A Task Scheduling Strategy in Cloud Computing with Service Differentiation

Yuanzheng Xue[1,2], Shunfu Jin[1,2], Xiushuang Wang[1,2]
[1] School of Information Science and Engineering
Yanshan University, Qinhuangdao 066004 - China
[2] Key Laboratory for Computer Virtual Technology
and System Integration of Hebei Province
Yanshan University, Qinhuangdao 066004 - China
[e-mail: jsf@ysu.edu.cn]
*Corresponding author: Shunfu Jin

Received April 8, 2017; revised June 27, 2018; accepted March 26, 2018; published November 30, 2018

Abstract

Task scheduling is one of the key issues in improving system performance and optimizing resource management in cloud computing environment. In order to provide appropriate services for heterogeneous users, we propose a novel task scheduling strategy with service differentiation, in which the delay sensitive tasks are assigned to the rapid cloud with high-speed processing, whereas the fault sensitive tasks are assigned to the reliable cloud with service restoration. Considering that a user can receive service from either local SaaS (Software as a Service) servers or public IaaS (Infrastructure as a Service) cloud, we establish a hybrid queueing network based system model. With the assumption of Poisson arriving process, we analyze the system model in steady state. Moreover, we derive the performance measures in terms of average response time of the delay sensitive tasks and utilization of VMs (Virtual Machines) in reliable cloud. We provide experimental results to validate the proposed strategy and the system model. Furthermore, we investigate the Nash equilibrium behavior and the social optimization behavior of the delay sensitive tasks. Finally, we carry out an improved intelligent searching algorithm to obtain the optimal arrival rate of total tasks and present a pricing policy for the delay sensitive tasks.

Keywords: cloud computing, service differentiation, queueing network, Jaya algorithm, pricing policy

This work was supported by National Natural Science Foundation (Nos. 61872311, 61472342) and Hebei Province Natural Science Foundation (No. F2017203141), China

http://doi.org/10.3837/tiis.2018.11.006 **ISSN : 1976-7277**

1. Introduction

Cloud computing as a new type of business computing service has been widely used by enterprises and individual users. Also cloud computing technology can deal with many services across the Internet [1]. According to NIST (National Institute of Standards and Technology), service modes in cloud computing are classified as SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service) [2-4]. IaaS providers supply storage space, computing and network resources with which users can execute operating systems, applications and any software. PaaS providers supply the software programming languages and system development tools to the users to deploy their own applications. SaaS providers supply applications to users through a client interface, such as a Web browser. An SaaS provider owns a small local data center, and can also acquire resources from public IaaS cloud [5-6].

Obviously, when all the tasks are processed by the local SaaS server, especially when there are a large number of tasks, the workload on the local SaaS server is too heavy. It is necessary to relieve the pressure on local SaaS server by shunting excess local task flow into public IaaS cloud. In [6], in order to minimize the operational cost of SaaS providers exploiting the hybrid cloud computing paradigm, Li et al. designed an online dynamic resource provision algorithm including long-term scheduling, short-term scheduling and dynamic provisioning. In [7], to minimize the cost of the overall infrastructure and maximize SaaS providers' profits, Liu et al. established a cloud service request model with SLA (Service-Level Agreement) constraints, and then presented a cost-aware service request scheduling approach based on genetic algorithm. In the literatures mentioned above, all the tasks are regarded as homogeneous.

With the diversification of users requirements for network Quality of Service (QoS), service differentiation has been studied for decades under different disciplines, such as processor sharing, packet switching, storage systems and Web servers [8]. A key challenge for service differentiation is how to satisfy different users. In [9], for the purpose of minimizing resource cost under response time constraints of multimedia services, Nan et al. proposed a queueing model to describe service differentiation, and optimized cloud resources in the first-come first-served (FCFS) scenario and the priority scenario, respectively. In [10], in order to share CPU service in clusters of servers, Katsalis et al. proposed Dynamic Weighted Round Robin (DWRR) algorithms, and used stochastic control theory to demonstrate that the objective of service differentiation is achieved by DWRR. In [11], a new auction-based resource allocation framework for cloud systems called Abacus was presented by Ding et al.. By exploiting Abacus, the service differentiation for tasks is effective with different budgets, utility properties and priorities. In these works, all the services are differentiated as abstract categories.

Inspired by this observation, in this paper, we propose a method to shunt the tasks. When a user requests for service from the SaaS provider, the request is handled by either local data center or public IaaS cloud. The public IaaS cloud consists of lots of VMs. Furthermore, we differentiate tasks as delay sensitive tasks and fault sensitive tasks. Accordingly, we divide public IaaS cloud into the rapid cloud and the reliable cloud to satisfy the requirements from heterogeneous users. Based on above, we propose a novel task scheduling strategy with service differentiation and establish a hybrid queueing network based system model. Moreover, we analyze the average response time of the delay sensitive tasks, and calculate the utilization of VMs in the reliable cloud. The reasonability and efficiency of the task scheduling

strategy are verified by statistical experiments with analysis and simulation. Furthermore, we analyze the Nash equilibrium behavior and the socially optimal behavior of the delay sensitive tasks, and improve an intelligent searching algorithm based on Jaya algorithm to obtain the socially optimal arrival rate of total tasks effectively. Finally, we present the pricing policy for the delay sensitive tasks.

The contributions of our paper are two-fold. One, a novel task scheduling strategy with service differentiation is proposed. With this proposed strategy, the traffic load on the local SaaS server can be relieved and diverse requirements from different users can be satisfied. Other one, a hybrid queueing network based system model is established to analyze the proposed strategy, and a Jaya algorithm is improved to obtain the socially optimal arrival rate.

The remainder of this paper is organized as follows. In Section 2, we propose a task scheduling strategy in cloud computing with service differentiation and establish a hybrid queueing network based system model. In Section 3, we analyze the system model in the steady state. With statistical experiments, we estimate the system performance of the proposed task scheduling strategy in Section 4. In Section 5, we improve an intelligent searching algorithm to obtain the socially optimal arrival rate of total tasks, and impose an admission fee for the delay sensitive tasks. Finally, Section 6 concludes the whole paper.

2. Task Scheduling Strategy and System Model

In this section, we firstly propose a task scheduling strategy with service differentiation for SaaS provider. Then, we establish a hybrid queueing network based system model accordingly.

2.1. Task Scheduling Strategy

With the development of cloud computing, more and more enterprises and individuals tend to use cloud service. The users will send out a large number of service requests to the SaaS provider. When all the tasks crowd into the local data center, the workload of the local data center is too heavy to be handled with. So the local server needs additional infrastructure resources to shunt the excess task flow. SaaS providers use a combination of their local servers and IaaS cloud infrastructures to carry out the tasks in order to speed up the processing and improve the service capacity. In conventional pattern of SaaS provider, all the tasks are considered as homogeneous and processed indiscriminately. However, in practice, some users need service with less delay, whereas some users need service with higher reliability. That is to say, the users are heterogeneous. From view of this scenario, we propose a task scheduling strategy with service differentiation.

When a user requests service from SaaS provider, SaaS provider will schedule this task appropriately to the local server or the public IaaS VMs. When the workload is lighter, the local server will process the tasks excellently without service differentiation. Hence, fewer tasks will be assigned to the IaaS cloud. However, when all the workload is only processed by the local SaaS server, with the increase in the workload, the latency of tasks will increase accordingly, in order to enhance the experience of users, more tasks should be shunted to the IaaS cloud. In this way, the average response time of tasks and the utilization of resources should be balanced. According to the service differentiation, we divide public IaaS cloud into two groups: the rapid cloud and the reliable cloud. There is a load balancer to coordinate heterogeneous requirements. The delay sensitive tasks will be assigned to the rapid cloud with high-speed processing, whereas the fault sensitive tasks will be assigned to the reliable cloud with service restoration.

The working principle of the proposed task scheduling strategy is shown in **Fig. 1**.

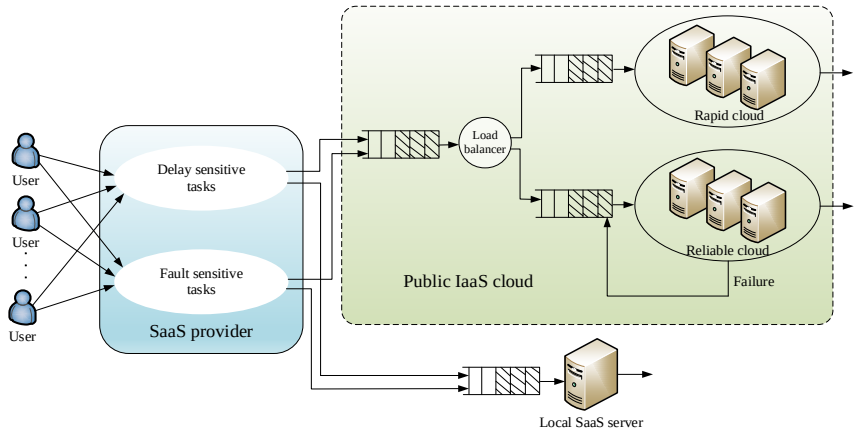

Fig. 1. The working principle of the task scheduling strategy

(1) We consider one local SaaS server with a perfect processing result. According to the number of total tasks including delay sensitive tasks and fault sensitive tasks, some tasks will be assigned to the local SaaS server, other tasks will be assigned to the IaaS cloud.

(2) The load balancer in IaaS cloud is a VM in practice. According to the ToS (Type of Service) of the IP (Internet Protocol) data package, the load balancer will assign the heterogeneous tasks to either of the two groups of IaaS VMs. The delay sensitive tasks are assigned to the rapid cloud, whereas the fault sensitive tasks are assigned to the reliable cloud.

(3) The VMs in the rapid cloud are configured with powerful hardware to support the fast processing speed. To satisfy the requirements from users with the delay sensitive tasks, the VMs in rapid cloud provide cloud service in a best-effort mode.

(4) The VMs in reliable cloud have the ability of service restoration. The tasks may be processed unsuccessfully due to the noisy channel. To detect a fault sensitive task, such as file transmission, is processed successfully or not, a checksum is inserted to data frame in the link layer. If the checksum is correct, the copy of the data frame will be silently discarded. Otherwise, the copy of this data frame will be taken out of the back-up [12] and sent out again, i.e., the transmission of this data frame will be restored by the IaaS VM.

2.2. Model Building

In the proposed task scheduling strategy with service differentiation, the input stream of the load balancer in IaaS cloud is part of the tasks from users' requests, the departure stream of the load balancer is the sum of the arrival stream of the rapid cloud and that of the reliable cloud. Hence, a hybrid queueing network based system model is established.

The tasks are supposed to arrive at the SaaS provider as a Poisson process with parameter $\lambda\,(\lambda>0)$. The process times of a task in the local server, the load balancer, a VM in the rapid cloud with high-speed processing and a VM in the reliable cloud with service restoration are supposed to follow exponential distribution with parameters $\mu_0\,(\mu_0>0)$, $\mu_1\,(\mu_1>0)$, $\mu_2\,(\mu_2>0)$ and $\mu_3\,(\mu_3>0)$, respectively. Obviously, when the arrivals at the SaaS provider follow a Poisson process with parameter λ, the arrivals at the load balancer will also follow a Poisson process. We assume that a task is assigned to the IaaS cloud with probability

δ $(0 \le \delta \le 1)$ (δ is called as shunting probability). Therefore, the tasks arrive at the load balancer in IaaS cloud with rate $\delta\lambda$, the tasks arrive at the local SaaS server with rate $(1-\delta)\lambda$. All the buffers are supposed to be infinite.

For the local SaaS server, we establish an M/M/1 queue. For the load balancer, we establish an M/M/1 queue. According to the Burke's theorem [13], we know that the output of the M/M/1 queue with arrival rate $\delta\lambda$ is also a Poisson process with the same parameter. The output stream of the load balancer is split between the rapid cloud and the reliable cloud. We denote the percentage that the fault sensitive tasks over total tasks as parameter p $(0 \le p \le 1)$. Hence, the arrival rate of the fault sensitive tasks at the reliable cloud is $p\delta\lambda$, and the arrival rate of the dalay sensitive tasks at the rapid cloud is $(1-p)\delta\lambda$. For the rapid cloud, we establish an M/M/*m* queue (*m* is the number of VMs in the rapid cloud), and for the reliable cloud, we establish a restoration driven M/M/*n* queue (*n* is the number of VMs in the reliable cloud).

3. Model Analysis

In this section, we analyze the hybrid queueing network based system model in steady state.

3.1. Steady-State Condition

The hybrid queueing network based system model is composed of four queues, including an M/M/1 queue with traffic load ρ_0 established from the view of the local SaaS server, an M/M/1 queue with traffic load ρ_1 established from the view of the load balancer, an M/M/*m* queue with traffic load ρ_2 established from the view of the rapid cloud, and a restoration driven M/M/*n* queue with traffic load ρ_3 established from the view of the reliable cloud.

By taking advantage of the existing results [14], the traffic loads ρ_0, ρ_1 and ρ_2 are given as follows:

$$\rho_0 = \frac{(1-\delta)\lambda}{\mu_0}, \quad \rho_1 = \frac{\delta\lambda}{\mu_1}, \quad \rho_2 = \frac{(1-p)\delta\lambda}{m\mu_2}. \tag{1}$$

The traffic load ρ_3 will be discussed in the following subsection.

The necessary and sufficient condition for the hybrid queueing network based system model to be stable is that all the above traffic loads are less than 1.

3.2. Steady-State Distribution

We note that the queues established from the views of the local SaaS server, the load balancer and the rapid cloud can be obtained from existing literatures. In this subsection, we focus on analysis of the restoration driven M/M/*n* queue.

As pointed in Section 2, the processing time B of a task on a VM in the reliable cloud follows exponential distribution with parameter μ_3. Let θ $(0 \le \theta \le 1, \overline{\theta} = 1-\theta)$ be the probability that a fault sensitive task is processed unsuccessfully at a time, θ is called as error rate. We note that the tasks that processed unsuccessfully are still in the system. Let β be the probability that a fault sensitive task processed successfully and leave the system during the time interval $(t, t+\Delta t)$. β can be presented as follows:

$$\beta = \overline{\theta} P\{B \le t + \Delta t \mid B > t\} = \overline{\theta}\mu_3 \Delta t + o(\Delta t). \quad (2)$$

Let $N(t)$ be the arrival process of the restoration driven M/M/n queue. $\{N(t), t \ge 0\}$ is a Poisson process with parameter $p\delta\lambda$, where p is the percentage that the fault sensitive tasks over total tasks in IaaS cloud, δ is the probability that a task is assigned to the IaaS cloud. Let $X(t) = i$, $i \in \{0,1,...\}$ be the total number of fault sensitive tasks (including the tasks that are processing) in the restoration driven M/M/n queue at the time instant t. $\{X(t), t \ge 0\}$ is a Markov process, and we describe $\{X(t), t \ge 0\}$ as system state. Let $F(\Delta t) = k$, $k \in \{0,1,..., \min(i,n)\}$ be the number of tasks processed successfully during the time interval $(t, t + \Delta t)$. $F(\Delta t)$ is a binomial distribution with parameters $\min(i,n)$ and β.

We prove that the Markov process $\{X(t), t \ge 0\}$ for the restoration driven M/M/n queue is a birth-and-death process as follows:

(1) $P_{i,i+1}(\Delta t)$ denotes the probability that there are i fault sensitive tasks at the time instant t, and $(i+1)$ fault sensitive tasks at the time instant $(t + \Delta t)$. That is, the number of tasks in the system increases by one during the time interval $(t, t + \Delta t)$. $P_{i,i+1}(\Delta t)$ can be given as follows:

$$\begin{aligned} P_{i,i+1}(\Delta t) &= P\{X(t+\Delta t) = i+1 \mid X(t) = i\} \\ &= \sum_{k=0}^{\min(i,n)} P\{F(\Delta t) = k, N(t+\Delta t) - N(t) = k+1 \mid X(t) = i\} \\ &= p\delta\lambda\Delta t + o(\Delta t), \ i \ge 0. \end{aligned} \quad (3)$$

(2) $P_{i,i-1}(\Delta t)$ denotes the probability that there are i fault sensitive tasks at the time instant t, and $(i-1)$ fault sensitive tasks at the time instant $(t + \Delta t)$. That is, the number of tasks in the system decreases by one during the time interval $(t, t + \Delta t)$. $P_{i,i-1}(\Delta t)$ can be given as follows:

$$\begin{aligned} P_{i,i-1}(\Delta t) &= P\{X(t+\Delta t) = i-1 \mid X(t) = i\} \\ &= \sum_{k=1}^{\min(i,n)} P\{F(\Delta t) = k, N(t+\Delta t) - N(t) = k-1 \mid X(t) = i\} \\ &= \begin{cases} i\overline{\theta}\mu_3\Delta t + o(\Delta t), & i = 1,2,...,n-1, \\ n\overline{\theta}\mu_3\Delta t + o(\Delta t), & i = n, n+1,.... \end{cases} \end{aligned} \quad (4)$$

(3) $P_{i,j}(\Delta t)$ denotes the probability that there are i fault sensitive tasks at the time instant t, j fault sensitive tasks at the time instant $(t + \Delta t)$, and $|j - i| \ge 2$. That is, there are more than one arrivals or departures in the system during the time interval $(t, t + \Delta t)$. $P_{i,j}(\Delta t)$ can be given as follows:

$$P_{i,j}(\Delta t) = P\{X(t+\Delta t) = j \mid X(t) = i\} = o(\Delta t), \ |j - i| \ge 2. \quad (5)$$

(4) $P_{i,i}(\Delta t)$ denotes the probability that there are i fault sensitive tasks at the time instant t, and i fault sensitive tasks at the time instant $(t + \Delta t)$ too. That is, the number of tasks in the system is fixed during the time interval $(t, t + \Delta t)$. $P_{i,i}(\Delta t)$ can be given as follows:

$$\begin{aligned} P_{i,i}(\Delta t) &= P\{X(t+\Delta t) = i \mid X(t) = i\} \\ &= 1 - p\delta\lambda\Delta t - \mu_3^i\Delta t + o(\Delta t). \end{aligned} \quad (6)$$

Based on Eqs. (3)-(6), we find that $\{X(t), t \geq 0\}$ is a birth-and-death process. The birth rate is $p\delta\lambda$, the death rate is given as follows:

$$\mu_3^i = \begin{cases} i\bar{\theta}\mu_3, & i = 1,2,...,n, \\ n\bar{\theta}\mu_3, & i = n+1, n+2,.... \end{cases} \tag{7}$$

The steady-state condition of the restoration driven M/M/*n* queue is given as follows:

$$\rho_3 = \frac{p\delta\lambda}{n\bar{\theta}\mu_3} < 1. \tag{8}$$

Let π_i be the steady-state distribution of $X(t)$. π_i is the probability that there are *i* tasks in the restoration driven M/M/*n* queue in steady state. π_i can be given as follows:

$$\pi_i = \lim_{t\to\infty} P\{X(t) = i\}, \ i = 0,1,.... \tag{9}$$

In context of the steady-state condition, we can build a set of equilibrium equations as follows:

$$\begin{cases} p\delta\lambda\pi_0 = \bar{\theta}\mu_3\pi_1, \\ (p\delta\lambda + i\mu_3)\pi_i = i\theta\mu_3\pi_i + p\delta\lambda\pi_{i-1} + (i+1)\bar{\theta}\mu_3\pi_{i+1}, \ i = 1,2,...,n, \\ (p\delta\lambda + n\mu_3)\pi_i = n\theta\mu_3\pi_i + p\delta\lambda\pi_{i-1} + n\bar{\theta}\mu_3\pi_{i+1}, \ i = n+1, n+2,.... \end{cases} \tag{10}$$

By solving Eq. (10) and using the normalization condition $\sum_{i=0}^{\infty}\pi_i = 1$, the steady-state distribution π_i of the restoration driven M/M/*n* queue can be given as follows:

$$\pi_i = \begin{cases} \dfrac{(n\rho_3)^i\pi_0}{i!}, & i = 1,2,...,n, \\ \dfrac{n^n\rho_3^i\pi_0}{n!}, & i = n+1, n+2,... \end{cases} \tag{11}$$

where $\pi_0 = \left(\sum_{i=0}^{n-1} \dfrac{(n\rho_3)^i}{i!} + \dfrac{(n\rho_3)^n}{n!(1-\rho_3)} \right)^{-1}$.

4. Performance Measures and Statistical Experiments

In this section, we derive some performance measures and provide statistical experiments to evaluate the task scheduling strategy proposed in this paper.

4.1. Performance Measures

In our proposed strategy, we focus on the average response time of the delay sensitive tasks and the utilization of VMs in the reliable cloud in which the fault sensitive tasks are processed.

We define the response time of a task as the time duration from a task arrives at the SaaS provider to the task is processed completely.

Let $E[T_0]$ be the average sojourn time of a task processed by the local server, and $E[T_1]$ be the average sojourn time of a task processed by the load balancer in IaaS cloud. Referencing to [14], $E[T_0]$ and $E[T_1]$ are given as follows:

$$E[T_0]=\frac{1}{\mu_0(1-\rho_0)},\qquad E[T_1]=\frac{1}{\mu_1(1-\rho_1)}. \tag{12}$$

Let $E[T_2]$ be the average sojourn time of a delay sensitive task processed by the rapid cloud. Referencing to [14], $E[T_2]$ is given as follows:

$$E[T_2]=\frac{\rho_2}{(1-p)\delta\lambda(1-\rho_2)^2}\times\frac{(m\rho_2)^m p_0}{m!}+\frac{1}{\mu_2} \tag{13}$$

where $p_0=\left(\sum_{i=0}^{m-1}\frac{(m\rho_2)^i}{i!}+\frac{(m\rho_2)^m}{m!(1-\rho_2)}\right)^{-1}$.

The average response time $E[T_{ds}]$ of the delay sensitive tasks can be given as follows:

$$E[T_{ds}]=(1-\delta)E[T_0]+\delta\left(E[T_1]+E[T_2]\right). \tag{14}$$

We define the utilization U_{rec} of VMs in the reliable cloud as the probability that the VMs are occupied by the fault sensitive tasks. U_{rec} can be calculated as the percentage that the average number of occupied VMs over the number of VMs in the reliable cloud. If the number of tasks in the reliable cloud is less than the number of VMs, the number of occupied VMs is the number of tasks. Otherwise, the number of occupied VMs is the number of VMs. U_{rec} can be given as follows:

$$U_{rec}=\sum_{i=1}^{n}\frac{i\pi_i}{n}+\sum_{i=n+1}^{\infty}\pi_i \tag{15}$$

where π_i is given in Eq. (11).

4.2. Statistical Experiments

In order to validate the effectiveness of our proposed task scheduling strategy, we provide statistical experiments. Based on Eqs. (14) and (15) given in Subsection 4.1, we obtain the analysis results using MATLAB. To verify our analytic model, we simulate the proposed task scheduling strategy. The simulation results are obtained using MyEclipse2014. In order to ensure the accuracy of the simulation experiments as far as possible, we make use of the Monte Carlo Method [15], and average over 10 independent runs. From the statistical experiments shown in **Figs. 2** and **3**, we see that the analysis results match well with the simulation results.

In order to make sense for parameter settings and satisfy steady-state condition, we set the service rate of the local SaaS server as $\mu_0=1.0$, the service rate of the load balancer in IaaS cloud as $\mu_1=0.8$, the service rate of a VM in the rapid cloud with high-speed processing as $\mu_2=0.2$ and the service rate of a VM in the reliable cloud with service restoration as $\mu_3=0.15$. Unless otherwise specified, the percentage that the fault sensitive tasks over total tasks is $p=0.4$ in experiments.

In conventional task scheduling strategy, all the tasks in SaaS provider are processed without service differentiation. By setting the percentage that the fault sensitive tasks over total tasks as $p=0$, all the tasks are considered as the delay sensitive tasks, we can obtain the average response time of the delay sensitive tasks with conventional strategy. By setting the percentage that the fault sensitive tasks over total tasks as $p=1$, all the tasks are considered as

the fault sensitive tasks, we obtain the utilization of VMs in the reliable cloud with conventional strategy. In Figures 2 and 3, the solid line represents the analysis results with the proposed task scheduling strategy, the dotted line represents the analysis results with the conventional task scheduling strategy.

Taking the number $m = 3$ of VMs in the rapid cloud as an example, we demonstrate the average response time $E[T_{ds}]$ of the delay sensitive tasks versus the arrival rate λ of total tasks in **Fig. 2**.

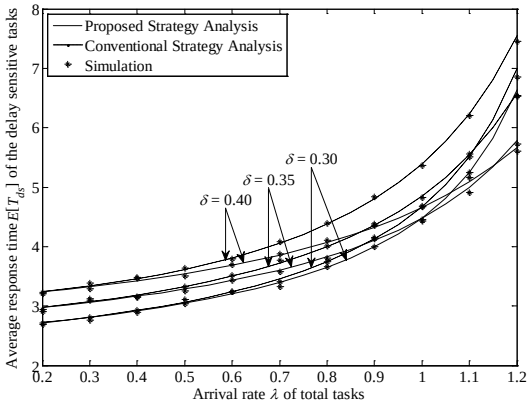

Fig. 2. Average response time $E[T_{ds}]$ of the delay sensitive tasks

In **Fig. 2**, we find that, for the same shunting probability δ, the average response time $E[T_{ds}]$ of the delay sensitive tasks will increase as the arrival rate λ of total tasks increases. The reason is that as the total arrival rate increases, the arrival rate of the delay sensitive tasks will increase accordingly, this will result in an increase in the number of the delay sensitive tasks waiting in the buffer. Therefore, the average response time will get greater.

We also observe that there is a reverse for the curves of average response time $E[T_{ds}]$ of the delay sensitive tasks as the arrival rate λ of total tasks increases. For a smaller arrival rate of total tasks, the average response time of the delay sensitive tasks will decrease as the shunting probability decreases. We note that the smaller the shunting probability is, the more the tasks will be processed by the local SaaS server. That is to say, when there are a small number of tasks arriving, if most of the tasks, no matter the delay sensitive tasks or the fault sensitive tasks, are processed on the local SaaS server, the average response time of the tasks will decrease. The reason is that the service resource of the local SaaS server is sufficient for a small number of tasks, and the local SaaS server has a higher processing speed. On the other hand, for a larger arrival rate of total tasks, the average response time of the delay sensitive tasks will decrease as the shunting probability increases. A larger shunting probability means that more tasks will be processed by the IaaS cloud. The processing capacity of the local SaaS server is insufficient for plenty of tasks, so more tasks, including the delay sensitive tasks and the fault sensitive tasks, will be assigned to the rapid cloud or the reliable cloud, respectively. We note that there are many VMs in the IaaS cloud, the service resource of the IaaS cloud is sufficient for a large number of tasks. So the average response time of the delay sensitive tasks will decrease as the shunting probability increases. In other words, for a smaller arrival rate of

total tasks, a smaller shunting probability should be set to decrease the average response time. For a larger arrival rate of total tasks, a larger shunting probability should be set to decrease the average response time. The average response time of the delay sensitive tasks reaches minimum with a certain shunting probability when the arrival rate of total tasks is fixed.

Compared the solid lines and the dotted lines with the same shunting probability δ, we find that the average response time of the delay sensitive tasks with the proposed strategy is lower than that with the conventional strategy. That is to say, the proposed task scheduling strategy performs better than the conventional task scheduling strategy on the response performance of the delay sensitive tasks.

Taking the number $n=3$ of VMs in the reliable cloud and the shunting probability $\delta=0.30$ as an example, we demonstrate the utilization U_{rec} of VMs in the reliable cloud versus arrivla rate λ of total tasks.

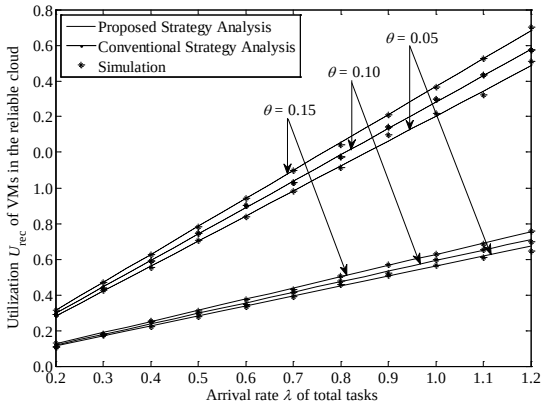

Fig. 3. Utilization U_{rec} of VMs in the reliable cloud

As seen in **Fig. 3**, we find that, for the same error rate θ, the utilization U_{rec} of VMs in the reliable cloud will increase as the arrivla rate λ of total tasks increases. The reason is that when arrivla rate λ of total tasks gets larger, more VMs in the reliable cloud are activated to process these tasks, so the utilization U_{rec} becomes greater.

We also observe that for the same arrivla rate λ of total tasks, the utilization U_{rec} of VMs in the reliable cloud will increase as the error rate θ increases. A higher error rate means that more fault sensitive tasks need to be reprocessed, so the tasks will occupy the VMs in the reliable cloud for a longer time. Therefore, the utilization of VMs in the reliable cloud will increase. The larger the error rate is, the greater the slope of the utilization of VMs in the reliable cloud will be. In other words, a higher error rate will make the VMs in the reliable cloud much busier.

Compared the solid lines and the dotted lines with the same error rate θ, we find that the utilization of VMs in the reliable cloud with proposed strategy is lower than that with conventional strategy. We note that a lower utilization always means a better response performance.

5. Performance Optimization and Pricing Policy

In this section, we firstly investigate the Nash equilibrium behavior and the socially optimal behavior of the delay sensitive tasks in the proposed task scheduling strategy. Then, in order to optimize the system performance with the maximum social profit, we propose a pricing policy for the delay sensitive tasks.

5.1. Performance Optimization

With the task scheduling strategy proposed in Section 2, we note that all the delay sensitive tasks make decisions independently to access the system in order to maximize their benefits. In other words, the Nash equilibrium behavior of a delay sensitive task is acted from its own viewpoint. Each delay sensitive task desires to access the system and to acquire service quickly. However, a higher arrival rate of total tasks means a higher arrival rate of the delay sensitive tasks, which will make the average response time of the delay sensitive tasks higher, then the individual benefit of each task will decrease. That is to say, the selfish behavior of the delay sensitive tasks will lead to a negative benefit for the system. The decision should be made at a social level. Therefore, it is necessary to coincide the Nash equilibrium behavior and the socially optimal behavior of the delay sensitive tasks.

In order to investigate the Nash equilibrium behavior and the socially optimal behavior of the delay sensitive tasks, we present several assumptions as follows:

(1) Both of the number of the delay sensitive tasks queueing at the system buffer and the number of tasks being processed are unobservable for a newly arriving delay sensitive task.

(2) The reward for a delay sensitive task processed completely is R.

(3) The cost of a delay sensitive task staying in the system is C per unit time.

(4) The benefits for all the delay sensitive tasks can be added together.

(5) The service discipline of the delay sensitive tasks is First Come First Served (FCFS).

We note that when the percentage $(1-p)$ that the delay sensitive tasks over total tasks is given, the arrival rate of the delay sensitive tasks is determined by the arrival rate of total tasks. Therefore, we study the Nash equilibrium behavior and the socially optimal behavior of the delay sensitive tasks versus the arrival rate of total tasks.

The individual benefit function $G_{ind}(\lambda)$ of a delay sensitive task can be given as follows:

$$G_{ind}(\lambda) = R - CE[T_{ds}] \tag{16}$$

where $E[T_{ds}]$ is the average response time of the delay sensitive tasks given in Eq. (14). In practice, when the users are strict with real-time, the reward R should be set lower and the cost C be set higher. On the other hand, when the user requirement for real-time is lower, the reward R should be set higher and the cost C be set lower.

The social profit function $G_{soc}(\lambda)$ is the aggregation of all the individual benefits of the arriving delay sensitive tasks. As shown in Subsection 2.2, the arrival rate of the delay sensitive tasks at the system is $(1-p)\lambda$. Then $G_{soc}(\lambda)$ can be given as follows:

$$G_{soc}(\lambda) = (1-p)\lambda(R - CE[T_{ds}]). \tag{17}$$

Applying the parameters used in Subsection 4.2, and setting the reward $R = 2.5$, the cost $C = 0.6$ as an example, we provide numerical experiments to explore the individual benefit function $G_{ind}(\lambda)$ and the social profit function $G_{soc}(\lambda)$.

Fig. 4 demonstrates the individual benefit function $G_{ind}(\lambda)$ of a delay sensitive task versus the arrival rate λ of total tasks.

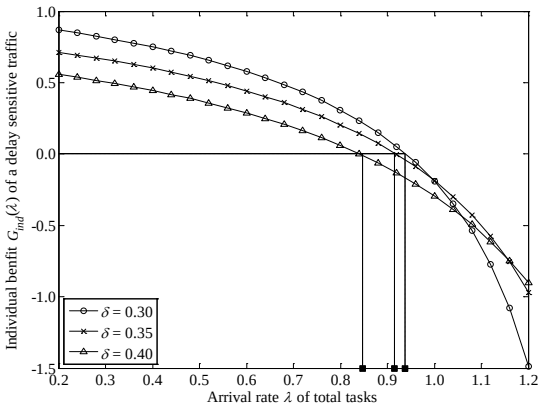

Fig. 4. Individual benefit $G_{ind}(\lambda)$ of a delay sensitive task

In **Fig. 4**, we observe that for the same shunting probability δ, the individual benefit $G_{ind}(\lambda)$ of a delay sensitive task presents a downtrend with the increase in the arrival rate λ of total tasks. The reason is that as the arrival rate of total tasks increases, the arrival rate of the delay sensitive tasks will increase certainly, which will inevitably lead to a higher average response time of the delay sensitive tasks, then the cost of the delay sensitive tasks will increase. Therefore, the individual benefit will decrease accordingly. We also observe that for each curve in **Fig. 4**, there is an unique value of λ subject to $G_{ind}(\lambda)=0$, and the unique value is called as the Nash equilibrium arrival rate λ^e of total tasks. The Nash equilibrium arrival rates λ^e of total tasks are marked with "■" in **Fig. 4**.

Fig. 5 demonstrates the social profit function $G_{soc}(\lambda)$ of the delay sensitive tasks versus the arrival rate λ of total tasks.

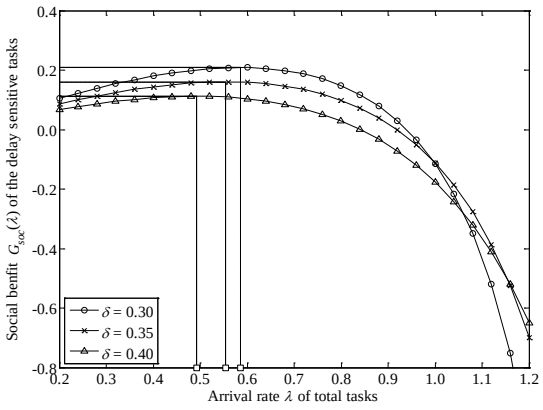

Fig. 5. Social profit $G_{soc}(\lambda)$ of the delay sensitive tasks

In **Fig. 5**, we observe that the change trend of the social profit $G_{soc}(\lambda)$ of the delay sensitive tasks experiences two stages. During the first stage, the social profit $G_{soc}(\lambda)$ presents an uptrend as the arrival rate λ of total tasks increases. We note that when the arrival

rate of total tasks is smaller, the average response time of the delay sensitive tasks is smaller too, then the cost of the delay sensitive tasks is smaller. For this case, the reward of the delay sensitive tasks is the dominant factor to influence the social profit. The larger the arrival rate of total tasks is, the more the reward of the delay sensitive tasks will earn, which means the social profit will increase. During the second stage, the social profit $G_{soc}(\lambda)$ presents a downtrend as the arrival rate λ of total tasks increases. We also note that as the arrival rate of total tasks increases, the average response time of the delay sensitive tasks increases immensely, the cost of the delay sensitive tasks will be greater accordingly. For this case, the increase in the average response time of the delay sensitive tasks causes more arriving delay sensitive tasks to wait in the buffer, and the arriving delay sensitive tasks can not be processed quickly. So the cost of the delay sensitive tasks becomes the dominant factor to influence the social profit. A bigger arrival rate of total tasks will lead to a greater cost of the delay sensitive tasks, then the social profit will decrease and tend to a negative value finally. Therefore, all the curves exhibit a property of being concave, which is coincident with what we observed from **Fig. 5**. The maximum social profit is the peak value for each curve, the corresponding arrival rate of total tasks is called as the socially optimal arrival rate λ^* of total tasks, i.e., $\lambda^* = \arg\max\{G_{soc}(\lambda)\}$. The socially optimal arrival rates λ^* of total tasks are marked with "□" in **Fig. 5**.

We note that the social profit $G_{soc}(\lambda)$ given in Eq. (17) is nonlinear, the strict monotonicity of the social profit is difficult to be addressed. So the socially optimal arrival rate of total tasks and the maximum social profit are difficult to be obtained by traditional optimization methods. For this reason, we turn to intelligent optimization algorithm to solve the socially optimal arrival rate λ^* of total tasks and give the maximum social profit $G_{soc}(\lambda^*)$ of the delay sensitive tasks.

Jaya algorithm [16] is a simple yet powerful optimization algorithm to search the best solution, and it does not need additional algorithm-specific parameters. The algorithm is based on the concept that the solution for a given problem should move towards the best solution and should avoid the worst solution. We note that the initial population is an important influence factor on the searching capacity of the optimization algorithm. We improve the Jaya algorithm by adopting a chaotic map named sine map [17] to generate more diverse initial population.

The main steps of the improved Jaya algorithm are presented in Algorithm 1.

Algorithm 1: Improved Jaya algorithm

Step 1: Initialize the maximum number $iter_{max}$ of iterations, the parameter a (for example, $a = 4$) of the sine map, the population size N, upper bound λ_{up} for the arrival rate of total tasks, lower bound λ_{low} for the arrival rate of total tasks. Set the initial number of iterations as $iter = 0$.

Step 2: Initialize the arrival rate λ_i, $i \in \{1,2,...,N\}$ of total tasks in population using sine map:

$$\lambda_{i+1} = \frac{a}{4}\sin(\pi\lambda_i),\ \lambda_1 = rand.$$

% $rand$ is the uniform random variable selected in the interval $(0,1)$.

Step 3: Make sure each initial arrival rate λ_i, $i \in \{1,2,...,N\}$ of total tasks is under constraint condition $\lambda_i \in [\lambda_{low}, \lambda_{up}]$:

$$\lambda_i = \frac{\lambda_i - \min\limits_{j\in\{1,\ldots,N\}}\{\lambda_j\}}{\max\limits_{j\in\{1,\ldots,N\}}\{\lambda_j\} - \min\limits_{j\in\{1,\ldots,N\}}\{\lambda_j\}} \times \left(\lambda_{up} - \lambda_{low}\right).$$

Step 4: Calculate the best arrival rate λ_{best} and the worst arrival rate λ_{worst} of total tasks:

$$G_{soc}(\lambda_i) = (1-p)\lambda_i(R - CE[T_{ds}]_i),\ i \in \{1,2,\ldots,N\},$$

$$\lambda_{best} = \underset{i\in\{1,\ldots,N\}}{\operatorname{argmax}}\{G_{soc}(\lambda_i)\},\ \lambda_{worst} = \underset{i\in\{1,\ldots,N\}}{\operatorname{argmin}}\{G_{soc}(\lambda_i)\}.$$

% $E[T_{ds}]_i$ is the average response time of the delay sensitive tasks with λ_i .

Step 5: Update the arrival rate λ_i, $i \in \{1,2,\ldots,N\}$ of total tasks in population:

$$\lambda_i^u = \lambda_i \times \left(1 + \exp\left(-\left(\frac{G_{soc}(\lambda_i)}{\max\limits_{j\in\{1,\ldots,N\}}\{G_{soc}(\lambda_j)\}}\right)^{iter}\right)\right)^{-1}$$

$$+rand \times (\lambda_{best} - \lambda_i) - rand \times (\lambda_{worst} - \lambda_i).$$

if $G_{soc}(\lambda_i) < G_{soc}(\lambda_i^u)$,
then $\lambda_i = \lambda_i^u$.
endif

Step 6: Update the number of iterations by $iter = iter + 1$.

if $iter \le iter_{max}$,
then go to **Step 4**.
else $\lambda^* = \underset{i\in\{1,\ldots,N\}}{\operatorname{argmax}}\{G_{soc}(\lambda_i)\}$.
endif

Step 7: Output λ^* and $G_{soc}(\lambda^*)$.

The complexity T of the improved Jaya algorithm depends on the number of iterations $iter_{\max}$ and the population size N. The complexity T is given as follows:

$$T = O\left(iter_{\max} \times N\right). \tag{18}$$

By setting the same parameters used in **Figs. 4** and **5** in the improved Jaya algorithm, we provide numerical results of the socially optimal arrival rate λ^* of total tasks, the socially optimal arrival rate $(1-p)\lambda^*$ of the delay sensitive tasks and the maximum social profit $G_{soc}(\lambda^*)$ of the delay sensitive tasks for different shunting probabilities δ in **Table 1**.

Table 1. Numerical results of socially optimal behavior of the delay sensitive tasks

Shunting probability δ	**Socially optimal arrival rate** λ^* **of total tasks**	**Socially optimal arrival rate** $(1-p)\lambda^*$ **of the delay sensitive tasks**	**Maximum social profit** $G_{soc}(\lambda^*)$
0.30	0.5855	0.3513	0.2086
0.35	0.5547	0.3328	0.1609
0.40	0.4928	0.2957	0.1121

Comparing **Figs. 4** and **5**, we find that, for all the shunting probability δ, the Nash equilibrium arrival rates λ^e (marked with "■") of total tasks are always greater than the socially optimal arrival rates λ^* (marked with "□") of total tasks, i.e. $\lambda^e > \lambda^*$. It means that there are more delay sensitive tasks joining the buffer under the Nash equilibrium behavior. To maximize the social profit, an appropriate admission fee should be charged to the delay sensitive tasks to correct the discrepancy between λ^e and λ^* [18].

5.2. Pricing Policy

In order to suppress the Nash equilibrium arrival rate λ^e of total tasks to the socially optimal arrival rate λ^* of total tasks, we present a pricing policy for the delay sensitive tasks. The individual benefit function $G'_{ind}(\lambda)$ of a delay sensitive task with an admission fee f is modified as follows:

$$G'_{ind}(\lambda) = R - CE[T_{ds}] - f. \tag{19}$$

The social profit function $G'_{soc}(\lambda)$ of the delay sensitive tasks can be modified as follows:

$$\begin{aligned} G'_{soc}(\lambda) &= (1-p)\lambda(R - CE[T_{ds}] - f) + (1-p)\lambda f \\ &= (1-p)\lambda(R - CE[T_{ds}]). \end{aligned} \tag{20}$$

Comparing Eqs. (18) and (20), we find that $G_{soc}(\lambda)$ is the same as $G'_{soc}(\lambda)$. That is, the admission fee f is only charged to each delay sensitive task, but it dose not work on the social profit. Actually, the admission fee f is still in the system, it is just transferred from the delay sensitive tasks to the SaaS provider.

By setting $G'_{ind}(\lambda) = 0$ in Eq. (19), we can obtain the admission fee f charged to the delay sensitive tasks as follows:

$$f = R - CE[T_{ds}]. \tag{21}$$

By substituting the socially optimal arrival rate λ^* of total tasks given in **Table 1** into Eq. (14), we can calculate the average response time $E[T_{ds}]$ of the delay sensitive tasks accordingly. Then we obtain the admission fee f using Eq. (21).

In **Table 2**, we present numerical results for the socially optimal arrival rate λ^* of total tasks, the socially optimal arrival rate $(1-p)\lambda^*$ of the delay sensitive tasks and the admission fee f charged to the delay sensitive tasks for different shunting probabilities δ.

Table 2. Numerical results of pricing policy

Shunting probability δ	**Socially optimal arrival rate** λ^* **of total tasks**	**Socially optimal arrival rate** $(1-p)\lambda^*$ **of the delay sensitive tasks**	**Admission fee** f
0.30	0.5855	0.3513	0.5937
0.35	0.5547	0.3328	0.4836
0.40	0.4928	0.2957	0.3790

From **Table 2**, we observe that the admission fee f will decrease as the shunting probability δ increases. Combining **Fig. 2** and **Table 2**, we find that as the shunting probability δ increases, the average response time of the delay sensitive tasks will increase accordingly, then the cost of the delay sensitive tasks will increase. For this case, the delay

sensitive tasks are reluctant to access the system. In order to attract more delay sensitive tasks to access the system, a lower admission fee should be set.

The results in Table 2 provide the cloud operator a reasonable pricing policy to maximize the social profit.

6. Conclusions

In this paper, we addressed the balance problem between the average response time of the tasks and the utilization of resource in cloud computing environment. We proposed a novel task scheduling strategy with service differentiation for the local SaaS server and the public IaaS cloud. By building a hybrid queueing network based system model, we presented the steady-state distribution of the system. Then we estimated the system performance in terms of average response time of the delay sensitive tasks and the utilization of VMs in the reliable cloud. The statistical experiments show that the average response time of the delay sensitive tasks reaches minimum with a certain shunting probability when the arrival rate of total tasks is fixed, while a higher error rate of the fault sensitive tasks leads the VMs in the reliable cloud to be much busier. Moreover, we studied the Nash equilibrium behavior and the socially optimal behavior of the delay sensitive tasks, and carried out an improved Jaya algorithm to obtain the socially optimal arrival rate. Finally, we provided a reasonable pricing policy for the delay sensitive tasks to optimize the system socially.

The achievement in this paper has potential application in managing a public cloud with heterogeneous users. In future work, we will study the task scheduling strategy with adaptive shunting probability.

References

[1] Syed Hamid Hussain Madni, Muhammad Shafie Abd Latiff, Yahaya Coulibaly and Shafi'i Muhammad Abdulhamid, "Resource scheduling for infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities," *Journal of Network and Computer Applications*, vol. 68, no. C, pp. 173-200, June, 2016. Article (CrossRef Link)

[2] M. Jaiganesh, B. Ramadoss, A. Vincent Antony Kumar and S. Mercy, "Performance evaluation of cloud services with profit optimization," in *Proc. of 11th Int. Conf. on Communication Networks*, vol. 54, pp. 24-30, August 21-23, 2015. Article (CrossRef Link)

[3] Qian Huang, "Development of a SaaS application probe to the physical properties of the Earth's interior: An attempt at moving HPC to the cloud," *Computers and Geosciences*, vol. 70, pp. 147-153, September, 2014. Article (CrossRef Link)

[4] Fuhong Lin, Xianwei Zhou, Daochao Huang, Wei Song and Dongsheng Han, "Service scheduling in cloud computing based on queuing game model," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 5, pp. 1554-1566, May, 2014. Article (CrossRef Link)

[5] David Candeia, Ricardo Araujo Santos and Raquel Lopes, "Business-driven long-term capacity planning for SaaS applications," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 290-303, July-Sept, 2015. Article (CrossRef Link)

[6] Song Li, Yangfan Zhou, Lei Jiao, Xinya Yan, Xin Wang and Michael Rung-Tsong Lyu, "Towards operational cost minimization in hybrid clouds for dynamic resource provisioning with delay-aware optimization," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 398-409, May-June, 2015. Article (CrossRef Link)

[7] Zhipiao Liu, Shangguang Wang, Qibo Sun, Hua Zou and Fangchun Yang, "Cost-aware cloud service request scheduling for SaaS providers," *The Computer Journal*, vol. 57, no. 2, pp. 291-301, February, 2013. Article (CrossRef Link)

[8] Ewnetu Bayuh Lakew, Cristian Klein, Francisco Hernandez-Rodriguez and Erik Elmroth, "Performance-based service differentiation in clouds," in *Proc. of 15th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing*, pp. 505-514, May 4-7, 2015. Article (CrossRef Link)
[9] Xiaoming Nan, Yifeng He and Ling Guan, "Towards optimal resource allocation for differentiated multimedia services in cloud computing environment," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 684-688, May 4-9, 2014. Article (CrossRef Link)
[10] Kostas Katsalis, Georgios S. Paschos, Yannis Viniotis and Leandros Tassiulas, "CPU provisioning algorithms for service differentiation in cloud-based environments," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 61-74, March, 2015.
[11] Article (CrossRef Link)
[12] Jianbing Ding, Zhenjie Zhang, Richard Tian Bai Ma and Yin Yang, "Auction-based cloud service differentiation with service level objectives," *Computer Networks*, vol. 94, no. C, pp. 231-249, January, 2016. Article (CrossRef Link)
[13] Behrouz A. Forouzan and Firouz Mosharraf, *Computer Networks: A Top Down Approach*, Mcgraw-Hill, New York, 2012. Article (CrossRef Link)
[14] Oliver C. Ibe, *Fundamentals of Stochastic Networks*, Wiley, New York, 2011. Article (CrossRef Link)
[15] Donald Gross and Carl M. Harris, *Fundamentals of Queueing Theory*, 3rd Edition, Wiley, New York, 2013. Article (CrossRef Link)
[16] Reuven Y. Rubinstein and Dirk P. Kroese, *Simulation and the Monte Carlo Method*, 2nd Edition, Wiley, New York, 2007. Article (CrossRef Link)
[17] Ravipudi Venkata Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 19-34, January, 2016. Article (CrossRef Link)
[18] Amir Hossein Gandomia and Xin-She Yang, "Chaotic bat algorithm," *Journal of Computational Science*, vol. 5, no. 2, pp. 224-232, March, 2014. Article (CrossRef Link)
[19] Refael Hassin and Moshe Haviv, *To Queue or not to Queue: Equilibrium Behaviour in Queueing Systems*, Kluwer Academic Publishers, London, 2003. Article (CrossRef Link)

Yuanzheng Xue received the B.Eng. degree from Hebei Normal University, Shijiazhuang, Hebei, China. Now Miss. Xue is a postgraduate student at School of Information Science and Engineering, Yanshan University, Qinhuangdao, China. Her research area is resource management in cloud computing.

Shunfu Jin received the B.Eng. degree in computer and application from North East Heavy Machinery College, Qiqihaer, China, the M.Eng. degree in computer science and Dr.Eng. degree in circuit and system from Yanshan University, Qinhuangdao, China. Now Dr. Jin is a professor at School of Information Science and Engineering, Yanshan University, Qinhuangdao, China. Her research interests include stochastic modeling for telecommunication, performance evaluation for computer system and network, application for queueing system.

Xiushuang Wang received the B.Eng. degree from Yanshan University, Qinhuangdao, Hebei, China. Now Miss. Wang is a postgraduate student at School of Information Science and Engineering, Yanshan University, Qinhuangdao, China. Her research area is performance evaluation of cloud computing.