and impose an appropriate admission fee on tasks.

The rest of this paper is organized as follows: In Sect. 2, we present our proposed system model with an energy-saving virtual machine (VM) allocation scheme along with its mathematical model. In Sect. 3, we present the steady-state distribution using the matrix geometric solutions and describe the performance measures. In Sect. 4, we present applications and their numerical results to evaluate the system performance with the proposed scheme. In Sect. 5, we impose a pricing policy for motivating tasks to accept the socially optimal strategy. Finally, we draw conclusions from the whole paper in Sect. 6.

2 System Model

In this section, we present our proposed system model with an energy-saving VM allocation and its mathematical model.

2.1 Energy-Saving VM Allocation

In a cloud computing system, many PMs constitute the real cloud servers. In order to process multiple tasks simultaneously and maintain an acceptable service level agreement (SLA), several identical VMs will be hosted on one PM. A VM is a software that works like a PM. In an open-source cloud platform, all the VMs hosted on a PM and the PM itself are always awake, even when there are no tasks to be processed. Thus, large amounts of power are wasted.

Sleep modes are intended to minimize the system power usage. In the sleep state, one or more operational components are powered down; only the event monitor stays active with a very low energy consumption. Obviously, by using the sleep mode, the energy consumption can be reduced. However, the system response performance will be degraded. With the aim of trading off the energy consumption against the response performance, we propose a novel energy-saving VM allocation scheme with sleep-delay within a cloud environment.

In the proposed energy-saving VM allocation scheme, the PM will switch among the awake state, the sleep-delay state and the sleep state. The state transition is illustrated in Fig. 1.

(1) *Awake State* During the awake state of a PM, there is at least one VM busy with task processing. The tasks are processed continuously following a first-come first-served discipline. The event monitor deployed in the PM is mainly used for listening to the system to see if all the tasks are processed completely. Once all the tasks in the system have been processed completely, i.e., the system becomes empty, under the control of the VM scheduler, all the VMs will remain awake for

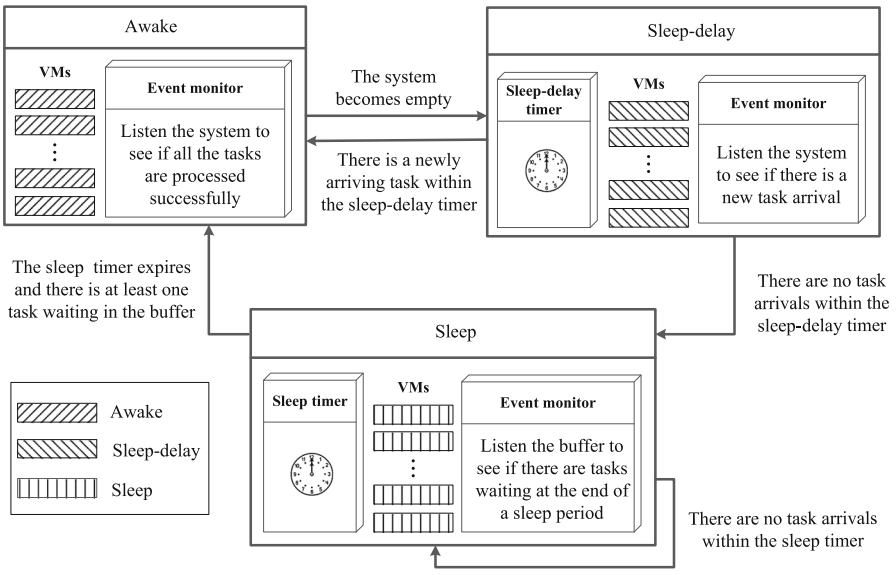

Fig. 1 State transition of the energy-saving VM allocation scheme with sleep-delay within a cloud environment

a period within the sleep-delay timer and be ready for providing service at any time. The PM will change into the sleep-delay state.

(2) *Sleep-delay State* Once the PM enters the sleep-delay state, a sleep-delay timer with a random duration will be activated to control the maximum time length of a sleep-delay period. During the sleep-delay state, the event monitor in the PM will listen to the system to see whether there are any new tasks arriving at the system.

If there is a task arrival before the sleep-delay timer expires, under the control of the VM scheduler, all the VMs will provide service immediately for the newly incoming task. The PM will switch to the awake state. In this way, the response performance of tasks will be guaranteed. In this case, the time period of the sleep-delay begins when the sleep-delay timer is activated and finishes the instant a new task arrives at the system.

If there are no task arrivals within the sleep-delay timer, under the control of the VM scheduler, all the VMs will go to sleep at the instant the sleep-delay timer expires. The PM will switch to the sleep state. In this case, the time period of the sleep-delay begins when the sleep-delay timer is activated and concludes when the sleep-delay timer expires.

(3) *Sleep State* Once the PM enters the sleep state, a sleep timer with a random duration will be activated to control the time length of a sleep period. Tasks arriving within the sleep timer will queue in the buffer. At the end of a sleep period, the event monitor in the PM mainly listens to the buffer to see if there are tasks queueing in the system. If there are no tasks waiting in the system buffer, another sleep timer will be activated. Under the control of the VM scheduler, all the VMs will begin another sleep period. The PM will remain in the sleep state. Otherwise, all the VMs will wake up to serve all the tasks in the system one by one, and the PM will return to the awake state.

In the sleep state, all the VMs in the PM no longer consume memory and CPU. Thus, the energy consumption in the sleep state is lower than that in any other states.

2.2 Mathematical Model

In order to investigate the influence of arrival behaviors on the system performance under different sleep parameters and sleep-delay parameters, we establish a mathematical model based on the proposed energy-saving VM allocation scheme.

The tasks submitted to the cloud computing system are regarded as customers. Each VM hosted on the PM is regarded as a server. The sleep state is regarded as a vacation and the sleep-delay state is regarded as a vacation-delay. Given this, we establish a multi-server queueing system with a synchronous multi-vacation and a vacation-delay.

We assume the tasks arriving at the system follow Poisson process with parameter λ ($0 < \lambda < +\infty$), and the service time of a task follows an exponential distribution with service rate μ ($0 < \mu + \infty$). In addition, we assume the timer lengths of the sleep-delay period and the sleep period follow exponential distributions with parameters β ($0 < \beta < +\infty$) and θ ($0 < \theta < +\infty$), respectively.

In this paper, we focus on all the identical VMs hosted on one PM. We suppose the number of VMs in the system is n and the system capacity is infinite. Let $X_t = i$ ($i =$

$0, 1, \cdots)$ be the number of tasks in the system at the instant t. X_t is also called the system level. We let $Y_t = j$ $(j = 0, 1, 2)$ be the PM state at the instant t. Y_t is called the PM state. $j = 0$ means the PM is in the sleep state, $j = 1$ means the PM is in the awake state, and $j = 2$ means the PM is in the sleep-delay state.

Based on the assumptions above, $\{(X_t,\ Y_t),\ t \geqq 0\}$ constitutes a two-dimensional Markov chain (MC). The state space of the MC is given as follows:

$$\boldsymbol{\Omega} = \{(0, 0) \cup (0, 2) \cup (i, j) : i \geqq 1,\ j = 0, 1, 2\}.$$

Let $\pi_{i,j}$ be the steady-state distribution of the two-dimensional MC. $\pi_{i,j}$ is defined as follows:

$$\pi_{i,j} = \lim_{t\to\infty} P\{X_t = i,\ Y_t = j\},\ i = 0, 1, \cdots,\ j = 0, 1, 2.$$

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

$$\boldsymbol{\pi}_i = (\pi_{i,0},\ \pi_{i,1},\ \pi_{i,2}),\ i = 0, 1, \cdots.$$

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

$$\boldsymbol{\Pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \cdots).$$

3 Performance Analysis

In this section, we present the transition rate matrix, the steady-state distribution, and the performance measures.

3.1 Transition Rate Matrix

One of the most important steps in analyzing the steady-state distribution of the MC is to construct the transition rate matrix.

Let $\boldsymbol{Q}$ be the one-step state transition rate matrix of the two-dimensional MC $\{(X_t,\ Y_t),\ t \geqq 0\}$, and $\boldsymbol{Q}_{x,y}$ be the sub-matrix of $\boldsymbol{Q}$ for the system level changing from x $(x = 0, 1, \cdots)$ to y $(y = 0, 1, \cdots)$. Each sub-matrix $\boldsymbol{Q}_{x,y}$ in the one-step state transition rate matrix $\boldsymbol{Q}$ will be dealt with in detail as follows.

(1) *System Level Increases*
$x = 0$ means that there are no tasks to be processed or being processed in the system. The PM can only be in the sleep state $(j = 0)$ or the sleep-delay state $(j = 2)$. For the case that the initial PM state is $j = 0$, if a task arrives at the system within the sleep timer, the system level increases by one, but the PM state remains fixed, and the transition rate will be λ. For the case that the initial PM state is $j = 2$, if a task arrives at the system within the sleep-delay timer, the system

level increases by one, and the PM returns to the awake state. The transition rate will also be λ. Thus, the sub-matrix $\boldsymbol{Q}_{0,1}$ can be written as follows:

$$\boldsymbol{Q}_{0,1} = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \lambda & 0 \end{pmatrix}. \tag{3.1}$$

x ($x \geqq 1$) means that there is at least one task in the system. The PM is in the sleep state ($j = 0$) or the awake state ($j = 1$). For these two cases, if a task arrives at the system, the system level increases by one, but the PM state remains fixed, and the transition rate will be λ. Thus, the sub-matrix $\boldsymbol{Q}_{x,x+1}$ can be written as follows:

$$\boldsymbol{Q}_{x,x+1} = \begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ x \geqq 1. \tag{3.2}$$

(2) *System Level Remains Fixed*

$x = 0$ means that the PM can only be in the sleep state ($j = 0$) or the sleep-delay state ($j = 2$). For the case that the initial PM state is $j = 0$, if the sleep timer expires and there are no tasks waiting at the system buffer, the PM will enter another sleep period, both the system level and the PM state will remain unchanged, and the transition rate will be $-\lambda$. For the case that the initial PM state is $j = 2$, if the sleep-delay timer expires and there are no tasks waiting at the system, the system level remains fixed, but the PM changes to the sleep state, and the transition rate is β. If there are no tasks arriving at the system within the sleep-delay timer, both the system level and the PM state remain fixed, and the transition rate will be $-(\lambda + \beta)$. Thus, the sub-matrix $\boldsymbol{Q}_{0,0}$ can be written as follows:

$$\boldsymbol{Q}_{0,0} = \begin{pmatrix} -\lambda & 0 & 0 \\ 0 & 0 & 0 \\ \beta & 0 & -(\lambda + \beta) \end{pmatrix}. \tag{3.3}$$

x ($1 \leqq x < n$) means that in the system, the number of tasks is less than the number of VMs. The PM can only be in the sleep state ($j = 0$) or the awake state ($j = 1$). For the case that the initial PM state is $j = 0$, if there are no tasks arriving at the system buffer within the sleep timer, both the system level and the PM state remain fixed, and the transition rate will be $-(\lambda + \theta)$. If the sleep timer expires and there is at least one task waiting in the system buffer, the system level remains fixed, but the PM changes to the awake state, and the transition rate will be θ. For the case that the initial PM state is $j = 1$, if neither an arrival nor a departure occurs, both the system level and the PM state remain unchanged, and the transition rate will be $-(\lambda + x\mu)$. Thus, the sub-matrix $\boldsymbol{Q}_{x,x}$ can be written as follows:

$$\boldsymbol{Q}_{x,x} = \begin{pmatrix} -(\lambda + \theta) & \theta & 0 \\ 0 & -(\lambda + x\mu) & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ 1 \leqq x < n. \tag{3.4}$$

x ($x \geqq n$) means that in the system, the number of tasks is greater than the number of VMs. The PM can only be in the sleep state ($j = 0$) or the awake state ($j = 1$). For the case that the initial PM state is $j = 0$, the corresponding transition rates are the same as that given in Eq. (3.4). For the case that the initial PM state is $j = 1$, if neither an arrival nor a departure occurs, both the system level and the PM state remain fixed, and the transition rate will be $-(\lambda + n\mu)$. Thus, the sub-matrix $\boldsymbol{Q}_{x,x}$ can be written as follows:

$$\boldsymbol{Q}_{x,x} = \begin{pmatrix} -(\lambda+\theta) & \theta & 0 \\ 0 & -(\lambda+n\mu) & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ x \geqq n. \tag{3.5}$$

(3) *System Level Decreases*
In the case where the system level is decreased, the initial PM state can only be in the awake state ($j = 1$).

$x = 1$ means that there is one task being processed in the system. If the only task in the system is completely processed, the system level decreases by one, and the PM switches to the sleep-delay state, and the transition rate will be μ. Thus, the sub-matrix $\boldsymbol{Q}_{1,0}$ can be written as follows:

$$\boldsymbol{Q}_{1,0} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \mu \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.6}$$

x ($1 < x \leqq n$) means that all the tasks are being processed and the system buffer is empty. If one of the tasks is completely processed, the system level decreases by one, but the PM state remains fixed, and the transition rate will be $x\mu$. Thus, the sub-matrix $\boldsymbol{Q}_{x,x-1}$ can be written as follows:

$$\boldsymbol{Q}_{x,x-1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & x\mu & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ 1 < x \leqq n. \tag{3.7}$$

x ($x > n$) means that there are n tasks being processed and $(x-n)$ tasks waiting in the system buffer. If one of the tasks is completely processed, the first task queueing in the system buffer will occupy the just evacuated VM to receive service. The system level decreases by one, but the PM state remains fixed, and the transition rate will be $n\mu$. Thus, the sub-matrix $\boldsymbol{Q}_{x,x-1}$ can be written as follows:

$$\boldsymbol{Q}_{x,x-1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & n\mu & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ x > n. \tag{3.8}$$

Now, all the sub-matrices in $\boldsymbol{Q}$ have been addressed. For convenience of description, we denote $\boldsymbol{Q}_{x,x-1}$ ($0 \leqq x < n$) as $\boldsymbol{B}_x$, $\boldsymbol{Q}_{x,x}$ ($0 \leqq x < n$) as $\boldsymbol{A}_x$, $\boldsymbol{Q}_{x,x+1}$ ($0 \leqq x < n$) as

$\boldsymbol{C}_x$. We note that starting from the system level n, all the sub-matrices of $\boldsymbol{Q}$ are repeated forever. So we denote the repetitive $\boldsymbol{B}_x, \boldsymbol{A}_x$ and $\boldsymbol{C}_x$ as $\boldsymbol{B}, \boldsymbol{A}$ and $\boldsymbol{C}$, respectively. Using the lexicographical sequence for the system level and the system state, the infinitesimal generator for the MC is given as follows:

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{A}_0 & \boldsymbol{C}_0 & & & & & \\ \boldsymbol{B}_1 & \boldsymbol{A}_1 & \boldsymbol{C}_1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \boldsymbol{B}_{n-1} & \boldsymbol{A}_{n-1} & \boldsymbol{C}_{n-1} & & \\ & & & \boldsymbol{B} & \boldsymbol{A} & \boldsymbol{C} & \\ & & & & \ddots & \ddots & \ddots \end{pmatrix}. \tag{3.9}$$

From the matrix structure of $\boldsymbol{Q}$ given in Eq. (3.9), we find that the system state transitions occur only between adjacent levels. Hence, the two-dimensional MC $\{(X_t,\ Y_t),\ t \geqq 0\}$ can be seen as a type of quasi birth-and-death (QBD) process.

3.2 Steady-State Distribution

To analyze this QBD process, we need to solve for the minimal non-negative solution of the matrix quadratic equation:

$$\boldsymbol{R}^2\boldsymbol{A} + \boldsymbol{R}\boldsymbol{B} + \boldsymbol{C} = \boldsymbol{0}. \tag{3.10}$$

The solution $\boldsymbol{R}$ is called the rate matrix [10]. Rate matrix $\boldsymbol{R}$ plays a crucial role in the analysis of the mathematical model. Under the condition that $\rho = \lambda(n\mu)^{-1} < 1$, Eq. (3.10) has a minimal non-negative solution $\boldsymbol{R}$.

Since the coefficient matrices of Eq. (3.10) are all upper triangular, and all the elements of the third row and the third column are zero, the rate matrix $\boldsymbol{R}$ can be written in the following form:

$$\boldsymbol{R} = \begin{pmatrix} r_{11} & r_{12} & 0 \\ 0 & r_{22} & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.11}$$

Substituting Eq. (3.11) into Eq. (3.10) yields the following set of equations:

$$\begin{cases} n\mu(r_{11}r_{12} + r_{12}r_{22}) + \theta r_{11} - (\lambda + n\mu)r_{12} = 0, \\ n\mu {r_{22}}^2 - (\lambda + n\mu)r_{22} + \lambda = 0, \\ -(\lambda + \theta)r_{11} + \lambda = 0. \end{cases} \tag{3.12}$$

By solving Eq. (3.12), we can obtain $r_{11} = \dfrac{\lambda}{\lambda + \theta}$, $r_{12} = \rho$ and $r_{22} = \rho$. For this case, $SP(\boldsymbol{R}) = \max\left(\dfrac{\lambda}{\lambda + \theta},\ \rho\right) < 1$ [11]. Thus, the rate matrix $\boldsymbol{R}$ is derived as

follows:

$$\boldsymbol{R} = \begin{pmatrix} \dfrac{\lambda}{\lambda+\theta} & \rho & 0 \\ 0 & \rho & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.13}$$

To use the matrix geometric solution method [12], the infinitesimal generator $\boldsymbol{Q}$ of the QBD process is re-partitioned as follows:

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{H}_{0,0} & \boldsymbol{H}_{0,1} & & & \\ \boldsymbol{H}_{1,0} & \boldsymbol{A} & \boldsymbol{C} & & \\ & \boldsymbol{B} & \boldsymbol{A} & \boldsymbol{C} & \\ & & \ddots & \ddots & \ddots \end{pmatrix}, \tag{3.14}$$

where

$$\boldsymbol{H}_{0,0} = \begin{pmatrix} \boldsymbol{A}_0 & \boldsymbol{C}_0 & & & \\ \boldsymbol{B}_1 & \boldsymbol{A}_1 & \boldsymbol{C}_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \boldsymbol{B}_{n-2} & \boldsymbol{A}_{n-2} & \boldsymbol{C}_{n-2} \\ & & & \boldsymbol{B}_{n-1} & \boldsymbol{A}_{n-1} \end{pmatrix}_{3n\times 3n}, \tag{3.15}$$

$$\boldsymbol{H}_{1,0} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{B} \end{pmatrix}_{3\times 3n}, \ \boldsymbol{H}_{0,1} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{C} \end{pmatrix}_{3n\times 3}. \tag{3.16}$$

Using the rate matrix $\boldsymbol{R}$ obtained from Eq. (3.13), we give a square matrix as follows:

$$B[\boldsymbol{R}] = \begin{pmatrix} \boldsymbol{H}_{00} & \boldsymbol{H}_{01} \\ \boldsymbol{H}_{10} & \boldsymbol{A} + \boldsymbol{R}\boldsymbol{B} \end{pmatrix}. \tag{3.17}$$

Then, $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}_1$ satisfy the following equation:

$$\begin{cases} (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1) B[\boldsymbol{R}] = 0, \\ \boldsymbol{\pi}_0 \boldsymbol{e} + \boldsymbol{\pi}_1 (\boldsymbol{I} - \boldsymbol{R})^{-1} \boldsymbol{e} = 1, \end{cases} \tag{3.18}$$

where $\boldsymbol{e}$ is a column vector with ones.

Based on Eq. (3.18), we further construct an augmented matrix as follows:

$$(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1) \left(B[\boldsymbol{R}] \,\vdots\, \begin{matrix} \boldsymbol{e} \\ (\boldsymbol{I} - \boldsymbol{R})^{-1} \boldsymbol{e} \end{matrix} \right) = (\underbrace{0, 0, \cdots, 0}_{3(n+1)}, 1). \tag{3.19}$$

By using the Gauss–Seidel method [13] to solve Eq. (3.19), we can obtain $\boldsymbol{\pi}_0$ and $\boldsymbol{\pi}_1$.

From the structure of the one-step state transition rate matrix $\boldsymbol{Q}$ given in Eq. (3.14), we know that $\boldsymbol{\pi}_i$ $(i = 2, 3, \cdots)$ satisfies the matrix geometric solution form as follows:

$$\boldsymbol{\pi}_i = \boldsymbol{\pi}_1 \boldsymbol{R}^{i-1}, \ i \geqq 2. \tag{3.20}$$

By substituting $\boldsymbol{\pi}_1$ obtained in Eq. (3.19) into Eq. (3.20), we can obtain $\boldsymbol{\pi}_i$ $(i = 2, 3, \cdots)$. Then, the steady-state distribution $\boldsymbol{\Pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \cdots)$ of the system can be presented numerically.

3.3 Performance Measures

We define the latency of a task as the duration from the instant a task arrives at the system to the instant when that task is completely processed. In other words, the latency of a task is the sum of the waiting time in the system buffer and the service time on the VM. The average latency of tasks is an important factor in measuring the QoE of cloud users.

According to the steady-state distribution given in Sect. 3.2, and using Little's law [14,15], the average latency W of tasks is then given as follows:

$$W = \frac{1}{\lambda}\left(\sum_{i=0}^{\infty} i(\pi_{i,0} + \pi_{i,1} + \pi_{i,2})\right). \tag{3.21}$$

We define the energy-saving degree of the system as the energy conservation per unit time for our proposed energy-saving VM allocation scheme. During the awake state and the sleep-delay state, energy will be consumed normally, while during the sleep state, energy can be saved. However, additional energy will be consumed when the PM switches from the sleep state to the awake state, and the VMs listen to the system buffer at the boundary of each sleep period.

Let C_a be the energy consumption per unit time in the awake state and the sleep-delay state, and C_s be the energy consumption per unit time in the sleep state. Let C_t be the energy consumption for each switching from the sleep state to the awake state, and C_l be the energy consumption for each listening. Without loss of generality, we suppose $C_a > C_s$ in this paper. Then, the energy-saving degree E_v of the system is given as follows:

$$E_v = \sum_{i=0}^{\infty} \pi_{i,0} \times (C_a - C_s) - \sum_{i=1}^{\infty} \pi_{i,0} \times \theta \times C_t - \sum_{i=0}^{\infty} \pi_{i,0} \times \theta \times C_l. \tag{3.22}$$

4 Numerical Results

In order to investigate the impact of the parameters on the system performance with the proposed energy-saving VM allocation scheme, we provide numerical experiments with analysis and simulations. All the experiments are performed on a computer configured with Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz, 8.00 GB RAM and 500G disk. Under the MATLAB platform, the analysis experiments are calculated based on Eqs. (3.21) and (3.22). All the simulation results are obtained using MyEclipse 2014 with an average of 20 different runs. We create a JOB class with attributes in terms of UNARRIVE, WAIT, RUN and FINISH to record the task state. We also cre-

ate a SERVER class with attributes in terms of BUSY, IDLE, SLEEP-DELAY and SLEEP to record the VM state. With the help of real implementations, several values of parameters are assumed in the numerical experiments by referencing [16].

The number of VMs is fixed as $n = 20$, the mean service time is presumed to be $(1/\mu = 5$ s), the arrival rate is supposed to be a variable between 0.1 and 3.9 tasks per second. In addition, when investigating the change trend for the energy-saving degree E_v of the system, the presumptions are that $C_a = 20$ mW, $C_l = 2$ mW, $C_s = 4$ mW and $C_t = 12$ mW. However, all the values assumed above can be easily replaced by those from other real systems. They are merely utilized to demonstrate how the mathematical model works. In all of the following figures, the analytical and simulation results are shown by lines and markers, respectively. As seen in Figs. 2 and 3, the analytical and simulation results are in strong agreement.

Figure 2 illustrates how the average latency W of tasks changes in relation to the arrival rate λ of tasks for the different sleep parameters θ and sleep-delay parameters β.

As can be seen from Fig. 2, for the same sleep parameter θ and the same sleep-delay parameter β, as the arrival rate λ of tasks increases, the average latency W of tasks firstly decreases slightly, then stabilizes at a certain value, and finally presents as an uptrend. When the arrival rate of tasks is smaller (such as $\lambda < 1.3$ for $\theta = 0.2$ and $\beta = 1.2$), as the arrival rate of tasks increases, the VMs are less likely to be asleep, so the service of a task is less likely to be delayed in the sleep state, and the average latency of tasks will decrease. When the arrival rate of tasks is moderate (such as $1.3 \leqq \lambda \leqq 2.5$ for $\theta = 0.2$ and $\beta = 1.2$), it is more likely that all the VMs are awake and no tasks are waiting in the system buffer. The latency of a task is only its service time, so the average latency of tasks will remain fixed. When the arrival rate of tasks further increases (such as $\lambda > 2.5$ for $\theta = 0.2$ and $\beta = 1.2$), even though all the VMs are awake, the processing ability of system is not sufficient to cater for the number of tasks in the system. For this case, the higher the arrival rate of tasks is, the more tasks there will be waiting in the system buffer. Thus, the average latency of tasks will increase accordingly.

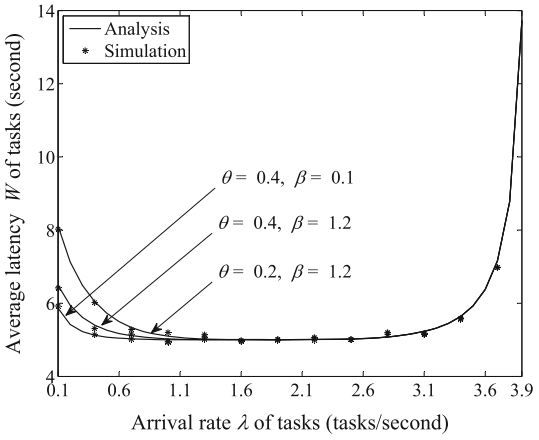

Fig. 2 Change trend for the average latency W of tasks

We also observe that the influences of the sleep parameter θ and the sleep-delay parameter β on the average latency W of tasks are reasonably small for a high arrival rate λ of tasks (such as $\lambda > 1.3$ for $\theta = 0.2$ and $\beta = 1.2$). The reason is that the VMs are more likely to be awake when there is a high arrival rate of tasks. Contrarily, the sleep parameter θ and the sleep-delay parameter β have remarkable influence on the average latency W of tasks for a smaller arrival rate λ of tasks (such as $\lambda \leqq 1.3$ for $\theta = 0.2$ and $\beta = 1.2$). For the same sleep parameter and the same arrival rate of tasks, the average latency of tasks increases as the sleep-delay parameter increases. When the sleep-delay parameter increases, the time length of the sleep-delay period will be shorter, and then the PM is more likely to enter into the sleep state rather than the awake state from the sleep-delay state. The process of a task will be delayed during the sleep state, so the average latency of tasks will increase. For the same sleep-delay parameter and the same arrival rate of tasks, the average latency of tasks decreases as the sleep parameter increases. When the sleep parameter gets larger, the time length of a sleep period gets shorter, i.e., the tasks arriving during the sleep period can be served quicker. So the average latency of tasks will decrease.

Figure 3 illustrates how the energy-saving degree E_v of the system changes in relation to the arrival rate λ of tasks for different sleep parameters θ and sleep-delay parameters β.

As can be seen from Fig. 3, for the same sleep parameter θ and the same sleep-delay parameter β, the energy-saving degree E_v of the system shows a downtrend, decreasing to 0 as the arrival rate λ of tasks increases. When the arrival rate of tasks is lower (such as $\lambda \leqq 1.3$ for $\theta = 0.4$ and $\beta = 1.2$), the VMs are less likely to be asleep as the arrival rate of tasks increases. Note that energy is conserved during the sleep state, so the energy-saving degree of the system will decrease. When the arrival rate of tasks is higher (such as $\lambda > 1.3$ for $\theta = 0.4$ and $\beta = 1.2$), the energy-saving degree of the system stabilizes at 0. For a higher arrival rate of tasks, the VMs are more likely to be awake. There is no energy conservation during the awake state, so the energy-saving degree of the system reduces to 0.

We also observe that the sleep parameter θ and the sleep-delay parameter β have little or no impacts on the energy-saving degree E_v of the system for a higher arrival

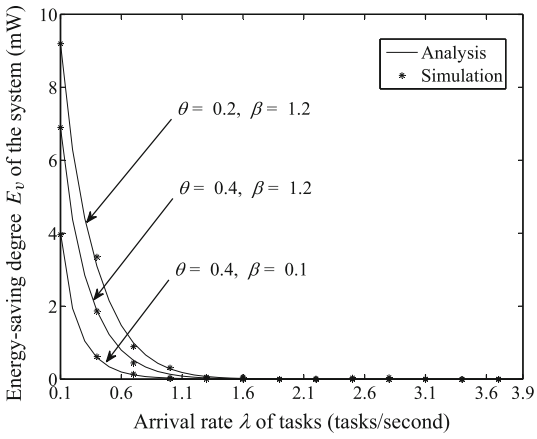

Fig. 3 Change trend for the energy-saving degree E_v of the system

rate λ of tasks (such as $\lambda > 1.3$ for $\theta = 0.4$ and $\beta = 1.2$). As the arrival rate of tasks increases, all the VMs are more likely to be awake all the time, so no energy will be conserved. On the other hand, the sleep parameter θ and the sleep-delay parameter β have remarkable impacts on the energy-saving degree E_v of the system for a lower arrival rate λ of tasks (such as $\lambda \leqq 1.3$ for $\theta = 0.4$ and $\beta = 1.2$). For the same sleep parameter and the same arrival rate of tasks, the energy-saving degree of the system increases as the sleep-delay parameter increases. When the sleep-delay parameter increases, the PM will easily switch into the sleep state from the sleep-delay state, so the energy-saving degree of the system will increase. For the same sleep-delay parameter and the same arrival rate of tasks, the energy-saving degree of the system decreases as the sleep parameter increases. When the sleep parameter becomes greater, the time length of a sleep period gets shorter, which leads to a lower energy-saving degree of the system.

Figures 2 and 3 confirm that a moderate arrival rate of tasks leads to the lowest average latency of tasks, while a lower arrival rate of tasks leads to the highest energy-saving degree of the system. Determining how to regulate the arrival rate of tasks by trading off against the average latency of tasks and the energy-saving degree of the system is an important issue to be addressed.

5 Performance Optimization and Pricing Policy

In this section, we present the Nash equilibrium behavior, the socially optimal behavior and the pricing policy.

5.1 Nash Equilibrium Behavior and Socially Optimal Behavior

In the proposed energy-saving VM allocation scheme, all the tasks make decisions independently to access the system and get service quickly for the purpose of gaining a certain benefit. A task that joins the queue may cause future tasks to spend more time in the system [17]. That is to say, the selfish behavior of all the tasks will lead to a higher arrival rate of tasks. However, a higher arrival rate of tasks will increase the average latency of tasks, and a higher latency of tasks will reduce the individual benefit to the point of creating a negative value. Without loss of generality, the decision to join or balk the system should be made from a social point of view.

In order to investigate the Nash equilibrium behavior [18] and motivate the cloud users to accept the socially optimal strategy, we present some hypotheses as follows:

(1) The number of all the tasks, including the tasks queueing in the system buffer and the tasks being processed in the VMs, is unobservable for newly arriving tasks. But each task has to make a decision to join or balk the system. The tasks are risk neutral, i.e., a decision to join the system is irrevocable, and reneging is not allowed.
(2) The reward of a task for completed service is R. The cost of a task staying in the system is C per unit time. The income of the cloud service provider (CSP)

from energy conservation is U per milliwatt. The benefits for the whole system, including all the tasks and the CSP, can be added together.

(3) In order to assure a positive benefit for an arriving task entering an empty system, we assume that $R > \frac{C}{\mu} + \frac{C}{\theta}$.

The individual benefit of a task is the difference between the reward for completed service and the cost for staying in the system. The individual benefit G_{ind} is then given as follows:

$$G_{\text{ind}} = R - CW. \tag{5.1}$$

The social profit is the aggregation of the individual benefits of all the tasks and the income of the CSP. Thus, the social profit G_{soc} can be calculated as follows:

$$G_{\text{soc}} = \lambda(R - CW) + UE_v. \tag{5.2}$$

By applying the parameters given in Sect. 4, and setting $\theta = 0.3$, $\beta = 1.2$, $R = 10$, $C = 1$ and $U = 1$ as an example, we provide numerical experiments to explore the changing trends of the individual benefit G_{ind} and the social profit G_{soc} in relation to the arrival rate λ of tasks for different service rates μ of tasks in Figs. 4 and 5, respectively.

As can be seen from Fig. 4, for all the service rate μ of tasks, there is a unique arrival rate λ of tasks (marked with "■") subject to $G_{\text{ind}} = 0$. We call the unique value the Nash equilibrium arrival rate λ_e of tasks. When $\lambda < \lambda_e$, the newly arriving tasks will be serviced quickly, so the cost of a task staying in the system is lower than its reward for a completed service. For this case, the individual benefit is positive. When $\lambda > \lambda_e$, the number of tasks waiting in the system buffer increases unbelievably, so the cost of a task staying in the system is higher than its reward for a completed service. For this case, the individual benefit is negative.

We also find that in all the curves of the social profit G_{soc} in Fig. 5, there is a unique arrival rate λ of tasks (marked with "□") subject to the maximum social profit G^*_{soc} for all the service rates μ of tasks. We call this unique value the socially optimal arrival rate λ^* of tasks. When the arrival rate of tasks is smaller (such as $\lambda < 0.4$ for

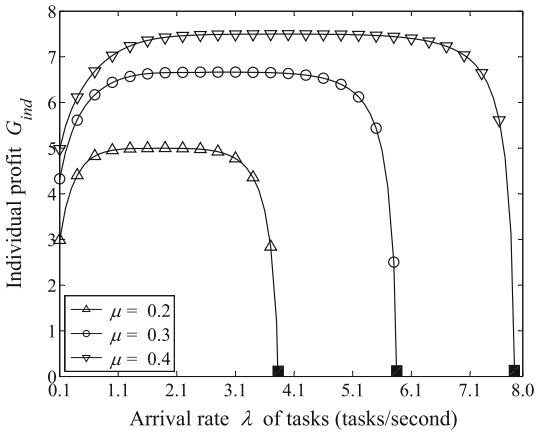

Fig. 4 Change trend for the individual benefit G_{ind}

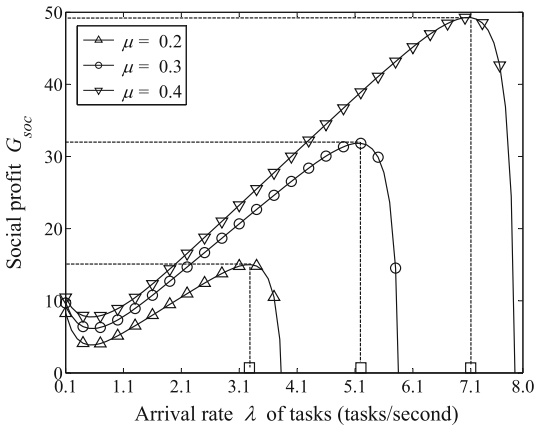

Fig. 5 Change trend for the social profit G_{soc}

$\mu = 0.2$), as the arrival rate of tasks increases, the individual benefit increases, whereas the energy-saving degree of the system decreases. The uptrend of the individual benefit is slower than the downtrend of the energy-saving degree of the system, so the social profit displays a downtrend. When the arrival rate of tasks becomes higher (such as $0.4 \leqq \lambda \leqq 2.5$ for $\mu = 0.2$), as the arrival rate of tasks increases, the individual benefit tends to be fixed after a slow rise, whereas the energy-saving degree of the system decreases continuously, so the social profit generally shows an uptrend. When the arrival rate of tasks increases to an even higher level (such as $\lambda > 2.5$ for $\mu = 0.2$), as the arrival rate of tasks further increases, the individual benefit decreases, whereas the energy-saving degree of the system stabilizes at 0, so the social profit displays a downtrend. Therefore, there is a peak value G^*_{soc} for the social profit with the socially optimal arrival rate λ^* of tasks for all the service rates μ of tasks.

Comparing the numerical results of the individual benefit G_{ind} in Fig. 4 and the social profit G_{soc} in Fig. 5, we find that the Nash equilibrium arrival rate λ_e of tasks is greater than the socially optimal arrival rate λ^* of tasks. That is to say, the Nash equilibrium behavior cannot lead to social optimization. In order to motivate tasks to accept the socially optimal strategy, a reasonable fee should be imposed on tasks to restrain their enthusiasm on receiving the cloud service.

5.2 Pricing Policy

In order to align the Nash equilibrium arrival rate λ_e of tasks with the socially optimal arrival rate λ^* of tasks, we present a pricing policy whereby tasks are charged an appropriate admission fee f.

With the admission fee f, the individual benefit G'_{ind} is modified as follows:

$$G'_{\text{ind}} = R - CW - f. \tag{5.3}$$

Accordingly, the social profit G'_{soc} can be modified as follows:

$$G_{\text{soc}} = \lambda(R - CW - f) + \lambda f + UE_v = \lambda(R - CW) + UE_v. \tag{5.4}$$

Setting $G'_{\text{ind}} = 0$ in Eq. (5.3), we can calculate the admission fee f of tasks as follows:

$$f = R - CW. \tag{5.5}$$

It is arduous work to address the strict monotonicity of the social profit G_{soc} based on Eqs. (5.2) and (5.4). This implies that with the traditional optimization method, it is difficult to obtain the socially optimal arrival rate λ^* of tasks and the maximum social profit G^*_{soc} [19]. For this, we turn to an intelligent optimization algorithm to search for the socially optimal arrival rate λ^* of tasks and the maximum social profit G^*_{soc}.

A genetic algorithm (GA) is a method used to search for a globally optimal solution of objective functions by simulating the natural evolutionary process [20,21]. In a conventional GA, both the crossover probability and the mutation probability are fixed. We note that when these probabilities are higher, the GA will become a random algorithm. Contrarily, when these probabilities are lower, the GA will converge quite slowly. For this reason, we present an improved adaptive GA by dynamically adjusting the crossover probability and the mutation probability.

The following lists the main steps for deriving an improved adaptive GA.

Step 1 Initialize the search space with the upper boundary λ^u and the lower boundary λ^l, the population size N, the minimum and maximum crossover probability P_{cl}, P_{cm}, the minimum and maximum mutation probability P_{ml}, P_{mm}. Set the initial number of evolution generations as $gen = 1$, and the maximum evolution generation as $gen_{\max} = 50$.

Step 2 Initialize the population $S = \{\lambda_1, \lambda_2, \cdots, \lambda_N\}$ randomly within the constraint condition $[\lambda^l, \lambda^u]$.

Step 3 For each individual λ_i, $i \in \{1, 2, \cdots, N\}$, calculate the fitness $G^{\lambda_i}_{\text{soc}}$ the selection probability P_i and the cumulative probability F_i.

$$G^{\lambda_i}_{\text{soc}} = \lambda_i(R - CW(\lambda_i)),$$

$$P_i = \frac{G^{\lambda_i}_{\text{soc}}}{\sum_{i=1}^{N} G^{\lambda_i}_{\text{soc}}},$$

$$F_i = \sum_{j=1}^{i} P_j.$$

% $W(\lambda_i)$ is the average latency of tasks with the arrival rate λ_i.

Step 4 Calculate the crossover probability P_c and the mutation probability P_m.

$$P_c = P_{\text{cl}} - \frac{(P_{\text{cm}} - P_{\text{cl}}) * \text{gen}}{\text{gen}_{\max}},$$

$$P_m = P_{\text{ml}} + \frac{(P_{\text{mm}} - P_{\text{ml}}) * \text{gen}}{\text{gen}_{\max}}.$$

Step 5 Perform the genetic operation to update the population S.

for $j = 1 : N$

slen = selection(S, F_i)

% Select two individuals to cross and mutate.

$S = \text{crossover}(S, \text{slen}, P_c)$
% Cross the selected individuals.
$S = \text{mutation}(S, \text{slen}, P_m)$
% Mutate the selected individuals.
endfor

Step 6 Check the number of evolution generations.
if $\text{gen} < \text{gen}_{\max}$
then $\text{gen} = \text{gen} + 1$, go to **Step 3**
endif

Step 7 Select the optimal individual among the population $\boldsymbol{S}$.

$$\lambda^* = \arg\max_{i \in \{1,\cdots,N\}} \{G_{\text{soc}}^{\lambda_i}\},$$
$$G_{\text{soc}}^* = \lambda^*(R - CW(\lambda^*)).$$

Step 8 Output λ^* and G_{soc}^*.

Applying the parameters used in Sect. 5.1 into the improved adaptive GA, we can obtain the socially optimal arrival rate λ^* of tasks with different service rates μ of tasks. Substituting the socially optimal arrival rate λ^* of tasks into Eq. (3.21), we calculate the average latency W of tasks. Furthermore, we can obtain the admission fee f of tasks with Eq. (5.5). The numerical results of the socially optimal arrival rate λ^* of tasks and the admission fee f of tasks are shown in Table 1. All the values of λ^* and f are exactly to four decimal places.

From Table 1, we observe that the larger the service rate μ of tasks is, the higher the socially optimal arrival rate λ^* of tasks is, and the higher the admission fee f will be. As the service rate μ of tasks increases, the time for a task's service to be completed becomes shorter, and the waiting cost is lowered. This rouses the enthusiasm of tasks to join the system, so the admission fee f should be set higher.

6 Conclusion

In this paper, we focused on how to achieve greener cloud computing under the constraints of a service level agreement. For this purpose, we proposed a novel energy-saving VM allocation scheme with sleep-delay. We presented a method to model and evaluate the proposed scheme by constructing a two-dimensional Markov chain. The proposed model quantified the effects of changes to different sets of parameters, such as the sleep parameter and the sleep-delay parameter. These effects were measured using two important criteria in terms of the average latency of tasks and the energy-saving

Table 1 Numerical results of the pricing policy

Service rate μ	Optimal arrival rate λ^*	Admission fee f
0.20	3.260 6	4.605 8
0.30	5.178 5	6.149 6
0.40	7.031 8	7.008 2

degree of the system. Through theoretical analysis and simulation of statistical experiments, the impacts of the system parameters on the system performance were revealed, and the effectiveness of the proposed scheme was validated. Moreover, from the perspective of economics, we studied the Nash equilibrium behavior and the socially optimal behavior of tasks, and carried out an improved adaptive genetic algorithm to obtain the socially optimal arrival rate. For resolving the deviation between the Nash equilibrium behavior and the socially optimal behavior, we presented a pricing policy tasks to maximize the social profit. Based on this pricing policy, cloud service providers could regulate the behavior of tasks to improve the system performance and optimize the cloud resource. Research into an energy-saving strategy with group synchronization and inter-group asynchronization could be considered as a future extension of this work.

References

[1] Hanini, M., Kafhali, S.: Cloud computing performance evaluation under dynamic resource utilization and traffic control. In: International Conference on Big Data, Cloud and Applications, Boston (2017)
[2] Sugumaran, R., Armstrong, M.: Cloud Computing. Wiley, Hoboken (2017)
[3] Andrews, J., Buzzi, S., Choi, W., Hanly, S., Lozano, A., Soong, A., Zhang, J.: What will 5G be?. IEEE J. Sel. Areas Commun. **32**, 1065–1082 (2014)
[4] Krein, P.: Data center challenges and their power electronics. CPSS Trans. Power Electron. Appl. **2**, 39–46 (2017)
[5] Fard, S., Ahmadi, M., Adabi, S.: Erratum to: a dynamic VM consolidation technique for QoS and energy consumption in cloud environment. J. Supercomput. **73**, 1–4 (2017)
[6] Khoshkholghi, M., Derahman, M., Abdullah, A., Subramaniam, S., Othman, M.: Energy-efficient algorithms for dynamic virtual machine consolidation in cloud data centers. IEEE Access **5**, 10709–10722 (2017)
[7] Cheng, C., Li, J., Wang, Y.: An energy-saving task scheduling strategy based on vacation queueing theory in cloud computing. Tsinghua Sci. Technol. **20**, 28–39 (2015)
[8] Guo, X., Niu, Z., Zhou, S., Kumar, P.: Delay-constrained energy-optimal base station sleeping control. IEEE J. Sel. Areas Commun. **34**, 1073–1085 (2016)
[9] Yang, J., Zhang, X., Wang, W.: Two-stage base station sleeping scheme for green cellular networks. J. Commun. Netw. **18**, 600–609 (2016)
[10] Zhao, Y., Jin, S., Yue, W.: Performance evaluation of the centralized spectrum access strategy with multiple input streams in cognitive radio networks. IEICE Trans. Commun. **E97–B**, 334–342 (2014)
[11] Choi, S., Kim, B., Sohraby, K., Choi, B.: On matrix-geometric solution of nested QBD chains. Queueing Syst. **43**, 5–28 (2003). https://doi.org/10.1023/A:1021884213344
[12] Kim, S., Kim, M., Kang, C.: Performance of a burst switching scheme for CDMA-based wireless packet data systems. IEICE Trans. Commun. **E86–B**, 1082–1093 (2003)
[13] He, H., Yuan, D., Hou, Y., Xu, J.: Preconditioned Gauss–Seidel iterative method for linear systems. Int. Forum Inf. Technol. Appl. **1**, 382–385 (2009)

[14] Ma, Z., Wang, P., Yue, W.: Performance analysis and optimization of a pseudo-fault Geo/Geo/1 repairable queueing system with N-policy, setup time and multiple working vacations. J. Ind. Manag. Optim. **13**, 1467–1481 (2017)
[15] Bhagat, A., Jain, M.: N-policy for M^x/G/1 unreliable retrial G-queue with preemptive resume and multi-services. J. Oper. Res. Soc. China **4**, 1–23 (2016)
[16] Jin, S., Ma, X., Yue, W.: Energy-saving strategy for green cognitive radio networks with an LTE-advanced structure. J. Commun. Netw. **18**, 610–618 (2016)
[17] Hassin, R., Haviv, M.: To Queue or not to Queue: Equilibrium Behaviour in Queueing Systems. Springer, Boston (2003)
[18] Jing, H., Aida, H.: A graphical game theoretic approach to optimization of energy efficiency in multihop wireless sensor networks. IEICE Trans. Commun. **E99–B**, 1789–1798 (2016)
[19] Yu, H., Zeng, W., Wu, D.: A Stochastic level-value estimation method for global optimization. J. Oper. Res. Soc. China **1**, 1–16 (2017)
[20] Portaluri, G., Giordano, S.: Power efficient resource allocation in cloud computing data centers using multi-objective genetic algorithms and simulated annealing. In: IEEE International Conference on Cloud Networking, pp. 319–321 (2015)
[21] Salido, M., Escamilla, J., Giret, A., Barber, F.: A genetic algorithm for energy-efficiency in job-shop scheduling. Int. J. Adv. Manuf. Technol. **85**, 1–12 (2016). https://doi.org/10.1007/s00170-015-7987-0