Cluster Comput (2015) 18:1595–1607
DOI 10.1007/s10586-015-0484-2

Random task scheduling scheme based on reinforcement learning in cloud computing

Zhiping Peng[1] · Delong Cui[1] · Jinglong Zuo[1] · Qirui Li[1] · Bo Xu[1] · Weiwei Lin[2]

Received: 28 November 2014 / Revised: 28 August 2015 / Accepted: 30 August 2015 / Published online: 11 September 2015

Abstract Task scheduling is a necessary prerequisite for performance optimization and resource management in the cloud computing system. Focusing on accurate scaled cloud computing environment and efficient task scheduling under resource constraints problems, we introduce fine-grained cloud computing system model and optimization task scheduling scheme in this paper. The system model is comprised of clearly defined separate submodels including task schedule submodel, task execute submodel and task transmission submodel, so that they can be accurately analyzed in the order of processing of user requests. Moreover the submodels are scalable enough to capture the flexibility of the cloud computing paradigm. By analyzing the submodels, where results are repeated to obtain sufficient accuracy, we design a novel task scheduling scheme based on reinforcement learning and queuing theory to optimize task scheduling under the resource constraints, and the state aggregation technologies is employed to accelerate the learning progress. Our results, on the one hand, demonstrate the efficiency of the task scheduling scheme and, on the other hand, reveal the relationship between the arrival rate, server rate, number of VMs and the number of buffer size.

Keywords Task scheduling · Cloud computing · Queuing theory · Reinforcement learning · State aggregation

✉ Delong Cui
delongcui@163.com

Zhiping Peng
pengzp@foxmail.com

Jinglong Zuo
oklong@gmail.com

Qirui Li
liqirui@foxmail.com

Bo Xu
xubo807127940@163.com

Weiwei Lin
linww@scut.edu.cn

1 College of Computer and Electronic Information, Guangdong University of Petrochemical Technology, Maoming, China

2 School of Computer Science and Engineering, South China University of Technology, Guangzhou, China

1 Introduction

Cloud computing is an outcome of combining the conventional computer with network technologies, i.e. distributed internet computing, parallel computation, utility computing, network storage technologies, virtualization, load balance, high availability, and others [1]. Distinct from traditional network server platforms, cloud computing offers an on-demand service model, available, convenient, on-demand, accessing a configurable computing resource pool, complete with network, server, memory, application software, services, and others. With little management or a slight interaction with the service providers satisfied, all these resources can be offered with great efficiency. Based on the service provisioning at different levels, three cloud service models have been proposed [2,3], namely Infrastructure-as-a-Service (IaaS), Platform-as-a Service (PaaS), and Software-as-a-Service (SaaS).

Due to large-scale servers, heterogeneous and diverse resources, extensive user groups, various types of application jobs and varied QoS target constraints, cloud computing systems are always dealing with massive jobs and data [4]. Under such a background, how to allocate and manage resources rationally and how to schedule the jobs efficiently in order that the massive jobs are able to be dispatched with a low cost expenditure in a relatively shorter time. Meanwhile, it has become a growing concern and one of the technical difficulties in the academic field to guarantee a high utilization of resources in cloud computing system and balance the overall loads [5].

Virtualization technology supports the physical resources shared by the logical independent multi-applications located in the same computer node. Therefore, virtualization provides a feasible solution to improve node utilization. However, the assemblage configuration scheme of most data centers is computed according to the peak demands of virtual machines (VMs), far more than ordinary demands. As a result, it remains difficult to guarantee the resource utilization by using virtualization technology [6–8], while most of them were hard to extend and can only be used in small-scale groups.

In addition when evaluating the cloud computing platform performance such as response time, a key requirement in an Service-Level-Agreement (SLA) for cloud users, most existing works adopt queuing models or other stochastic models, and use the expected response time such as the metric. Meanwhile, the major Cloud service providers (e.g. Amazon EC2 [9], Microsoft Azure [10], etc.) commonly gets a charge of resource usage at an hourly rate. Therefore, no matter what aspects are focused on by various cloud computing models, it is a key issue to optimize the response time respectively at certain occupancy rates. Although the methods in current literature are mostly modeling the whole cloud computing platform into singular queuing models, these monolithic analytical models are restrictive in terms of extendibility, simplicity and cost of solution.

Hence, an in-depth look at the scheduling strategies is not only of theoretical value, but also of practical significance, particularly in the commercial services based on cloud computing model. In this paper, we conduct detailed research on user task scheduling of cloud computing services. Main contributions can be summarized in the following areas: First, we design various functional submodels at different servicing stages in a complex cloud computing center. Secondly, the performance analysis of response time is obtained by queuing theory. Thirdly, user task optimization scheduling is achieved by using reinforcement learning strategies. Fourthly, resources adaptation adjustment to workload and system dynamics are both taken into consideration.

The remainder of this paper is organized as follows: Sect. 2 reviews the related work; Sect. 3 explores the system models, including task schedule submodel (TSSM), task execute submodel (TESM) and task transmission submodel (TTSM); based on the proposed models, we optimize task scheduling based on reinforcement learning and accelerate the learning progress by using state aggregation technology in Sects. 4; 5 presents extensive performance evaluations; and finally, in Sect. 6, we reach the conclusions and skeleton of our future work.

2 Related works

Cloud computing is an area of research which has attracted much attention, but only a few among those who do this work use rigorous analytical approaches so far addressing the performance issues. When evaluating the service delay performance, a key requirement in SLA for Internet-Data-Center (IDC) operations, most existing works adopt queuing models or other stochastic models, and use the expected service delay as the metric. According to the different focus points, the research achievements using queuing theory in literature can be classified as follow

2.1 Theoretical research

The simplest model of the cloud computing platform was M/M/1 queuing system. Because of the dynamic internet traffic and the dynamic virtual machine environment where the system tended to run, this kind of cloud computing system model ignored too many system details. However, it served as the baseline and starting point of other cloud computing queuing models. In [11], some key performance indicators, such as mean server rate and distribution of response time, could be obtained by considering fault recovery situations. Assuming the inter-arrival and service times in the system were both exponentially distributed with a fixed buffer size $m+r$, a cloud could be modeled as an $M/M/m/m+r$ queuing system. But according to the authors' own argument, the components of response time such as waiting time, service time, and execution time, independent and identically distributed (i.i.d) random, were unrealistic. An improved work had been done by [12], the authors employed a Markov request queue model to assess the performance indicators such as waiting time and completion time obtained by considering the resources shared among VMs and various types of failures. However, the buffer size of the subtask was not considered, which was unrealistic in the cloud computing platform.

The statistical results indicated that inter-arrival time and/or service time did not commonly satisfy exponential distribution. This domain had been explored in depth by Khazaei et al. [13–15] in the context of queuing theory.In [13], the author assumed that the service time is i.i.d random variables, then the cloud computing center was modeled as an $M/G/m/m+r$ queuing model. So the probability distribution of the request response time and the relationship between the server number and input buffer size were obtained by embedded Markov chain. The extended related work can be seen in [14, 15].

2.2 Energy management

In [16], the authors focused on the power management of internet data center for minimizing the total electricity cost under the multiple electricity markets environment. Modeled as an $M/M/n$ queues system, the data center considered the total electricity cost, the workload constraint and client end-to-end delay constraint respectively, and the total electricity cost minimization was formulated as a constrained mixed-integer programming issue, solved by employing Brenner's fast polynomial-time algorithm [17]. The extended related work can be seen in [18,19].

In [20], the authors addressed making distributed routing and server management decisions in large-scale, geographically distributed data centers. The data center was modeled as m parallel structure and interplay queuing system. Based on the system model, the authors put forward a two-time-scale control algorithm aiming at facilitating power cost reduction and power consumption versus delay trade-off techniques, extending the traditional Lyapunov approach to optimize it.

2.3 Resource allocation

In [21], the authors dealt with the combined issues of power and performance management in cloud data centers, and proposed a dynamic resource management scheme by leveraging both of the techniques such as dynamic voltage/frequency scaling and server consolidation, thus to achieve energy efficiency and desired application-level performance. The novelty of the proposed scheme was its integration with timing analysis, queuing theory, integer programming, and control theory techniques. In [22], despite the varying event arrival rates, a queuing theory based approach was pursued to achieve specified response time target; by drawing the necessary computing resources from a cloud, a distinct query engine was modeled as an atomic unit to predict response times. Several similar units hosted on a single node were modeled as a multiple class M/G/1 queuing system and the response times were deemed to meet specified targets although being subject to varying event arrival rates over time. Correlation work also extend to multimedia cloud and large web server clusters [23,24].

There was also some research done regarding the use of reinforcement learning for some special applications in cloud computing platform. Reinforcement learning was first introduced to the autonomic resource allocation in cloud computing environment [25], and two novel ideas were presented: (1) a predetermined policy was used in the initial learning period and (2) an approximation to the Q-function servers as a neural network [26].

In [27], the authors developed a general reinforcement learning framework with models for classes of jobs (best-effort availability and responsiveness), for objective functions, and for the infrastructure. In [28], the author proposed a Q-learning-based leasing policy for a scientific workload in academic data centers, which extended the local resource space to balance operating costs and service quality. In [29], the authors proposed a reinforcement learning based approach to autonomic configuration and reconfiguration of multi-tier web systems. Experiment results demonstrated that this approach could auto-configure web system dynamically not only to the change of workload, but also to that of virtual machine resources.

Distinguished from prior works, a user task scheduling model combined with queuing theories and reinforcement learning based on cloud computing environment is proposed in this paper. By introducing interacting analytical submodels [30], the model includes the crucial features of cloud centers, namely the batch arrival of user requests, the resource virtualization, and the realistic servicing steps. As a result, the important performance metrics, such as task blocking probability and total waiting time incurred on user requests, are obtained.

3 System models

As mentioned in Sect. 2, due to high dynamics and complexity of cloud computing platform, the proposed singular queuing models are hard to analyze and extend. In this paper, we use the model [30] for reference, redesign and introduce interacting analytical submodels, as shown in Fig. 1, the complexity of system is divided into submodels so that they can be accurately analyzed in order of processing of users' requests. Moreover, the submodels are scalable enough to capture the flexibility (on-demand services) of the cloud computing paradigm. The model proposed in this paper incorporates task schedule submodel (TSSM), task execute submodel (TESM) and task transmission submodel (TTSM). The submodles are interactively functioning so that the output of one submodel is the input of the others and vice versa. The construction and function of each submodel will be discussed in detail below.

3.1 TSSM

TSSM is constructed by a finite-buffer-size users global task queue and a task scheduler. In cloud computing environment, users submit task requests and receive execution results by computer network. Users global task queue accepts users task requests and lines them up into a global finite queue in the order of task request arrivals first in first out, FIFO. Task dispatcher schedules users task requests to a designated computing server (VM). The amount of cloud users is large, but it is a low probability that each user submits task requests at the same time, so the task requests arrival can be modeled as a Poisson process [31,32] with mean arrival rate λ^{tssm}.

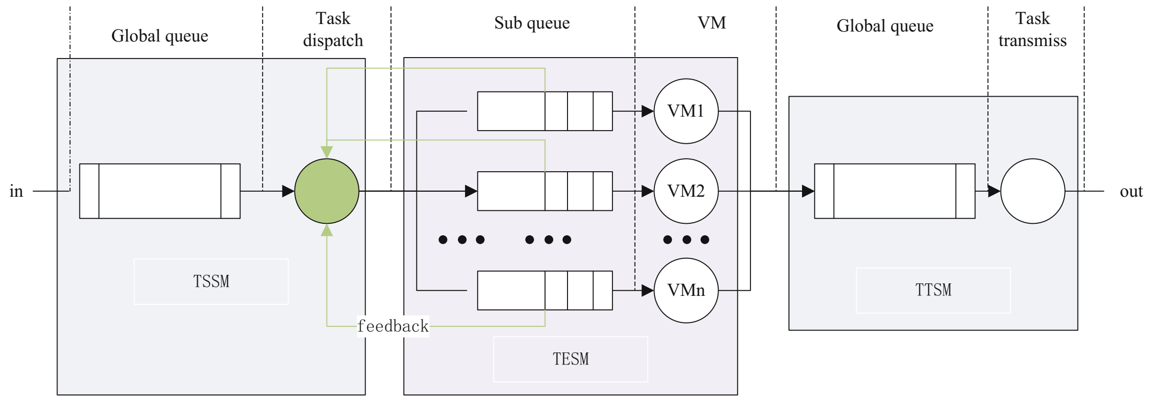


Fig. 1 Queuing model of cloud computing platform

Assuming task dispatcher with a mean service rate μ^{tssm}, and the global queue adopts total acceptance policy, we model TSSM as an M/M/1 queuing system. In the TSSM, the mean response time $rt^{tssm} = \frac{1/\mu^{tssm}}{1-\lambda^{tssm}/\mu^{tssm}}$ can be obtained when the condition $\lambda^{tssm} < \mu^{tssm}$ is satisfied so as to maintain a stable queue.

In this paper, we proposed a novel task scheduling scheme based on reinforcement learning (RL) to actualize users requests optimization scheduling in cloud computing environment. To the best of our knowledge, the scheme would be the first one to combine queuing theory and RL principle to users tasks scheduling in this environment. The details of the task scheduling scheme will be described in detail in Sect. 4.

3.2 TESM

As shown in Fig. 1, the kernel of TESM is the parallel connection structure with a set of computation servers, each of which is formed by a subtask buffer queue and a VM. The workflow of each computing server can be described as: first, tasks dispatcher allots each users request to a designated buffer queue; secondly, the VM takes out users requests from the corresponding buffer queue and offers it to the computing server; finally, VM transports the execution results to TTSM.

In this paper, we assume that all VMs are homogeneous and each buffer size is m, so the TESM can be modeled as a parallel connection structure with a series of $M/M/1/m$ queuing system. As the service processing of the multiple computing queues is the same, the probability of a request assigned to the ith computing queue for service is denoted by p_i. The conditions $\lambda_i^{tesm} = p_i \lambda^{tssm}$ and $\sum_{i=1}^{N} p_i = 1$ are satisfied in order to keep a stable queue. Assuming VM with a mean service rate is μ^{tssm}, the mean response time to user requests in the ith computation queue is denoted by $rt_i^{tssm} = \frac{1/\mu^{tssm}}{1-\lambda_i^{tssm}/\mu^{tssm}}$. On the possibility that a request may be allocated to any one computing queue, the equilibrium response time is given by $rt^{tesm} = \sum_{i=1}^{n} p_i rt_i^{tesm} = \sum_{i=1}^{n} p_i \frac{1/\mu^{tssm}}{1-\lambda_i^{tssm}/\mu^{tssm}}$.

3.3 TTSM

TTSM receives execution results from TESM, and transmits them back to the requesting clients. TTSM is constructed by a global results queue and a task transmitter. Results queue receives the execution results from VMs and lines up all results in the global finite queue in the order of executed results arrivals (FIFO). Task transmitter takes out the executed results from results queue and transports them back to requesting users. The mean arrival rate λ^{ttsm} at the results queue is equal to λ^{tssm} in that no requests are dropped in TTSM. Assuming task transmitter with a mean service rate is denoted by μ^{ttsm}, and $\lambda^{tssm} < \mu^{tssm}$ is satisfied, similar to TSSM, the TTSM can also be modeled as an M/M/1 queuing system, where the mean response time can be formulated as $rt^{ttsm} = \frac{1/\mu^{ttsm}}{1-\lambda^{tssm}/\mu^{tssm}}$.

Thus the total mean response time in cloud computing environment can be given as $rt^{tot} = rt^{tssm} + rt^{tesm} + rt^{ttsm}$.

4 Task scheduling based on RL

Tasks dispatcher is a controller of tasks execution sequence and resource allocation. A suitable tasks dispatcher is not only to reduce system response time but also to improve resource utility and throughput. Therefore, it is a core issue to employ appropriate tasks scheduling scheme in the cloud computing platform. One example is the famous Google cloud computing platform—the research areas of Hadoop task scheduling algorithm are mainly classified as follows: the data localization, the infer mechanism, the shuffle, the heterogeneous environment and others.

In this section, we will elaborate the proposed task scheduling scheme by employing RL to users request scheduling in cloud computing environment. The contents are organized as follows: Sect. 4.1 presents an overview of Reinforcement Learning; Sect. 4.2 describes the proposed task scheduling scheme in detail; And in Sect. 4.3, the method to accelerate learning progress is illustrated.

4.1 RL

Reinforcement learning (RL) is a general class of algorithms in the field of machine learning that aims at allowing an agent to learn how to behave in an environment, where the only feedback consists of a scalar reward signal. The concept of RL is seen as a learning paradigm, in which an agent is supposed to determine the appropriate action at a set of states and several actions for each state. When the processing enhanced with repeated steps, the problem is known as a Markov decision process (MDP), which is denoted by a state transition probability function $P_a(s, s') = P_r(s_{t+1} = s'|s_t = s, a_t = a,)$ and a reward function $R_a(s, s') = E(r_{t+1}|s_t = s, a_t = a, s_{t+1} = s')$.

When transiting between states, the learning agent apperceives its current states $s_t \in S$ and the available action set $a_t \in A(s_t)$, then transits to next state s_{t+1} by taking action $a_t \in A(s_t)$ and receives an immediate reward r_t from the environment. The action selection decision is determined not only by the immediate reward, but also by the future rewards the following states would yield [33,34]. In this paper, we use Q-learning to actualize the user request schedule. The Q-value function of taking action a in state s can be defined as:

$$Q(s, a) = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} \tag{1}$$

where $0 < \gamma < 1$ is a discount factor helping value function convergence. We can get the unique optimal value function $Q^*(s, a)$ and define it as the solution to the below equation:

$$Q^*(s, a) = \sum_{s_{t+1} \in S} P_a(s_t, s_{t+1}) \left(R_a(s_t, s_{t+1}) + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})\right) \tag{2}$$

where s_{t+1} and a_{t+1} are the next state and action respectively.

The agent aims to set up a policy $\pi : S \rightarrow$ A to maximize the collected cumulative rewards in the long term through the repetition of trial-and-error interactions. Thus the optimal policy π^* can be defined as below, maximizing expected reward from any initial state.

$$\pi^*(s) = \arg\max_a \left(R_a(s_t, s_{t+1}) + \gamma \sum_{s_{t+1} \in S} P_a(s, s_{t+1}) Q^*(s_{t+1}, a_{t+1})\right) \tag{3}$$

4.2 Task schedule scheme

As mention in Sect. 3, TSSM can be modeled as an M/M/1 queuing system. Tasks dispatcher actualizes user requests scheduling from global queue to subtask queue. At the decision time T_i (the moment of ith request arrival at TESM), assuming the number of deployed VMs is N, and the buffer size of each VM is M, the remainder buffer memories of each VM are denoted by s_i, so the allocation remainder capacity of task dispatcher is $0 \leq \sum_{i=0}^{N} s_i \leq M{*}N$. The workflow of task dispatcher at the decision time T_i can be described as follow: First, task dispatcher makes a scheduling policy according to the parameters such as the allocation remainder capacity s_i of each VM, pre-tasks execute situation in VMs, predict execute time of current task, etc; secondly, ith user request is placed to designated subtask buffer queue; finally, tasks dispatcher update the allocation remainder capacity and wait the next user request arrival.

When in the entire operation process of the cloud computing system, the computer server will execute the user's requests continuously until the buffer queue is empty, so the remaining capacity of each computing server buffer queue in resource pool is changing in real time. Therefore, the optimal objective of the task dispatcher is to schedule the tasks efficiently in association with multiple computing servers, thus to reduce the response time and facilitate the quality of service for the user.

The solution can be mathematically formulated as

$$\begin{array}{l}
\underset{\left\{0 \leq \sum_{i=0}^{N} s_i \leq M*N\right\}}{\text{Minimize}} \quad rt^{tot} \\
\textit{subject to} \\
\quad 0 < \lambda^{tssm} < \mu^{tssm} \\
\quad 0 < \lambda^{tssm} / \mu^{tssm} < 1, \forall i = 1, \ldots, N \\
\quad \sum_{i=1}^{n} p_i \lambda_i^{tesm} < \mu^{tesm}, \forall i = 1, \ldots, N \\
\quad 0 < \sum_{i=1}^{n} p_i \lambda_i^{tesm} / \mu^{tesm} < 1, \forall i = 1, \ldots, N \\
\quad 0 < \lambda^{ttsm} < \mu^{ttsm} \\
\quad 0 < \lambda^{ttsm} / \mu^{ttsm} < 1 \\
\quad \left(rt^{tssm} + rt^{tesm} + rt^{ttsm}\right) \leq SLA \\
\quad \sum_{i=1}^{n} p_i \lambda_i^{tesm} \leq \lambda^{tssm}, \forall i = 1, \ldots, N
\end{array} \tag{4}$$

The task scheduling problem can be cast as a MDP, where state space is denoted by S, action set A, and immediate reward function $r(s, a)$.

State Space. As for the task scheduling problem, a state can be defined as the remainder of configuration capacity of i^{th} VM, denoted by s_i. For the cloud computing platform, the state space can be represented by a vector in the form as: $s_i = (s_1, s_2, \ldots, s_n)$.

Action Set. For the ith user request, we define it as action space $(0/1)_i^j$ which means the ith user request is assented to jth VM. For example, the action space can be represented by a vector in the form of $a_i = (0, 1, 0, \ldots, 0)_i^2$, which indicates current user's request (ith user request) is assigned to 2th VM.

Immediate Reward. The immediate reward is used to reflect the correct running state and the efficiency of task scheduling. The two situations of queuing theory are considered by designing a reward function. (1) If the buffer size is M, the user request tasks will be executed immediately without waiting and has the minimized response time; (2) even though the subtask queue has many buffer sizes, if the current user task running in VM has a long execution time, the tasks in this queue will have a long waiting time and response time. The agent receives the reward $r$at decision time t,given as

$$r = \begin{cases} 1 \quad if \quad wt < \overline{wt} \quad \& \quad s_i = \max(s_i) \\ 0 \quad if \quad wt < \overline{wt} \\ -1 \quad otherwise \end{cases} \tag{5}$$

where $\overline{wt}$ is the mean waiting time of recent user requests. For a given users task, task dispatcher will receive a positive reward 1 if the current task is scheduled to the VM which has the remainder maximum buffer memories and the waiting time less than the mean of recent n user tasks; if only the condition of waiting time is less than the mean waiting time, the reward is 0; otherwise task dispatcher will receive a demerit of -1.

We employ the Q-learning method as our optimization scheduling algorithm. By continuous interactions with the environment and tryouts, Q-learning evaluates the feedback from the environment to optimize future decision-making. Characterized by on-line learning, modeling-free, with the greatest long-term cumulative reward instead of immediate one, it has become one of the most important methods in reinforcement learning, especially for the learners who know little about the surroundings, and self-adapt to learning in a dynamic, complex environment. In light of the continuously cumulative formats, after each immediate reward r is collected, the mean Q-value of an action a on state s, denoted by $Q(s, a)$, can be refined at once:

$$Q(\mathrm{s}_t, \mathrm{a}_t) = Q(\mathrm{s}_t, \mathrm{a}_t) + \alpha * \left[\mathrm{r}_{t+1} + \gamma * Q(\mathrm{s}_{t+1}, \mathrm{a}_{t+1}) - Q(\mathrm{s}_t, \mathrm{a}_t)\right] \tag{6}$$

where α is a learning rate parameter that facilitates convergence to the true Q-values in the presence of noisy or stochastic rewards and state transitions [35], and the discount rate is denoted by γ to guarantee the updated reward convergence in continuing task. The pseudo code of the Q-value learning algorithm is illustrated in Algorithm 1.

Algorithm 1 Q-value Learning Algorithm

```
1:  Initialize Q table
2:  Initialize state $s_t$
3:  error = 0
4:  repeat
5:    for each state $s$ do
6:      $a_t = get\_action(s_t)$ using $\varepsilon - greedy$ policy
7:        for (step=1; step<LIMIT; step++) do
8:          Take action $a_t$ observe $r$ and $S_{t+1}$
9:          $Q_t = Q_t + \alpha * (r + \gamma * Q_{t+1} - Q_t)$
10:         $error = MAX(error \mid Q_t - Q_{previous-t})$
11:         $s_t = s_{t+1}, a_{t+1} = get\_action(s_t), a_t = a_{t+1}$
12:       end for
13:   end for
14: until $error < \theta$
```

4.3 State aggregation

However, for the coordinate optimization problem in a multiple VMs system, the state space will grow exponentially or geometrically as the number of stations and the capacity of buffer increase. As a result, the learning process will suffer from the issue of dimensionality, which may have a negative influence on convergence speed and optimized value. For a remedy, we employ state aggregation technologies to accelerate the learning progress.

Definition abstract function $\phi : S \to E_i$, $\forall s_i \in S$, $\phi(s_i) \in E_i$ is an abstract state aggregation. The inverse function of abstract function $\{\phi^{-1}(e_i)|e_i \in E_i\}$ divide original state space S_i into sub category E_i, that is abstract state space. The definition of abstract function is

$$E = \begin{cases} S_v \quad if \quad s_i = 0 \\ S_l \quad if \quad s_i \in \left(1, \frac{M}{3}\right) \\ S_m \quad if \quad s_i \in = \left(\frac{M}{3} + 1, \frac{2M}{3}\right) \\ S_h \quad if \quad s_i \in = \left(\frac{2M}{3} + 1, M - 1\right) \\ S_f \quad if \quad s_i = M \end{cases} \tag{7}$$

subscript v, l, m, h, f indicate the state of remainder buffer memories is full, less, middling, more, vain, respectively.

After state aggregation, abstract state space is $E_i = \{S_v, S_l, S_m, S_h, S_f\}$. For ith users request, the action that the tasks dispatcher will take is only related to state aggregation. Obviously, there are two abstract states in which the action the tasks dispatcher will take is special.

1. queuing vacant state S_v. Tasks dispatcher rejects to schedule the users request tasks to the VM whose abstract state is S_v, so the action is $a_i(S_v) = 0$.

2. queuing full state S_f. Tasks dispatcher prior scheduling user request task to the VM which abstract state is S_f, so the action is $a_i(S_f) = 1$.

The task requests will be assigned in priority to the dispatchers with full remaining buffer queue. For ith user task, the optimization policy of tasks dispatcher is denoted as $v_i(a(s_v), a(s_l), a(s_m), a(s_h), a(s_f))$, and the state transmit denoted as $(e_i(t_i), a_i(e_i(t_i)), e_i(t_{i+1}))$. For a respective sample observation $< s_i(T_n), a_i(e_i(t_i)), s_i(T_{n+1}), w_i(T_n), \mu_i(T_n) >$, is the interval between two consecutive decisions, and the reward is $r(e_i(t_i), a_i(e_i(t_i)), e_i(t_{i+1}))$.

The pseudo code of our state aggregation Q -value learning algorithm is presented in Algorithm 2.

Algorithm 2 The state aggregation Q-value learning algorithm

```
1: Initialize State aggregation $\phi : S_i \rightarrow E_i$
2: Initialize $Q$ table
3: error = 0
4: repeat
5:   for each abstract state $E_i$ do
6:     Update $Q$-value table using Algorithm 1
7:   end for
8: until $error < \theta$
```

5 Performance evaluation

To evaluate the efficiency of our approach, implementations have been performed on the simulation and real cloud computing environment, respectively.

5.1 Simulation cloud computing environment experiment results

Using MATLAB R2012a by MathWorks Inc. [36], we have developed a discrete event simulator of the cloud server farm to validate the efficiency of task scheduling solution and have compared the performance information among the below alternative schemes in our simulations: (1) the proposed task scheduling scheme, denoted by Q-sch, in which the users request tasks are optimally scheduled to the VMs by reinforcement learning; (2) random scheduling scheme, denoted by Ran-sch, in which the users request tasks are scheduled to the VMs by random; (3) equal scheduling scheme, denoted by Equ-sch, in which the users request tasks are scheduled to the VMs in order; (4) mix-scheduling, denoted by Mix-sch, in which the current users request task is first scheduled to the VM random, if the remaining buffer memorizes of the VM is 0, then rescheduling the task to the VMs by maximizing the remaining buffer memory.

The above scheduling schemes can be classified into two types: the static scheduling scheme and the dynamic scheduling scheme, according to the relationship between user requests ID and VMs ID. Equ-sch is a static task scheduling scheme because once the user request ID is ascertained, it will be executed on which VM is also ascertained. The other scheduling schemes are dynamic schemes in contrast.

5.1.1 Comparison of response time when various arrival rate

A comparison of response times with the arrival rates ranging from 10 requests/s to 20 requests/s is demonstrated in Fig. 2, from which we can generalize the following conclusions: (1) the response time of every scheduling scheme increases with the increase of arrival rates; (2) the response time of every scheduling scheme trickles up convergence, and (3) compared to the other schemes under the same server rate constraint, the proposed scheduling scheme obtains a much shorter response time.

Guided by the observation, we find that the proposed scheduling scheme efficiently reduces the response time and facilitate the quality of server (as shown in Fig. 2, such as when the arrival rate is 10 requests, the response reduces by 20 %). The results also reveal when response time of cloud computing platform is close to the baseline of SLA, most of the scheduling schemes cannot guarantee the system performance.

Figure 3 gives a close look into the response time in the four schemes when λ reaches 40 requests and server rate μ reaches 200 requests. From the simulation comparison results under the same arrival rate and server rate constraint, we gain further insights into the reasons behind Fig. 2. The random scheduling scheme dispenses user task request to running VM, which fails to consider the balanced workload of various VMs, thus taking the longest response time. Meanwhile, the equal scheduling scheme only guarantees the balanced workloads of various VMs. However, when the interval of tasks arrival is short and remaining buffer size of each VM is full, the workload balance of various VMs cannot be solved yet. The mix-scheduling scheme optimizes the random scheduling scheme, while giving performance similar to Ran-buffer scheme under these system parameters. The proposed task scheduling scheme reckons the remaining buffer memory and execution time of the current tasks in VMs, so it can find the dynamic optimal solution so to minimize the response time. At this moment, the response time of Ran-sch, Mix-sch, Q-sch and Equ-sch is 1903, 1915, 1868, 1947 ms respectively.

Conclusion, although Ran-sch, Mix-sch and Equ-sch belong to different kinds of scheduling scheme, the execution results of these schemes, as shown in Fig. 4, can allocate running time and workflow among VMs, while they cannot adjust the utilization rate and workflow balance dynamically,

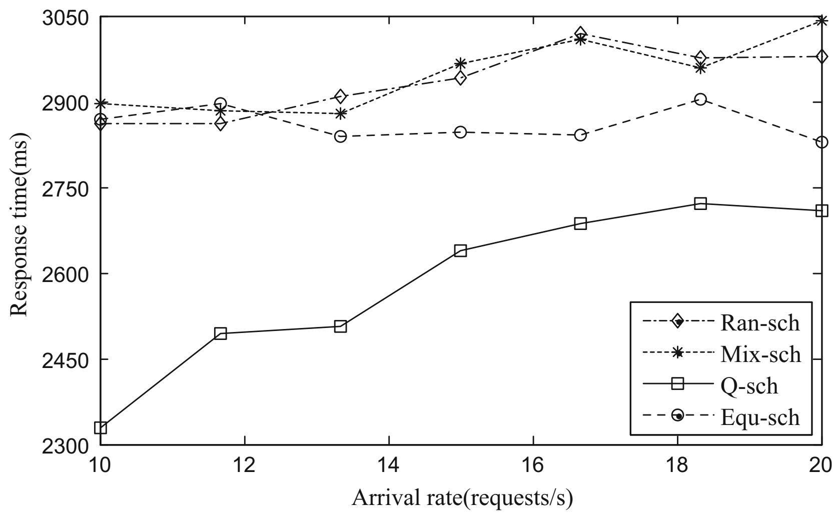


Fig. 2 Simulation comparison results of response time when the arrival rate λ varies from 10 to 20 requests/s under the same server rate constraint

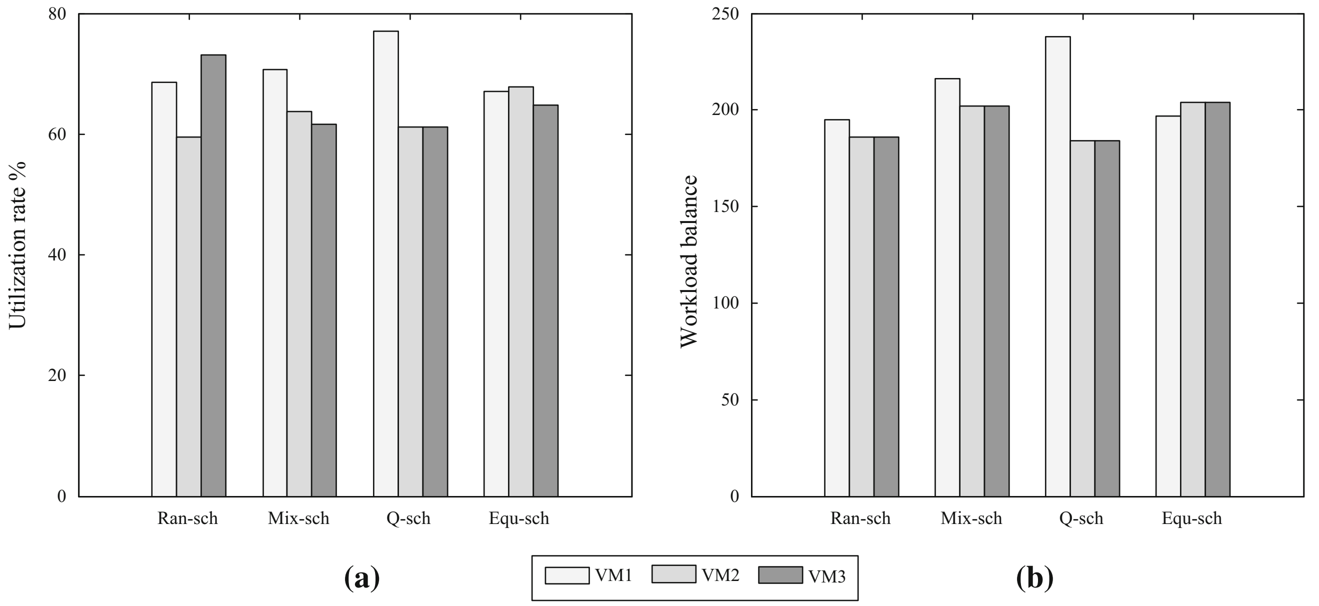


Fig. 3 Simulation comparison results under the same arrival rate and server rate constraint: **a** comparison of VM resources utilization rate response time when arrival rate $\lambda = 20$ requests and server rate $\mu = 200$ requests. **b** Comparison of VM workload balance when with the same arrival rate and server rate in Fig. 4a

resulting in a higher response time. In contrast, Q-sch according to the remaining buffer sizes and execution time of the current tasks in VMs, avoids the situation of shorter queue with longer wait time, thus reducing the response time.

5.1.2 Comparison of response time when server rates vary

Figure 4 illustrates a comparison of response time when the server rates range from 1 request to 5 requests. We can obtain two conclusions from the observation: (1) the response times of various scheduling scheme lines reduce with the increasing server rates; (2) despite of the various server rates, the response time discrepancy of each scheduling is nearly invariant. Compared with the other scheduling scheme, the response time of the proposed scheme reduces nearly 800 ms under the same arrival rate constraint. The reason has been discussed in detail in Sect. 5.1.

5.1.3 Comparison of response time with various buffer memory sizes

Observed from Fig. 5, the response time rises rather evenly with buffer size with the number of VMs constraint. Khazaei et al. [14] demonstrates that when the number of servers is large, the buffer size exerts a virtually undetectable impact. The resulting evidence of the experiment in this paper demon-

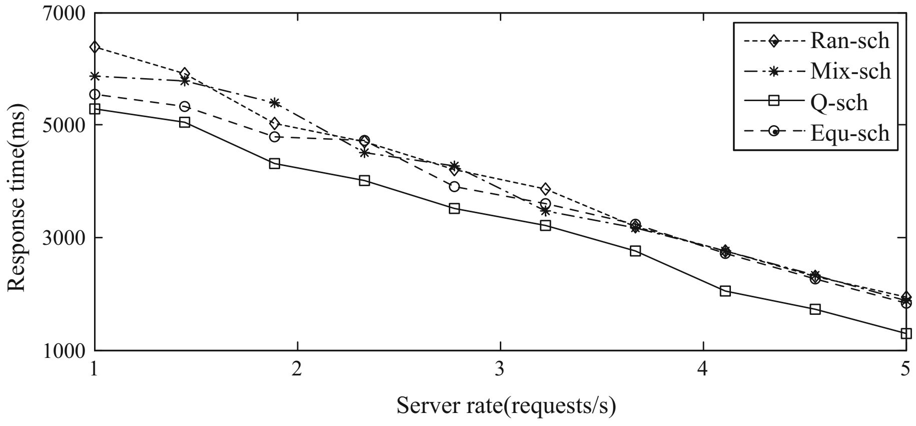


Fig. 4 Simulation comparison results of response time when server rate μ varies from 1 request to 5 requests and the same arrival rate constraint

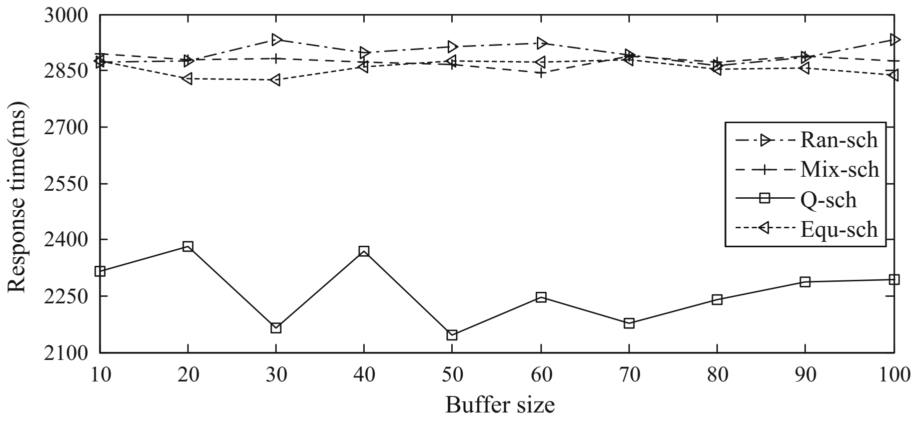


Fig. 5 Comparisons of response times with buffer size varying from 10 to 100 by 10 steps

strates that even though the number of VMs is small, the response time is insensitive to buffer size.

5.1.4 Comparison of response time when various arrival rate and server rate

From Fig. 6, we can obtain the relationship between response times with different arrival rates and server rates. The response time rises exponentially as arrival rate increases and serve rate decreases. With the aid of the experiments, since the user request arrival rate is observable and predictable in real cloud computing environment, the system parameters can be dynamically adaptive-adjusted in the following two aspects to improve the quality of the server. First, we can dynamically optimize the allocation of cloud computing resources based on the changes of user arrival rates to adjust the service rate adaptively, thus guaranteeing the response time and SLA; secondly, under the determined resource allocation, user requests are dynamically scheduled with optimizing efficiency, so to decrease the response time and facilitate service quality improvement.

5.2 Real cloud computing environment experiments results

We employ CloudSim [37] as our real cloud computing experiments environment, and compare its performance with that of other similar task scheduling algorithms based on two entirely different task submit methods as following:

(i) Type 1: User requests are arriving continuously. The scheduling manner under the cloud computing platform requires the dispatch is able to respond to tasks in time, which demands a higher ability of real-time response and lower time complexity. We called this episode as online task scheduling.
(ii) Type 2: User requests are arriving in batches. This manner requires the scheduler is able to optimize the tasks arrangement according to the task resources requirements, user's requests and the running state of system platform. Corresponding, we called this episode as offline task scheduling.

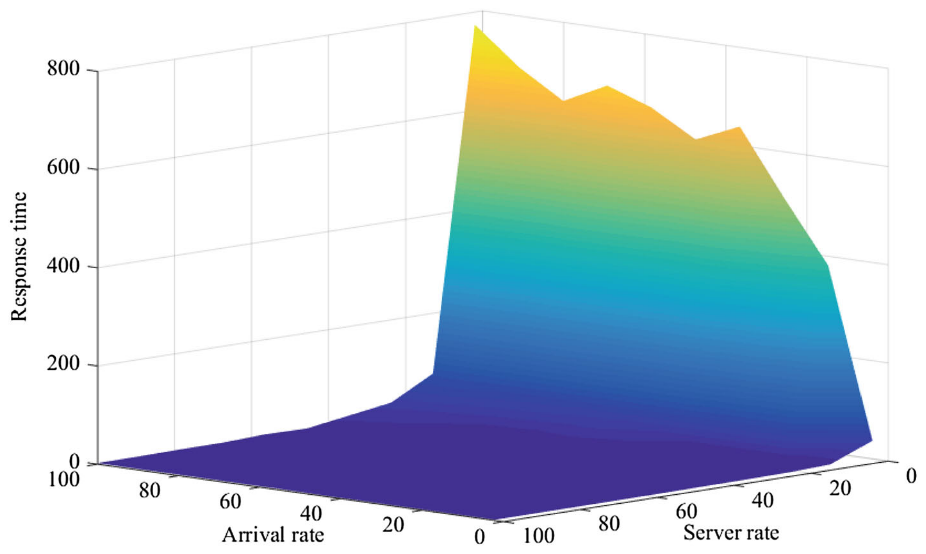


Fig. 6 Response time with different arrival rate and server rate settings

Table 1 Parameter settings type 1

Parameter	Values
Length of job	10,000–20,000 million instructions
Total number of jobs	100–1000
Total number of VMs	5
VM frequency	1,000–3,000 million instructions per second
VM memory (RAM)	512–2048 mega byte
VM bandwidth	500–1,000 mega byte per second
Number of VM buffer	10–50
Number of PEs requirements	5
Number of datacenters	1
Number of hosts	1

5.2.1 Online task scheduling

For the online task scheduling, user requests arrive at the task dispatch continuously, whose manner is similar to that of web requests or database querying. Under this situation, the real-time scheduling requirement is the most important performance index of online task scheduling scheme. In this experiment, we compared the proposed Q-sch with the famous real time task scheduling algorithm, such as FIFO [38], fair scheduling [39], greedy scheduling [40] and rand scheduling in terms of throughput and the average wait time. Table 1 shows the parameter settings for the type 1.

Figure 7 show the average response time of the various task scheduling schemes compared under the same cloud computing resources environment, from which we can generalize the conclusions as follows: (1) the average response time of every scheduling scheme increases with the number of tasks arrivals in one time slot; (2) the response time of

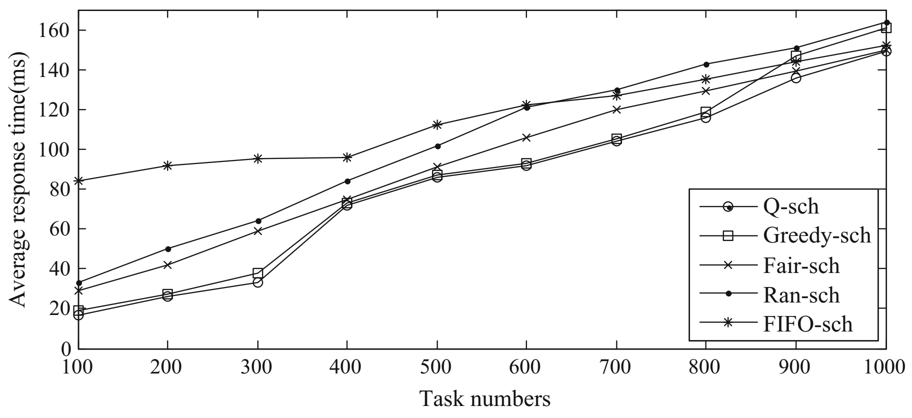


Fig. 7 Comparisons of average response times under various task numbers

Table 2 Parameter settings type 2

Parameter	Values
Length of job	10,000–100,000 Million Instructions
Total number of jobs	10–30
Total number of VMs	10
VM frequency	100–500 Million Instructions per second
VM memory(RAM)	512–2048 Mega Byte
VM bandwidth	500–1,000 Mega Byte per second
Number of VM buffer	10–50
Number of PEs requirements	1–2
Number of datacenters	2
Number of hosts	2

every scheduling scheme trickles up convergence, and (3) compared to the other scheduling schemes, the proposed scheduling scheme obtains the lowest performance index in average response time.

5.2.2 *Offline task scheduling*

For the offline task scheduling, user requests arrive at the task dispatch in batches, whose manner is similar to that of scientific calculation or business statistics. Under this situation, the real-time scheduling requirement is the most important performance index of offline task scheduling scheme. In this experiment, we compared the proposed Q-sch with the famous heuristic task scheduling algorithms, such as genetic algorithms (GA) and modified GA (MGA) [41] in terms of average makespans. Table 2 shows the parameter settings for the type 2.

Figure 8 show the average makespan of the various task scheduling schemes compared under same cloud computing resources environment, and the experiment results also demonstrate the superiority of the proposed task scheduling scheme.

6 Conclusions and future work

In this paper, we provide the insightful view about the task scheduling optimization problem for cloud computing platform. A queuing model, comprised of three concatenated submodels, namely TSSM, TESM and TTSM, is proposed to characterize the service process in cloud environment. In light of the proposed queuing model, we theoretically analyze the response time of each submodel and design a novel task scheduling scheme using reinforcement learning to formulate a cloud computation model and optimize the response time minimization issue under given cloud computing resources. We evaluate the proposed task scheduling optimization schemes in extensive simulations, Guided by the empirical observation, we find that the proposed task scheduling schemes can not only optimally utilize cloud resources and load balancing to obtain the minimal response time with resource constraint, but also reveal the relationship between response time and arrival rate, server rate, number of VMs, and number of buffer size, etc.

In the future, we plan to extend our schemes for dynamic task scheduling and resource allocation to get the minimal response time, providing highly satisfactory services and/or avoidance of SLA violations. Looking into the cloud entities and considering the details of cloud computing platform, such as VMs failures, VMs migrate, costs of communication, and burst arrivals of requests, VMs cluster for different kinds of requests will be other dimensions of extension.

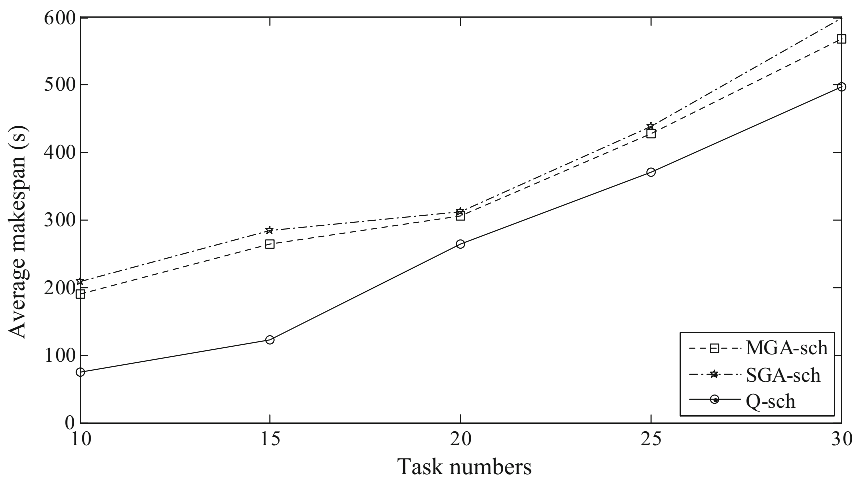


Fig. 8 Comparisons of average makespan under various task numbers

Acknowledgments The work presented in this paper was supported by National Natural Science Foundation of China (No. 61272382, 61402183).

References

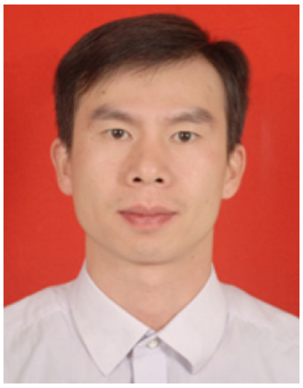
1. Vaquero, L., Rodero-Merino, L., Caceres, J., et al.: A break in the clouds: towards a cloud definition. ACM SIGCOMM Comput. Commun. Rev. **39**(1), 50–55 (2009)
2. Zhou, M., Zhang, R., Zeng, D., et al.: Services in the cloud computing era: a survey. In: Proceeding of the Fourth International Universal Communication Symposium (FIUCS2010), pp. 40–46 (2010)
3. Buyya, R., Yeo, C., Venugopal, S.: Market-oriented cloud computing: vision, hype, and reality for delivering it services as computing utilities. In: Proceeding of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC2008), pp. 5–13 (2008)
4. Delimitrou, C., Kozyrakis, C.: QoS-aware scheduling in heterogeneous datacenters with paragon. IEEE Micro **34**(3), 17–30 (2013)
5. Wang, W.J., Chang, Y.S., Lo, W.T., et al.: Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments. J. Supercomput. **66**(2), 783–811 (2013)
6. Kusic, D., Kephart, J., Hanson, J., et al.: Power and performance management of virtualized computing environments via lookahead control. Clust. Comput. **12**(1), 1–15 (2009)
7. Karve, A., Kimbrel, T., Pacifici, G., et al.: Dynamic placement for clustered Web applications. In: Proceeding of the 15th International Conference on World Wide web (WWW2006), pp. 593–604 (2006)
8. Heath, T., Diniz, B., Carrera, E.V., et al.: Energy conservation in heterogeneous server clusters. In: Proceeding of the 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP2005), pp. 186–195 (2005)
9. http://aws.amazon.com/ec2/
10. http://www.microsoft.com/windowsazure/
11. Yang, B., Tan, F., Dai, Y.S., et al.: Performance evaluation of cloud service considering fault recovery. In: Proceeding of the First International Conference on Cloud Computing (CloudCom 09), pp. 571–576 (2009)
12. Liu, X.D., Tong, W.Q., Zhi, X.L., et al.: Performance analysis of cloud computing services considering resources sharing among virtual machines. J. Supercomput. **69**, 357–374 (2014)
13. Khazaei, H., Misic, J., Misic, V.: Performance analysis of cloud computing centers using M/G/m/m+r.queuing systems. IEEE Trans. Parall. Distr. **23**(5), 936–943 (2012)
14. Khazaei, H., Misic, J., Misic, V.: Modeling of cloud computing centers using M/G/m queues. In: Proceeding of the 31st International Conference on Distributed Computing Systems Workshops (ICDCSW2011), pp. 87–92 (2011)
15. Khazaei, H., Misic, J., Misic, V.: Performance analysis of cloud centers under burst arrivals and total rejection policy. In: Proceeding of IEEE Global Telecommunications Conference (Globecom2011), pp. 1–6 (2011)
16. Rao, L., Liu, X., Xie, L., Liu, W.: Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment. In: Proceeding of IEEE INFOCOM, pp. 1–9 (2010)
17. Brenner, U.: A faster polynomial algorithm for the unbalanced Hitchcock transportation problem. Oper. Res. Lett. **36**, 408–413 (2008)
18. Luo, J.Y., Rao, L., Liu, X.: Eco-IDC: trade delay for energy cost with service delay guarantee for internet data centers. In: Proceeding of IEEE International Conference Cluster Computing (CLUSTER2012), pp. 45–53 (2012)
19. Luo, J.Y., Rao, L., Liu, X.: Temporal load balancing with service delay guarantees for data center energy cost optimization. IEEE Trans. Parall. Distr. **25**(3), 775–784 (2014)
20. Yao, Y., Huang, L., Sharma, A. et al.: Data centers power reduction: a two time scale Approach for delay tolerant workloads. In: Proceeding of IEEE INFOCOM, pp. 1431–1439 (2012)
21. Gao, Y.Q., Guan, H.B., Qi, Z.W., et al.: Service level agreement based energy-efficient resource management in cloud data centers. Comput. Electr. Eng. **40**, 1621–1633 (2014)
22. Suresh, V., Ezhilchelvan, P., Watson, P.: Scalable and responsive event processing in the cloud. Phil. Trans. R. Soc. A **371**(1983), 20120095 (2013)
23. Nan, X.M., He, Y.F., Guan, L.: Queueing model based resource optimization for multimedia cloud. J. Vis. Commun. Image Represent. **25**, 928–942 (2014)
24. Huynh, N., Tran, M., Nam, T.: Tool-driven strategies for resource provisioning of single-tier web applications in clouds. In: Proceeding of the 5th International Conference on Ubiquitous and Future Networks (ICUFN2013), pp. 795–799 (2013)
25. Tesauro, G., Jong, N., Das, R., et al.: A hybrid reinforcement learning approach to autonomic resource allocation. In: Proceedings of IEEE International Conference on Autonomic Computing (ICAC '06), pp. 65–73 (2006)
26. Xavier, D., Nicolas, R.: From data center resource allocation to control theory and back. In: Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD), pp. 410–417 (2010)
27. Julien, P., Cecile, G.R., Balazs, K., et al.: Multi-objective reinforcement learning for responsive grids. J. Grid Comput. **8**, 473–492 (2010)
28. Alexander, F., Matthias, H.: Improving scheduling performance using a Q-Learning-based leasing policy for clouds. In: Proceedings of 18th International Conference on Euro-par Parallel Processing (Euro-Par), pp. 337–349 (2012)
29. Bu, X.P., Rao, J., Xu, C.Z.: A reinforcement learning approach to online web systems auto-configuration. In: Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS2009), pp. 2–11 (2009)
30. Khazaei, H., Misic, J., Misic, V., et al.: A fine-grained performance model of cloud computing centers. IEEE Trans. Parall. Distr. **24**(11), 2138–2147 (2013)
31. Grimmett, G., Stirzaker, D.: Probability and random processes, 3rd edn. Oxford Univ Press, Oxford (2010)
32. Gross, D.: Fundamentals of queueing theory. Wiley-India Press, New Jersey (2008)
33. Sutton, R., Barto, A.: Reinforcement learning: an introduction. MIT Press, Cambridge (1998)
34. Wiering, M., Otterlo, M.: Reinforcement learning: state-of-the-art. Springer, New York (2012)
35. Tang, H., Pei, R., Zhou, L., et al.: Coordinate control of multiple CSPS system based on state aggregation method. Acta Autom. Sin. **40**(5), 901–908 (2014). (in Chinese)
36. http://www.mathworks.cn/
37. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource pro visioning algorithms. Softw. Pract. Exp. **41**(1), 23–50 (2011)
38. Isard, M., Budiu, M., Yu, Y., et al.: Distributed data-parallel programs from sequential building blocks. In: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems(EuroSys2007), pp. 59–72 (2007)
39. http://hadoop.apache.org/common/docs/current./capacity_sheduler.html[EB/OL]
40. Dong, Z.Q., Ning, L., Roberto, R.C., et al.: Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers. J. Cloud Comput. Adv. Syst. Appl. **4**, 1–14 (2015)

41. Kaur, S., Verma, A.: An efficient approach to genetic algorithm for job scheduling in cloud computing environment. Int. J. Info. Technol. Comput. Sci. **4**(10), 74–79 (2012)

Zhiping Peng received his B.S. and M.S. degrees from China University of Petroleum (Hua Dong) and Huazhong University of Science and Technology in 1996 and 2001, respectively, and received the Ph.D. degree in Computer Application from South China University of Technology in 2007. He is currently a professor at Guangdong University of Petrochemical Technology. He has published more than 50 papers in refereed journals and conference proceedings. His research interests include cloud computing, machine learning and multi-agent system.

Delong Cui received his M.S. degree in communication and information system in Southwest Jiao Tong University in 2008. He is currently an associate professor at Guangdong University of Petrochemical Technology. He has published more than 20 papers in refereed journals and conference proceedings. His research interests include cloud computing and reinforcement learning.

Jinglong Zuo received his M.S. degree in software engineering in Huazhong University of Science and Technology in 2007. He is currently an associate professor at Guangdong University of Petrochemical Technology. He has published more than 10 papers in refereed journals and conference proceedings. His research interests include cloud computing and reinforcement learning.



Qirui Li received his M.S. degree in Computer Science and Technology in South China University of Technology in 2008. He is studying in Guangzhou University for Ph.D. degree now. He is currently a lecturer at Guangdong University of Petrochemical Technology. He has published more than 10 papers in refereed journals and conference proceedings. His research interests include Artificial Intelligence, cloud computing, image processing and pattern recognition.

Bo Xu received his M.S. degree in Computer Science and Technology in Hunan University in 2009. He is a Ph.D. candidate at School of Software Engineering, South China University of Technology and also a teacher in Guangdong University of Petrochemical Technology. He is a Senior Member of China Computer Federation (CCF). His research interests include cloud computing, computational intelligence, multi-agent system.

Weiwei Lin received his BS and MS degrees from Nanchang University in 2001 and 2004, respectively, and the Ph.D. degree in Computer Application from South China University of Technology in 2007. He is currently working as an Associate Professor at the South China University of Technology. He has published more than 40 papers in refereed journals and conference proceedings. His research interests include distributed computing, big data and cloud computing.