Cluster Computing
https://doi.org/10.1007/s10586-018-2841-4

Towards operational cost minimization for cloud bursting with deadline constraints in hybrid clouds

Chunlin Li[1,2,3] · Jianhang Tang[1] · Youlong Luo[1]

Received: 19 October 2017 / Revised: 21 March 2018 / Accepted: 20 August 2018

Abstract
In hybrid clouds, there is a technique named cloud bursting which can allow companies to expand their capacity to meet the demands of peak workloads in a low-priced manner. In this work, a cost-aware job scheduling approach based on queueing theory in hybrid clouds is proposed. The job scheduling problem in the private cloud is modeled as a queueing model. A genetic algorithm is applied to achieve optimal queues for jobs to improve the utilization rate of the private cloud. Then, the task execution time is predicted by back propagation neural network. The max–min strategy is applied to schedule tasks according to the prediction results in hybrid clouds. Experiments show that our cost-aware job scheduling algorithm can reduce the average job waiting time and average job response time in the private cloud. In additional, our proposed job scheduling algorithm can improve the system throughput of the private cloud. It also can reduce the average task waiting time, average task response time and total costs in hybrid clouds.

Keywords Hybrid clouds · Cloud bursting · Load balancing · Genetic algorithm · BP neural network

1 Introduction

Cloud computing has emerged as a new computing paradigm that offers an innovative business model for enterprises [1, 2]. Many enterprises including shopping websites and online social media sites can process the bursting data by applying hybrid clouds [3], as shown in Fig. 1. In the hybrid cloud environment, cloud bursting can handle applications in the private cloud with cheaper expenses and burst into the public cloud when private cloud resources run out [4]. In this scenario, costs of extra public resources should be paid. Therefore, how to reduce total monetary costs in the hybrid cloud environment is a momentous problem [5].

Electronic commerce which is a transaction of buying or selling online has made major changes to the way we live. Many shopping websites offer discounted prices to entice shoppers. For example, Chinese Singles' Day, 11 November, has been the biggest shopping spree in the word. Chinese netizens spent 25 billion dollars buying goods on TaoBao [6] in a 24-h online sale on 11 November 2017. Amazon [7] faces the same situation on the day following Thanksgiving Day named Black Friday. Workloads are much more than usual when the peak comes. If companies build an infrastructure to meet different demands of peak workloads, most of the resources are waste in normal times. Therefore, more and more enterprises apply the cloud bursting to process peak workloads in a low-priced manner.

Cloud bursting can allow companies to expand their capacity to meet the demands of peak workloads in a low-priced manner. However, there are still many challenges in hybrid clouds. Firstly, private cloud resources should be fully utilized. Then, prices and performances of public cloud resources are always different due to the different service providers. How to deploy new applications with the cheapest monetary costs is a complicated problem [8].

✉ Chunlin Li
chunlin74@aliyun.com

[1] School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430063, China

[2] Key Lab of Industrial Internet of Things and Networked Control, Ministry of Education, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

[3] Jiangsu Key Laboratory of Meteorological Observation and Information Processing, Collaborative Innovation Center of Atmospheric Environment and Equipment Technology, Nanjing University of Information Science and Technology, Nanjing 210044, China

Published online: 25 August 2018

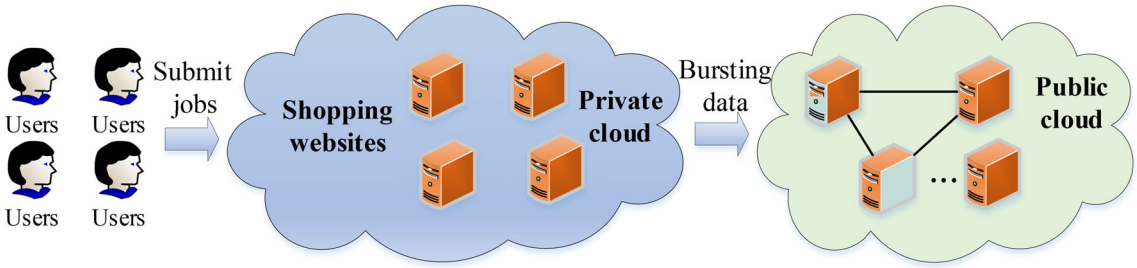


Fig. 1 Application scenarios in hybrid clouds

To solve above challenges, a cost-aware job scheduling approach based on queueing theory in hybrid clouds is proposed. The main idea of this approach is as follows: jobs are dispatched to corresponding job queues in the private cloud. If the private cloud cannot meet the demands of users, the public cloud with the cheapest costs will be rented. Therefore, the utilization rate of the private cloud is improved and total costs of hybrid clouds drop greatly.

The main contributions of this paper are shown as follows:

(1) In order to improve the utilization rate of the private cloud and reduce total costs of hybrid clouds under different deadline constraints, a cost-aware job scheduling approach based on queueing theory in hybrid clouds is proposed.
(2) For the purpose of improving the utilization rate of the private cloud, a load balancing job scheduling method in the private cloud is proposed. The job scheduling problem in the private cloud is modeled as a queueing model. A genetic algorithm (GA) is given to dispatch jobs to optimal queues in the private cloud.
(3) In order to reduce the monetary cost of hybrid clouds under different deadline constraints, the execution time of each task is predicted by back propagation (BP) neural network. If some tasks cannot meet the deadline in the private cloud according to the prediction results, the public cloud with the cheapest monetary costs will be applied. Experiments show that our algorithm can improve the utilization rate of private cloud and reduce total monetary costs of the hybrid clouds.

The rest of the paper is organized as follows: Sect. 2 is about related works. Section 3 presents the cost-aware job scheduling approach based on queueing theory in hybrid clouds. Section 4 describes the details of the implementation of our proposed algorithms. Section 5 presents the comparison and analysis of experiment results. We make some conclusions in Sect. 6.

2 Related work

In this part, first we review the recent efforts focusing on the cloud bursting, and then discuss the previous studies about the scheduling algorithms in hybrid clouds.

2.1 Cloud bursting in hybrid clouds

Many researchers have done lots of work about cloud bursting to obtain elastic cloud resources with the cheapest costs in hybrid clouds. Loreti and Ciampolini [9] presented a software layer which can deploy and dynamically scale the virtual clusters in hybrid clouds. This system was proposed to deal with cloud bursting in case of big data. Acs et al. [10] presented the criteria for idealistic cloud bursting and a technique to reduce the complexity of the cloud bursting procedure. This technique used nested virtualization. Farahabady et al. [11] proposed a framework named PANDA for static scheduling Bag-of-Tasks (BoTs) applications in hybrid clouds. This framework can effectively leverage the monetary cost of hybrid clouds. Farokhi et al. [12] presented a hybrid vertical memory elasticity controller which was designed by the feedback control loop. This hybrid controller can meet application performance targets with better stability. Clemente-Castelló et al. [13] studied how to combine various data locality techniques designed for hybrid cloud bursting to obtain scalability for iterative applications in a cost-effective way. Lee and Lian [14] proposed a new cloud bursting algorithm which considered time-varying electricity rates with the private cloud and the time-invariant rental rate of public clouds. Charrada and Tata [15] proposed an algorithm named forward–backward-refinement (FBR) which was adapted for service-based applications. This FBR algorithm had a good performance not only for architecture-based compositions but also for architecture-based compositions of services.

Although there are many methods of cloud bursting in hybrid clouds mentioned above, none of those approaches considered different average service rate in the private cloud.

2.2 Scheduling algorithm in hybrid clouds

More and more enterprises have employed hybrid clouds to meet the demands of peak workloads. It is significant to study the scheduling algorithms of hybrid clouds. Zhang et al. [16] proposed a heuristic algorithm to schedule tasks in hybrid clouds with constraints of resource demands and costs, which

included two task sequencing and task scheduling phases. A Longest Task First method was applied to generate a task sequence. A Task Assignment method is proposed to schedule all tasks in the obtained sequence. Daniel and Raviraj [17] applied the Fast Quadratic Lyapunov Algorithms in different time granularities to schedule and reschedule of multimedia contents to improve the profit of multimedia cloud service providers in distributed hybrid clouds. Li et al. [18] proposed a distributed scheduling algorithm for resource intensive mobile applications in the hybrid cloud environment. This scheduling algorithm included local cloud agent scheduling, public cloud service scheduling and mobile application QoS optimization. Zhu et al. [19] studied a special workflow scheduling problem in a hybrid-cloud-based workflow management system. In this system, tasks were linearly dependent, compute-intensive and stochastic, which closely resembled real-time and workflow-based applications. Zuo et al. [20] presented a task-oriented multi-objective scheduling algorithm. This algorithm can optimize the finite pool of computing resources in hybrid clouds under deadline and cost constraints by ant colony optimization. Champati and Liang [21] studied joint task scheduling and offloading problem in hybrid cloud environment. They did not assume that task processing times were known a priori. They proposed a Greedy-One-Restart algorithm which can estimate the processing times of tasks and reschedule those tasks that turn out to need long processing times.

In summary, the above scheduling approaches in hybrid clouds have been applied widely. However, the private cloud should be the main part of hybrid clouds. It is significant to improve the utilization rate of the private cloud and reduce total costs of hybrid clouds. Moreover, an accurate prediction of the execution time of tasks is a crucial step for scheduling. Our proposed scheduling approach can reduce the monetary cost of hybrid clouds while improving the utilization rate of private cloud.

3 Cost-aware job scheduling approach based on queueing theory in hybrid clouds

First, we introduce the cost-aware job scheduling model in this section. Then, a load balancing job scheduling method in the private cloud is proposed. Finally, an optimal task scheduling method based on BP neural network in hybrid clouds is proposed.

3.1 Cost-aware job scheduling model in hybrid clouds

In order to meet the elastic demands of users in hybrid clouds, the private cloud should be fully utilized and the public cloud with the cheapest monetary costs should be applied. Therefore, our proposed cost-aware job scheduling approach based on queueing theory in hybrid clouds included two components: job scheduling method in the private cloud and task scheduling method in hybrid clouds, as shown in Fig. 2. In the private cloud, job scheduling problem is modeled as a queueing model. A GA is applied to achieve the optimal job arrival rates for all job queues according to the queue states. Then, the private cloud resources can be fully leveraged. After jobs are scheduled in the private cloud, the execution time of each task of these jobs is predicted by the BP neural network. Based on the prediction results, the max–min strategy is applied to dispatch tasks to the resources of hybrid clouds under different deadline and cost constraints. The tasks will be dispatch to the public cloud resources with the cheapest costs under different cost constraints. Therefore, the costs of the hybrid clouds drop greatly.

The main notations are summarized in Table 1.

3.2 Load balancing job scheduling method in the private cloud

In hybrid clouds, jobs submitted by users are dispatched to the private cloud first. In order to improve the utilization rate of the private cloud, the load balancing job scheduling method in the private cloud is proposed. Jobs are submitted to the private cloud successively. This paper supposes that the inter-arrival time of coming jobs obey the negative

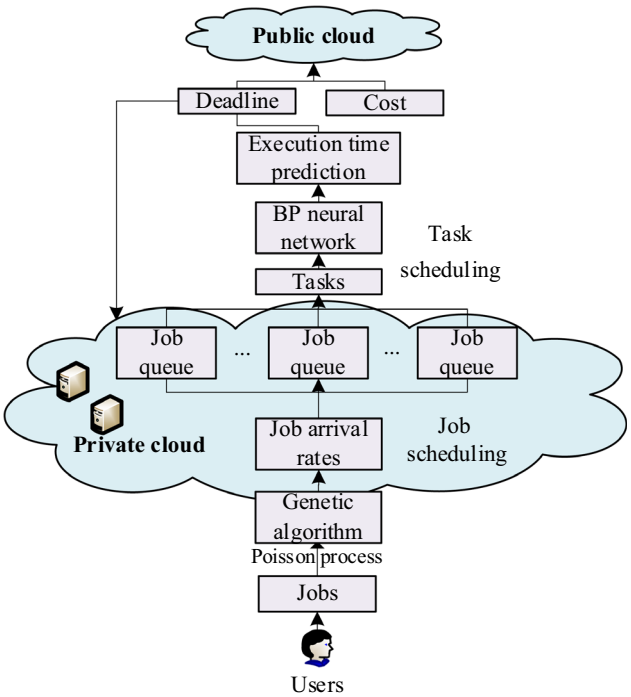


Fig. 2 Cost-aware job scheduling model in hybrid clouds

Table 1 Summary of notations

Notations	Definition
N_q	The number of job queues
λ	The total job arrival rate
μ_j	The average service rate of queue j
Q_j^{avg}	The average number of waiting jobs in queue j
Q_j^{ini}	The initial number of jobs in queue j
T_j^r	The average response time of queue j
T_j^w	The average waiting time of queue j
T_j^s	The average service time of queue j
T_{total}^r	The total response time of private cloud
p_j	The probability of assigning a job to queue j
λ_j	The arrival rate of queue j
JP_i	Priority of job i
$Container_R$	Resource R of hybrid clouds
J_i	The ith job
m_{ik}	Map task k of job i
$r_{ik'}$	The reduce task k' of job i
$mCost[i, k, R]$	The cost of map task k of job i in hybrid cloud resource R
$rCost[i, k', R]$	The cost of reduce task k' of job i in hybrid cloud resource R
$mEEt[i, k, R]$	The execution time of map task k of job i in hybrid cloud resource R
$rEEt[i, k', R]$	The execution time of reduce task k' of job i in hybrid cloud resource R
Rft_R	The finish time of current tasks in hybrid cloud resource R

exponential distribution. Let λ be the average job arrival rate of coming jobs per unit interval to the private cloud. Then, submission process and service process of jobs can be regarded as a queuing model. In this work, there are N_q data centers in the private cloud. We assume that there is a $M/M/c$ queuing model for data center j with average service rate μ_j for c parallel servers. Due to limited resources in each data center, they can only serve c jobs simultaneously. Additional jobs should wait in a queue to be processed latter. Assume that data center j consists of jth $M/M/c$ queue. The stabilizing condition of the $M/M/c$ queueing model is

$$\rho_j = \frac{\lambda_j}{c\mu_j} < 1, \tag{1}$$

where λ_j denotes the average job arrival rete of queue j and ρ_j means the service strength of queue j in the private cloud. And we have $\sum \lambda_j = \lambda$. Then, the probability P_{jn} that n jobs are being executed in queue j can be achieved as follows,

$$P_{jn} = \begin{cases} \dfrac{\lambda_j^n}{n! \times \mu_j^n} P_{j0}, & n < c, \\ \dfrac{\lambda_j^n}{c! \times \mu_j^n \times c^{(n-c)}} P_{j0}, & n \geq c, \end{cases} \tag{2}$$

where P_{j0} denotes the probability that there is no job in queue j, which can be calculated by Eq. (3),

$$P_{j0} = \left\{ \left[\sum_{n=0}^{c-1} \frac{(c\rho_j)^n}{n!} \right] + \left[\frac{(c\rho_j)^c}{c!(1-\rho_j)} \right] \right\}. \tag{3}$$

In addition, the probability that a new arrival job should wait in queue j can be achieved according to Eq. (4)

$$P_{j\infty} = \sum_{n=c}^{\infty} P_{jn} = \frac{(c\rho_j)^c P_{j0}}{c!(1-\rho_j)}. \tag{4}$$

It leads to achieving average number of clients in a $M/M/c$ queuing model, Q_{avg}, as shown in Eq. (5),

$$Q_j^{avg} = \frac{\rho_j P_{j\infty}}{1-\rho_j}. \tag{5}$$

Therefore, according to the Little's law, the response time of queue j can be obtained by addition of waiting time and service time as shown in Eq. (6)

$$T_j^r = T_j^w + T_j^s = \frac{Q_j^{avg} + Q_j^{ini}}{\lambda_j} + \frac{1}{\mu_j}, \tag{6}$$

where Q_{init} indicates the initial number of jobs in queue j from previous assignments. The total response time of the private cloud can be achieved from Eq. (7),

$$T_{total}^r = \sum_{j=1}^{N_q} p_j T_j^r, \tag{7}$$

where $p_j = \lambda_j/\lambda$ is the probability of assigning a job to queue j.

According to above equation, total response time T^r_{total} of the private cloud is a function of variable λ_j, which is followed by constraints equation (8),

$$\begin{aligned} &\min\ T^r_{total} = \sum_{j=1}^{N_q} p_j T^r_j = \sum_{j=1}^{J} \frac{\lambda_j}{\lambda}\left(\frac{Q^{avg}_j + Q^{ini}_j}{\lambda_j} + \frac{1}{\mu_j}\right) \\ &s.t.\ \sum_{j=1}^{N_q} \lambda_j = \lambda. \end{aligned} \tag{8}$$

As shown in Eq. (8), it is quite difficult to achieve global optimal solutions by applying deterministic polynomial time algorithms. GA are considered as an efficient and stable artificial intelligent technique to search out global optimum solutions, especially in the complex and vast search space. In our work, GA is applied to obtain optimal average job arrival rates for job queues, which can dispatch jobs to job queues according to the initial states and average job service rates. In order to achieve the optimum solutions, GA needs multiple iterations to obtain the convergence. The high number of iterations will cause high time complexity in GA. An unsuitable initial population may reduce efficiency for GA. In this paper, an initial population is generated based on current workloads and resource utilization rates of queues to narrow down the solution domain.

The factors that affects workloads of a queue include the initial amount of jobs and resource utilization rate. The resource utilization rate of queue j can be calculated by addition of resource utilization rate of virtual CPU cores and resource utilization rate of memory as depicted in Eq. (9),

$$Ru_j = \delta_1 \frac{usedVCPU_j}{VCPU_j} + \delta_2 \frac{usedMem_j}{Mem_j}, \tag{9}$$

where $usedVCPU_j$ denotes the number of virtual CPU cores which have been used in queue j, $VCPU_j$ represents the total number of virtual CPU cores, $usedMem_j$ indicates the used memory of queue j, and Mem_j indicates the total memory of queue j. And $\delta_1 + \delta_2 = 1$.

Then, the total efficacy coefficient of queue j can be calculated from Eq. (10),

$$\eta^{(j)} = \sqrt{\frac{InitQ_{\max} - InitQ_j}{InitQ_{\max} - InitQ_{\min}} \cdot \frac{Ru_{\max} - Ru_j}{Ru_{\max} - Ru_{\min}}}. \tag{10}$$

$InitQ_{\max}$ and $InitQ_{\min}$ indicates the maximal and minimal initial amount of jobs in all queues in the private cloud. $InitQ_j$ is the initial amount of jobs in data center j. $Ru_{\max}$ and $Ru_{\min}$ denote the maximal and minimal resource utilization rate of all queues in the private cloud, respectively. Ru_j indicates the resource utilization rate of queue j.

Then, the initial probability p_j of dispatching a job to queue j in the private cloud can be calculated, as shown in Eq. (11),

$$p_j = \frac{\eta^{(j)}}{\sum_j \eta^{(j)}}. \tag{11}$$

In a GA, the greater similarities among individuals in each generation, the smaller fitness differences among individuals will be. It means that the evolution of the population tends to converge. In this case, the crossover probability should be decreased and the mutation probability should be increased.

Let EX be the mean value of fitness values and DX be the fitness variance, as shown in Eq. (12),

$$\begin{cases} EX = \dfrac{1}{G}(f_1 + f_2 + \cdots + f_G), \\ DX = \dfrac{1}{G}\sum_{\omega=1}^{G}(f_\omega - EX)^2, \end{cases} \tag{12}$$

where G is the population size, and f_ω is the fitness of individual ω as depicted in Eq. (13),

$$f_\omega = \frac{1}{T^r_{total}}. \tag{13}$$

The similarity can be achieved from Eq. (14),

$$factor = \frac{EX + 1}{\sqrt{DX}}. \tag{14}$$

Hence, crossover probability p_c and mutation probability p_m can be defined in Eq. (15),

$$\begin{cases} p_c = \dfrac{1}{1 + e^{-2factor^{-1}}} - 0.1, \\ p_m = \dfrac{0.1}{7 \cdot (1 + e^{factor^{-1}})}. \end{cases} \tag{15}$$

3.3 Optimal task scheduling method based on BP neural network in hybrid clouds

3.3.1 Execution time prediction based on BP neural network

Most of jobs are executed cyclically in the large-scale system. It means that each job with different data will be executed again and again. Therefore, it is significant to analyze the historical information of tasks of these jobs. In this paper, there are two BP neural networks. One is proposed to predict the execution time of each map task, and the other one is proposed to predict the execution time of each reduce task. The factors that affect the execution time of map tasks or reduce tasks are the inputs for BP neural networks. $X_{map} = [x_1, x_2, \ldots, x_{map}]$ is the input vector for BP neural network of each map task and $Y_{reduce} = [y_1,$

Table 2 Factors of map task or reduce task

Parameter and notations	Description and usage
nodePerformance	The I/O performance and CPU performance of the node
nodeLoad	The resource utilization rate of the node
nodeBandwidth	Node bandwidth
mapComplexity	The computational complexity of the map task
mapDataSize	The data size of the map task

$y_2,\ldots,y_{reduce}]$ is the input vector for BP neural network of each reduce task. The details of $X_{map} = [x_1, x_2,\ldots,x_{map}]$ or $Y_{reduce} = [y_1, y_2,\ldots,y_{reduce}]$ are shown in Table 2.

Each output layer has a one-dimensional output vector which denotes the execution time of each map or reduce task. Let T_{reduce} and T_{reduce} denote the execution time of the map and reduce task, respectively.

3.3.2 Task scheduling based on execution time prediction in hybrid clouds

In hybrid clouds, users submit jobs with many demands, such as deadline and cost constraints. In order to meet the demands of particular jobs, a priority of each job is proposed according to its workloads, deadline and cost constraints. Jobs in each queue will be sorted according to their priorities. Let J_i be the job i with deadline D_i and cost constraint C_i. Then, the priority of job i can be defined in Eq. (16),

$$JP_i = \frac{DataSize_i + Complexity_i}{D_i} + C_i. \tag{16}$$

Let $J_i = \{m_{i1}, m_{i2},\ldots,m_{im}, r_{i1}, r_{i2},\ldots,r_{ir}\}$ be a job that contains m map tasks and r reduce tasks. In this work, job i is defined as a three tuple $U = \{J_i, D_i, C_i\}$ where D_i denotes the deadline, C_i denotes the cost constraint. In our paper, we suppose that the private is costless. Hence, C_i just indicates the cost of the public cloud. Map task m_{ik} of job i is defined as a two tuple $m_{ik} = \{mW_{ik}, mD_{ik}\}$ where mW_{ik} indicates the workload and mD_{ik} indicates the input data size. And reduce task $r_{ik'}$ is defined as a two tuple $r_{ik'} = \{rW_{ik'}, rD_{ik'}\}$ where $rW_{ik'}$ indicates the workload and $rD_{ik'}$ indicates the input data size. We apply the number of instructions to denote the task workload. The relationship between m_{ik} and $r_{ik'}$ can be obtained as follows,

$$rD_{ik'} = \sum_{k=1}^{\theta} f_k \times mD_{ik}, \quad 0 < f_k < 1, \tag{17}$$

where f_k denotes the ratio of the input data size of reduce task k' to the input data size of map task k and θ indicates the number of map tasks handled by a reduce task.

Let C_R ($R \in \{1, 2,\ldots,N_R\}$) represent a container which indicates a logical bundle of cloud computing resources (e.g., $\langle 2$ CPU, 4 GB RAM$\rangle$) bound to a particular node in hybrid clouds. It can be defined as Eq. (18),

$$C_R = \{Cost_R, Stg_R, Cin_R, Cout_R, Band_R, Rft_R\}, \tag{18}$$

where $Cost_R$ (\$/MIs) represents the monetary cost per million instructions of resource R, Stg_R (\$/MBs) represents the storage cost of resource R, Cin_R (\$/s) represents the transmission cost of input data, $Cout_R$ (\$/s) represents the transmission cost of output data, $Band_R$ (MB/s) represents the network bandwidth of resource R, Rft_R represents the finish time of the current task in resource R. In this paper, the monetary cost of the private cloud is costless. Thus, we set $Cost_R = 0$, $Stg_R = 0$, $Cin_R = 0$ and $Cout_R = 0$ in the private cloud.

Enterprises that provide public cloud services will charge the service fee that includes the computing cost, storage cost and data transmission cost. Therefore, the total monetary cost can be calculated by addition of computing cost, storage cost and data transmission cost. The cost of map task $mCost[i, k, R]$ and reduce task $rCost[i, k', R]$ in the public cloud can be achieved in Eq. (19),

$$\begin{cases} mCost[i,\ k,\ R] = Cost_R \cdot mW_{ik} + Stg_R \cdot mD_{ik} + mDtt_{ik} \cdot (Cin_R + Cout_R) + \dfrac{\rho \cdot mD_{ik}}{Band_R} \cdot Cout_R, \\ rCost[i,\ k',\ R] = Cost_R \cdot rW_{ik'} + Stg_R \cdot rD_{ik'} + rDtt_{ik'} \cdot (Cin_R + Cout_R) + \dfrac{rD_{ik'}}{Band_R} \cdot Cin_R, \end{cases} \tag{19}$$

where ρ is the ratio of the amount of map output data to the amount of map input data.

Let $mEEt[i, k, R]$ represent the estimated execution time of map task k of job i in $Container_R$ and $rEEt[i, k', R]$ represent the estimated execution time of reduce task k' of

job i in $Container_{R'}$. The estimated execution time can be calculated by addition of execution time predicted by BP neural network and data transmission time, as shown in Eq. (20),

$$\begin{cases} mEEt[i,\ k,\ R] = T_{map} + \dfrac{mD_{ik}}{Band_R}, \\ rEEt[i,\ k',\ R] = T_{reduce} + \dfrac{mD_{ik}}{Band_R}. \end{cases} \tag{20}$$

Then, the task scheduling problem in hybrid clouds can be formulated as follows,

$$\min \left(\sum_{R=1}^{N_R} \sum_{k=1}^{m} mCost[i,\ k,\ R] \times a_{i,k,R} + \sum_{R=1}^{N_R} \sum_{k'=1}^{r} rCost[i,\ k',\ R] \times a_{i,k',R} \right)$$

$$\min \left(\max_R \left(m_R \left(\sum_{k=1}^{m} mEEt[i,\ k,\ R] \cdot a_{i,k,R} + Rft_R \right) \right) + \max_R \left(r_R \left(\sum_{k'=1}^{r} rEEt[i,\ k',\ R] \cdot a_{i,k',R} + Rft_R \right) \right) \right)$$

$$s.t. \begin{cases} \left(\max_R \left(m_R \left(\sum_{k=1}^{m} mEEt[i,\ k,\ R] \cdot a_{i,k,R} + Rft_R \right) \right) + \max_R \left(r_R \left(\sum_{k'=1}^{r} rEEt[i,\ k',\ R] \cdot a_{i,k',R} + Rft_R \right) \right) \right) \le D_i \\ \left(\sum_{R=1}^{N_R} \sum_{k=1}^{m} mCost[i,\ k,\ R] \times a_{i,k,R} + \sum_{R=1}^{N_R} \sum_{k'=1}^{r} rCost[i,\ k',\ R] \times a_{i,k',R} \right) \le C_i \\ a_{i,k,R} \in \{0,\ 1\},\ \sum_{R=1}^{N_R} a_{i,k,R} = 1 \\ a_{i,k',R} \in \{0,\ 1\},\ \sum_{R=1}^{N_R} a_{i,k',R} = 1 \\ m_R \in \{0,\ 1\},\ m_R = \vee_{k=1,2,\ldots,m} a_{i,k,R} \\ r_R \in \{0,\ 1\},\ r_R = \vee_{k'=1,2,\ldots,r} a_{i,k',R}, \end{cases}$$

where $a_{i,k,R}$ and $a_{i,k',R}$ represent the binary picking variables. They denote whether a task is assigned to container R or not. m_R and r_R denote whether container R is used or not.

This programming problem can be regarded as a variation of the knapsack problem, which is a NP-hard problem in hybrid clouds. Therefore, a heuristic method that applies the max–min strategy to reduce the time complexity is proposed. First, tasks are sorted according to their attributes. A task which is failed recently will be set to a high priority. Hence, the priority of a failed map task is set to 5, the priority of a failed reduce task or reduce task is set to 10 and the priority of a map task is set to 20 respectively. That is to say, if a failed map task and a reduce task request apply resource simultaneously, resources will be assigned to the failed map task first. Then, tasks with high priority will be allocated to resources with the minimal finish time in the private cloud. If the private cloud cannot meet the deadline, the public cloud with cheapest monetary costs will be applied.

4 The algorithm of cost-aware job scheduling approach based on queueing theory in hybrid clouds

Our proposed cost-aware job scheduling algorithm includes the load balancing job scheduling algorithm in the private cloud and the optimal task scheduling algorithm in hybrid clouds, as shown in Fig. 3. The load balancing job scheduling algorithm is proposed to dispatch jobs to optimal queues in the private cloud, which can improve the utilization rate of private cloud resources. The optimal task scheduling algorithm in hybrid clouds consists of two sub-algorithms, optimal task scheduling algorithm in the private cloud and optimal task scheduling algorithm in the public cloud. The optimal task scheduling algorithm in the private cloud is proposed to schedule tasks in the private cloud and obtain the set of tasks that cannot meet the deadline constraint in the private cloud. The optimal task scheduling algorithm in the public cloud is proposed to schedule the tasks that cannot be processed in the private cloud to public cloud resources under different deadline and cost constraints.

4.1 Load balancing job scheduling algorithm in the private cloud

Algorithm 1 shows the pseudo-code of the load balancing job scheduling algorithm in the private cloud. First, optimal queue choice probability p_j is achieved by the GA (Algorithm 1 Lines 1– 3). Finally, jobs are dispatched to queues according to p_j respectively (Algorithm 1 Lines 4–6).

Algorithm 1: Load balancing job scheduling algorithm in the private cloud

Input: $\mathrm{Job}[1,2...,I]$, $\mathrm{Queue}[1,2,...,J]$

Output: Job scheduling result $\mathrm{HashMap}\langle \mathrm{Job}[1,2...,I], \mathrm{Queue}[1,2,...,N_{dc}]\rangle$

1 **for each** $\mathrm{queue} \in \mathrm{Queue}[1,\ldots,J]$ **do**

2 $p_j \leftarrow Genetic(\mathrm{Queue}[1,\ldots,J])$

3 **end for each**

4 **for each** $\mathrm{job} \in \mathrm{Job}[1,\ldots,I]$ **do**

5 This job will be dispatched to a queue according to p_j

6 **end for each**

7 **return** $\mathrm{HashMap}\langle \mathrm{Job}[1,\ldots,I], \mathrm{Queue}[1,\ldots,J]\rangle$

4.2 Optimal task scheduling algorithm in hybrid clouds

The optimal task scheduling algorithm in hybrid clouds consists of the optimal task scheduling algorithm in the private cloud and optimal task scheduling algorithm in the public cloud.

4.2.1 Optimal task scheduling algorithm in private cloud

Algorithm 2 shows the pseudo-code of the optimal task scheduling algorithm in the private cloud. First, tasks are sorted according to their priorities (Algorithm 2, Line 2). Then, the execution time of a map task is predicted and this map task is dispatched to the resources with shortest finish time (Algorithm 2, Lines 7–12). The execution time of a reduce task is predicted and this reduce task is assigned to the resources with shortest finish time (Algorithm 2, Lines 13–19). Finally, if a task can meet the deadline constraint in the private cloud, this task will be executed in the private cloud. Otherwise, the public cloud will be applied (Algorithm 2, Lines 22–26).

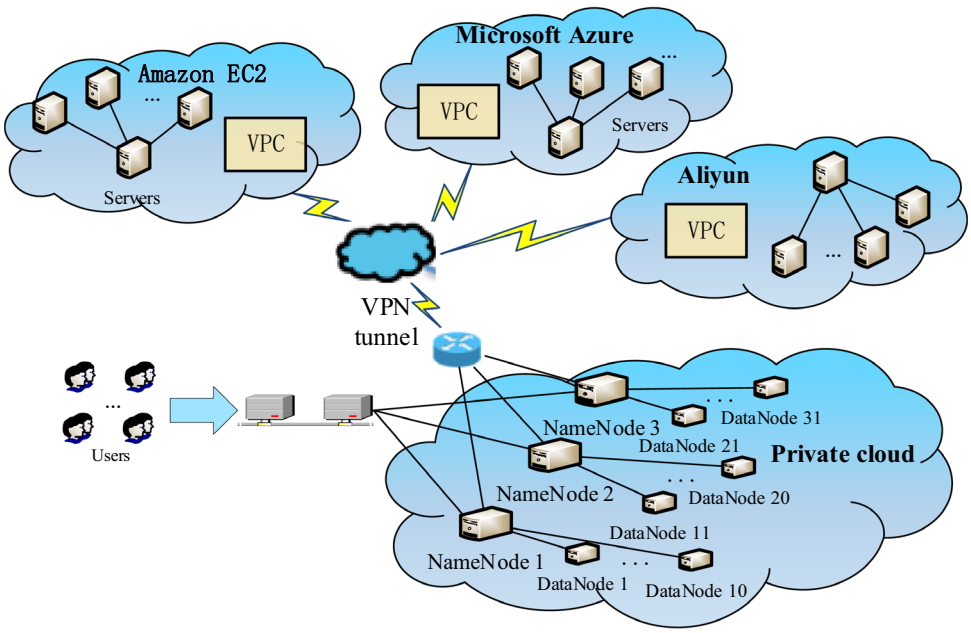


Fig. 3 The experimental environment

Algorithm 2: Optimal task scheduling algorithm in the private cloud

Input: The submitted job J_i, available resources of private cloud ARP

Output: Tuple $\langle PriM_i, TRP_i \rangle$ // $PriM_i$ records the task mapping of job i in the private cloud, TRP_i represents a task set that requires the public cloud

1: $PriM_i \leftarrow \varnothing$, $TRP_i \leftarrow \varnothing$
2: **sort(**Task priority**)**
3: **for each** $task \in J_i$ **do**
4: $resourceR \leftarrow$ the first resource; $ftR \leftarrow \infty$; $tmp \leftarrow Rft_R$
6: **for each** $R \in APR$ **do**
7: **if** $task$ is a map task **then**
8: $T_{map} \leftarrow BP(X_{map})$ // BP neural network predict the execution time of a map task
9: **if** $mEEt[i,k,R] + Rft_R < ftR$ **then**
10: $ftR \leftarrow mEEt[i,k,R] + Rft_R$
11: $resourceR \leftarrow R$
12: **end if**
13: **else** $task$ is a reduce task
14: $T_{reduce} \leftarrow BP(X_{reduce})$ // BP neural network predict the execution time of a reduce task
15: **if** $rEEt[i,k',R] + Rft_R < fitR$ **then**
16: $ftR \leftarrow rEEt[i,k',R] + Rft_R + \max Ft(map\ task)$ // $\max Ft(map\ task)$ denotes the maximum finish time of map tasks which be handled by reduce task k'
17: $resourceR \leftarrow R$
18: **end if**
19: **end if**
20: **end for**
21: $Rft_R \leftarrow fitR$
22: **if** $Rft_R > D_i$ **then**
23: $TRP \leftarrow task_k$; $Rft_R \leftarrow tmp$
24: **else**
25: Dispatch $task_k$ to $resourceR$; Update ($PriM_i$)
26: **end if**
27: **end for**
28: **return** $\langle PriM_i, TRP_i \rangle$

4.2.2 Optimal task scheduling algorithm in the public cloud

Algorithm 3 shows the pseudo-code of our optimal task scheduling algorithm in the public cloud. First, public cloud resources that can meet the deadline constraints and need the cheapest monetary costs are selected (Algorithm 3, Lines 3–12). Then, if the public cloud resources exist, the instances of the public cloud are created and tasks are dispatched to the public cloud. Otherwise, the system returns $\langle Null, Null, False \rangle$ (Algorithm 3, Lines 13–17). Finally, if the monetary costs can meet the cost constraints, the system returns $\langle PubRE_i, PubM_i, True \rangle$. Otherwise, the system returns $\langle PubRE_i, PubM_i, False \rangle$.$\langle PubRE_i, PubM_i, False \rangle$. (Algorithm 3, Lines 19–23).

Algorithm 3: Optimal task scheduling algorithm in public cloud

Input: TRP_i represents a task set that requires public cloud resources, $PubR$ represents the public resource set, T_{init} denotes initialization time of public resource containers

Output: Triples $\langle PubRE_i, PubM_i, RT_i \rangle$ // $PubRE_i$ indicates the set of containers of public cloud resources to execute tasks. $PubM_i$ records the mapping of tasks assigned to public cloud resources , RT_i represents whether the task scheduling results meet the deadline and cost constraints or not.

```
PubRE_i ← ∅ ; PubM_i ← ∅ ; SumCost_i ← ∅
for each task ∈ TRP_i do
    for each R ∈ PubR
        if task is a map task then
            SumCost_k ← SumCost_k + mCost[i,k,R]
            T_task ← mEEt[i,k,R]
        else task is a reduce task
            SumCost_k ← SumCost_k + rCost[i,k',R]
            T_task ← rEEt[i,k',R] + max Ft(map task)
        end if
    end for
    Select the public resource R with minimum monetary cost subject to T_task + T_init < D_i
    if R exists then
        Create R ; dispatch task_k into R ; Update ( PubM_i )
    else
        return ⟨Null, Null, False⟩
    end if
end for
if SumCost_i < C_i then
    return ⟨PubRE_i, PubM_i, True⟩
else
    return ⟨PubRE_i, PubM_i, False⟩
end if
```

Then, we analyze the time complexity of our proposed algorithms. In our load balancing job scheduling algorithm, the number of generations is *gens*, the time complexity of the mutation is $O(G{\cdot}N_{Job})$ where G is the number of population numbers in a population, the time expense of the crossover is $O(G{\cdot}N_{Job})$, the time expense of the selection is $O(G)$. The total time complexity of the GA is $O(gens{\cdot}G{\cdot}N_{Job})$ where the number maximum of generations is *gens*. Thus, the total time complexity of our load balancing job scheduling algorithm in the private cloud is $O(gens{\cdot}G{\cdot}N_{Job})$. In our optimal task scheduling algorithm in hybrid clouds, the number of hidden layers is N_h in the BP neural network. After training a BP neural network, the time complexity of the task execution time prediction is $O(N_{task}{\cdot}N_h)$ where N_{task} denotes the number of tasks submitted by users. The number of tasks which are dispatched to the public cloud is N'_{task}. The time complexity of task scheduling in the private cloud is $O((N_{task} - N'_{task}) \cdot R \cdot N_h)$ where R denotes the number of private cloud resources. The time cost of task scheduling in the public cloud is $O((N'_{task}){\cdot}r{\cdot}N_h)$ where r denotes the number of public cloud resources. The time expense of our proposed optimal task scheduling algorithm in hybrid clouds is $O((N_{task} - N'_{task}) \cdot R \cdot N_h + (N'_{task}) \cdot r \cdot N_h)$. Then, the total time complexity of our proposed cost-aware job scheduling algorithm based on queueing theory in hybrid clouds is $O(gens{\cdot} G \cdot N_{Job} + (N_{task} - N'_{task}) \cdot R \cdot N_h + (N'_{task}) \cdot r \cdot N_h)$.

5 Performance evaluation

5.1 Experimental environment

5.1.1 Experimental setup

In this part, our proposed cost-aware job scheduling approach based on queueing theory in hybrid clouds will be experimentally verified. For this experiment, the operating system is Ubuntu 14.04.1 LTS. The Strongswan 5.5.1, OpenVPN 2.3.2 and Flex GateWay are deployed to connect the private cloud and the public cloud. The Hadoop 2.7.1 is adopted to perform the experiment. The version of the Java Development Kit is Java SE 8u131. The development environment is Linux Eclipse 4.5.0. The private cloud includes 3 NameNode and 30 DataNodes with different configurations. The experimental configuration is shown in Fig. 3.

The private cloud configurations are shown in Table 3. The public cloud providers include Aliyun, Amazon Elastic Compute Cloud (EC2) and Microsoft Azure. The instance types of Microsoft Azure, Amazon (EC2) and Aliyun comprise varying combinations of CPU, memory, networking capacity and costs. Microsoft Azure instances offer 2–8 vCPUs, 3.5–14 GB of RAM memory and 10–30 Mbps network bandwidth. Amazon (EC2) instances offer 2–8 vCPUs, 8–32 GB of RAM memory and 10–40 Mbps network bandwidth. Aliyun instances offer 1–8 vCPUs, 1–16 GB of RAM memory and 10–50 Mbps network bandwidth. The public cloud configurations are shown in Table 4.

In hybrid clouds, the network bandwidth between private cloud and public cloud is 10 Mbps. In the network of the private cloud, all servers can communicate with each other. The network bandwidth in the private cloud is 30 Mbps. The network bandwidth in the public cloud is achieved according to the different instance types.

Users communicate with the private cloud and public cloud via HTTP protocol. SSL/TLS is used to provide encryption and secure credentials. Public Key Infrastructure is used for authentication. Virtual Private Cloud (VPC) enables users to access the public cloud resources over an IPsec tunnel by setting up a Virtual Private Network (VPN), which allows the use of hybrid clouds.

5.1.2 Test case

In our experiments, the data set comes from Stanford Network Analysis Project (SNAP) [22]. The data set contains purchasing information and comments of about 1 million commodities of Amazon. The average data sizes of data sets are set to 32 and 80 GB, respectively. The proportions of the various types of jobs are shown in Table 5. In addition, the data size of each split is 128 MB.

Table 3 Private cloud configurations

Hosts	CPU/memory/disk	IP
NameNode 1	Intel i7 (4 cores with each 3.6 GHz)/16 GB/2 TB	192.168.203.10
DataNode 1–10	Intel i5 (4 cores with each 3.3 GHz)/4 GB/500 GB	192.168.203.20–192.168.203.29
NameNode 2	Core 4/16G/2T	115.28.19.197
DataNode 11–20	Core 4/4G/500G	10.144.17.24–10.144.17.33
NameNode 3	Core 4/16G/2T	105.36.48.175
DataNode 21–30	Core 4/4G/500G	10.136.25.65–10.136.25.74

Table 4 Public cloud configurations

Facilitators	Instance types	CPU	Cost (10^{-6} $/s)
Microsoft Azure	Core 2, RAM 3.5 GB, bandwidth 10 Mbps	Intel Xeon(R) E5-2650 v2 2.60 GHz	28
	Core 4, RAM 7 GB, bandwidth 20 Mbps		73
	Core 8, RAM 14 GB, bandwidth 30 Mbps		228
Amazon EC2	Core 2, RAM 8 GB, bandwidth 10 Mbps	Intel Xeon(R) E5-2676 v3 2.4 GHz	27
	Core 4, RAM 16 GB, bandwidth 30 Mbps		65
	Core 8, RAM 32 GB, bandwidth 40 Mbps		248
Aliyun	Core 1, RAM 1 GB, bandwidth 10 Mbps	Intel Xeon(R) E5-2650 v2 2.60 GHz	29
	Core 4, RAM 8 GB, bandwidth 20 Mbps		89
	Core 8, RAM 16 GB, bandwidth 50 Mbps		235

Table 5 Job proportions of different job sets

Average data size	Small jobs (0, 16 GB]	Medium jobs (16, 64 GB]	Large jobs (64, 128 GB]
32 GB	48%	6%	46%
80 GB	7%	23%	70%

Table 6 Experimental parameters

Parameters	Definitions	Values
λ	The average job arrival rate	[30, 480]
N_q	The number of job queues	3
AS	The average data size of the job set	[32, 80]
DS	The data size of the job	(0, 128 GB]
N_{ct}	The number of concurrent tasks	(32, 512]
G	The population size of genetic algorithm	100
I	The number of jobs	1000
$Datasize(Task)$	The total data size of tasks	[2, 64 GB]
$N(Map)$	The number of map tasks	[16, 500]
$N(Reduce)$	The number of reduce tasks	1
D_i	The deadline of jobs	[40, 310]

5.1.3 Experiment test parameters

In our experiments, the private cloud contains three data centers. Each data center consists of 1 NameNode and 10 DataNodes. The selections of the public cloud contain Microsoft Azure, Amazon EC2 and Aliyun with different costs. All required experimental parameters are listed in Table 6.

5.1.4 Performance metrics

In order to evaluate the performance of our proposed load balancing job scheduling algorithm in the private cloud, evaluation metrics consist of the average job waiting time, average job response time and system throughput. The job waiting time is the interval from the time that a job arrives to the time that it begins to be executed. The job response time is the interval from the time that a job arrives to the time that its execution finishes. The throughput of the whole system is the number of jobs processed by the system per unit time.

In the verification experiments of the optimal task scheduling algorithm in hybrid clouds, evaluation metrics include the average task waiting time, average task response time and total monetary costs. The task waiting time is the interval from the time that a task arrives to the time that it begins to be executed. The task response time is the interval from the time that a task arrives to the time that its execution finishes. The total monetary costs denote the renting expenses of the public cloud.

5.2 Comparison and analysis

5.2.1 Performance evaluation of load balancing job scheduling algorithm in the private cloud

In this section. We evaluate the performance of our proposed load balancing job scheduling algorithm in the private cloud (*LBJS*) and three benchmark algorithms, *SRPT* algorithm [23], *SWAG* algorithm [24] and round robin algorithm (*RR*), in terms of the average waiting time, average response time and system throughput. Each experiment has been executed for 12 times and average values of performance metrics have been calculated to ensure the validity of the experimental results. How the average waiting time, average response time and system throughput are affected by various parameters is analyzed by varying different values.

5.2.1.1 Impacts of different job arrival rates and average data sizes In this sub-part, the number of concurrent tasks is 128. How the job arrival rate affects the average waiting time, job average response time and system throughput is investigated by varying its value from 30 to 480.

As shown in Fig. 4a, b the average job waiting time acts as an increasing function of the job arrival rate among all algorithms. This is because the queue backlog is bigger when more jobs arrive at the system. As shown in Fig. 4a, our proposed *LBJS* algorithm compared with *SRPT* algorithm, *SWAG* algorithm and *RR* algorithm can reduce the average job waiting time up to 26.29, 16.63 and 45.09%, respectively. In Fig. 4b, our proposed *LBJS* algorithm

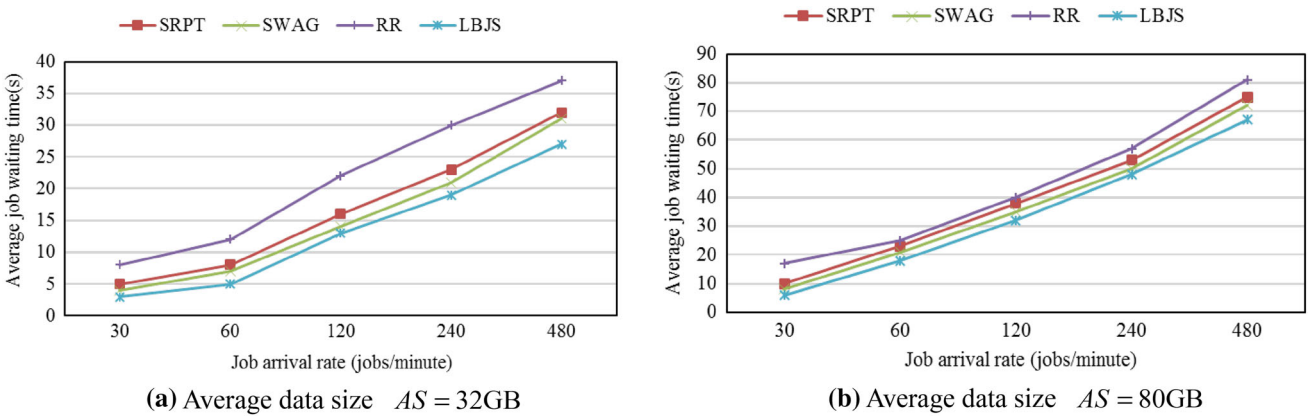


Fig. 4 Impacts of different job arrival rates on the average waiting time

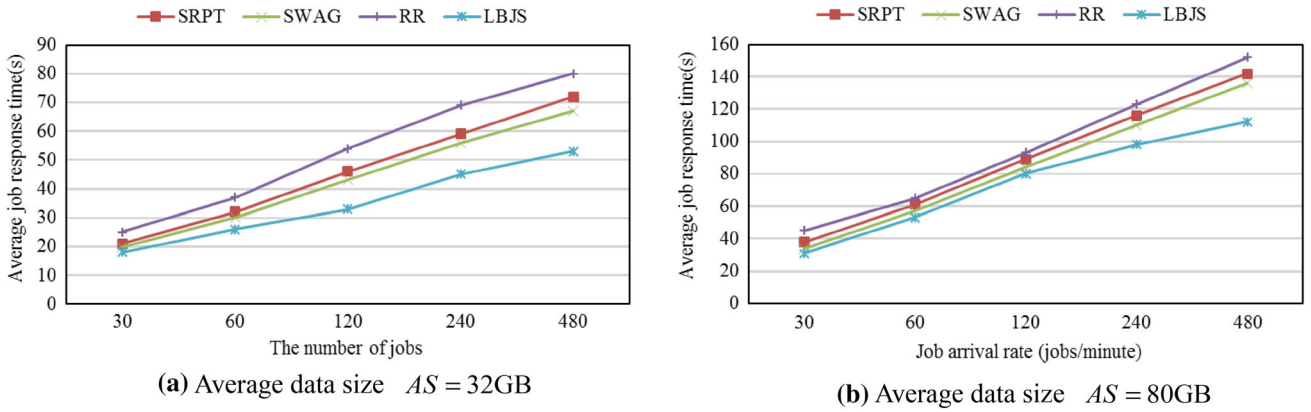


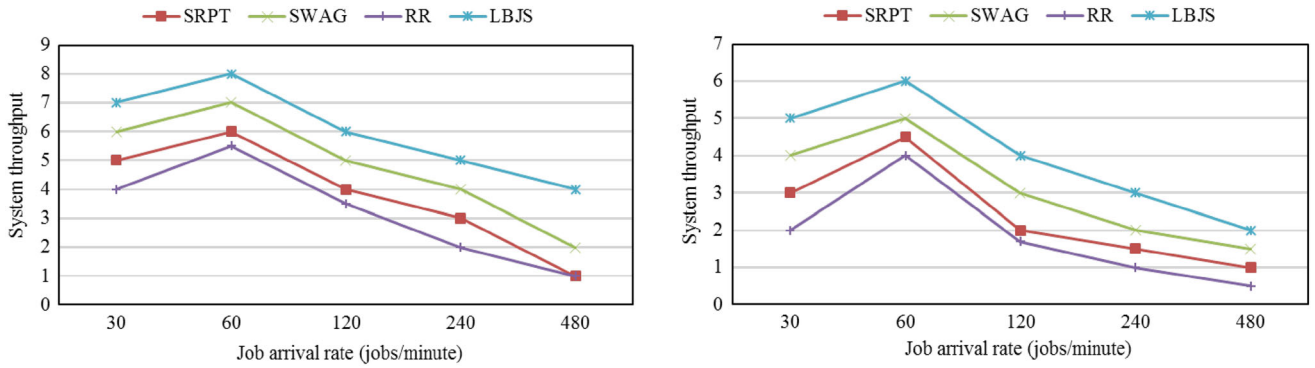
Fig. 5 Impacts of different job arrival rates on average response time

compared with *SRPT* algorithm, *SWAG* algorithm and *RR* algorithm can reduce the average job waiting time up to 19.53, 11.76 and 29.16%, respectively. The reason is that our proposed algorithm can choose optimal job queues for jobs according to queue states in the private cloud.

As observed from Fig. 5a, b, the average response time increases when the job arrival rate increases. As shown in Fig. 5a, our proposed *LBJS* algorithm compared with *SRPT* algorithm, *SWAG* algorithm and *RR* algorithm can reduce the average response time up to 22.28, 17.43 and 33.03%, respectively. In Fig. 5b, our proposed *LBJS* algorithm compared with *SRPT* algorithm, *SWAG* algorithm and *RR* algorithm can reduce the average job response time up to 15.66, 9.83 and 21.91%, respectively. The reason is that our proposed algorithm can dispatch jobs efficiently according to different average service rates of queues in the private cloud.

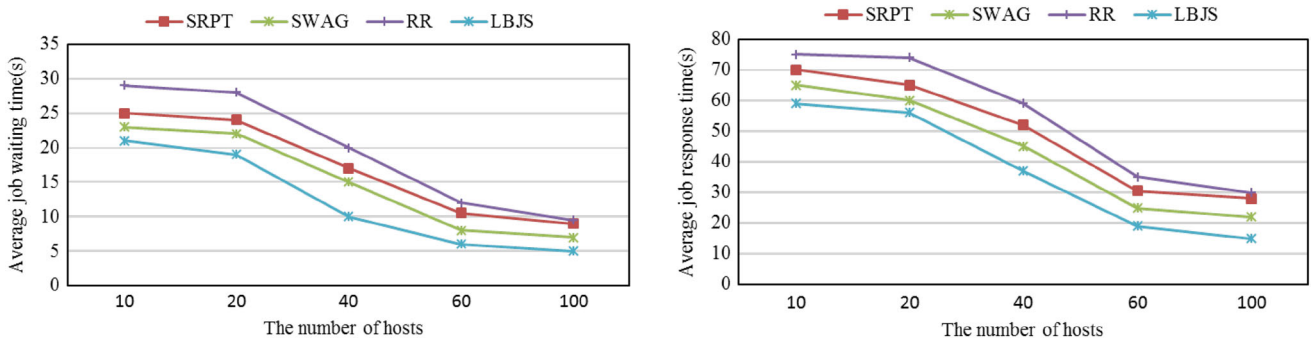
As observed from Fig. 6a, b, the system throughput decreases after an initial increase among all algorithms when the job arrival rate increases. As shown in Fig. 6a, our proposed *LBJS* algorithm compared with *SRPT* algorithm, *SWAG* algorithm and *RR* algorithm can improve the system throughput up to 33.33, 14.29 and 45.46% when the job arrival rate is 60 jobs/min. In Fig. 6b, our proposed *LBJS* algorithm compared with *SRPT* algorithm, *SWAG* algorithm and *RR* algorithm can improve the system throughput up to 31.36, 20.63 and 51.21% when the job arrival rate is 60 jobs/min. The reason is that our proposed algorithm can make full use of system resources to improve the throughput in the private cloud.

5.2.1.2 System scalability In this part, we test the scalability of algorithms. For the strong scaling tests, the average data size of job sets is 80 GB and the average job arrival rate is set to 60 jobs/min. How the number of hosts affects average job waiting time, average job response time and system throughput is investigated by varying its value from 10 to 100. As shown in Fig. 7a, b, both average job waiting time and average job response time of our proposed *LBJS* algorithm decline rapidly when the number of hosts increases from 20 to 60. When the number of hosts is 60, our proposed *LBJS* algorithm compared with *SRPT*

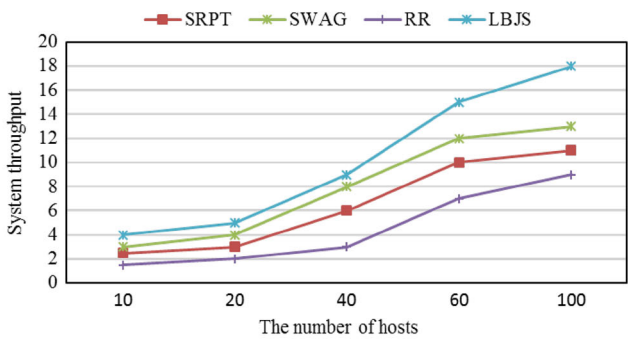


(a) Average data size $AS = 32\text{GB}$ **(b)** Average data size $AS = 80\text{GB}$

Fig. 6 Impacts of different job arrival rates on system throughput



(a) Average job waiting time **(b)** Average job response time

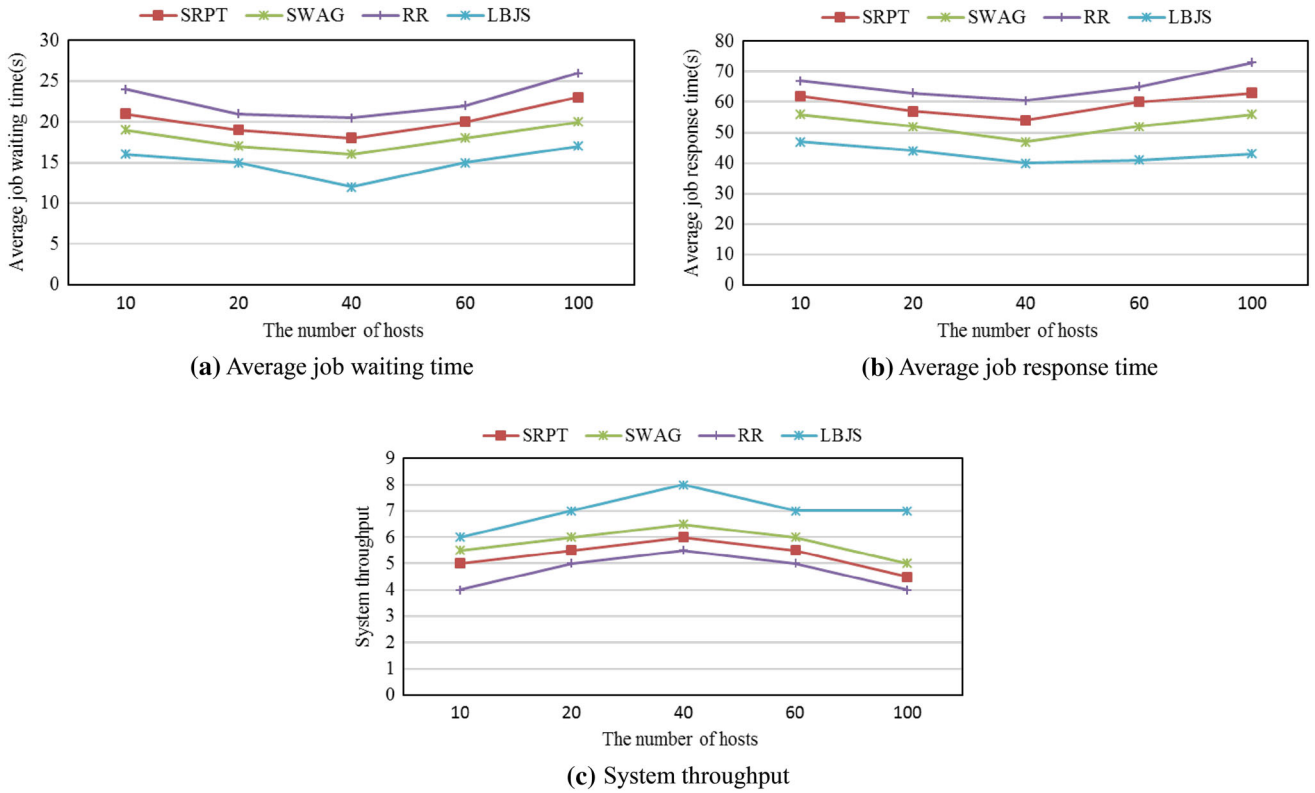


(c) System throughput

Fig. 7 Strong scaling results

algorithm, *SWAG* algorithm and *RR* algorithm can reduce the average job response time up to 37.7, 24 and 45.72%, respectively. As observed from Fig. 7c, the system throughput acts as an increasing function of the number of hosts. Compared with *SRPT* algorithm and *SWAG* algorithm, our *LBJS* algorithm can improve the system throughput up to 63.64 and 38.46% when the number of hosts is 100.

For the weak scaling results, we increase the number of hosts and proportionally the average job arrival rate. The ratio of the job arrival rate to the number of hosts is set to 2/21.1. The average data size of job sets is 80 GB. As shown in Fig. 8a, the average job waiting time increases after an initial decrease. In Fig. 8b, when the number of hosts is 100, our proposed *LBJS* algorithm compared with *SRPT* algorithm, *SWAG* algorithm and *RR* algorithm can reduce the average job response time up to 31.75, 23.22 and 41.1%, respectively. Figure 8c shows that our *LBJS* algorithm leads to a 45.46% over *RR* algorithm in terms of system throughput when the number of hosts is 40.



(a) Average job waiting time

(b) Average job response time

(c) System throughput

Fig. 8 Weak scaling results

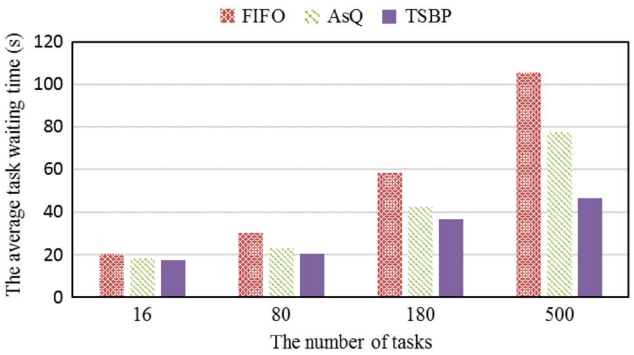
5.2.2 Performance evaluation of the optimal task scheduling algorithm in hybrid clouds

In this section, the performance of our proposed task scheduling algorithm based on BP neural network in hybrid clouds (*TSBP*) is compared with FIFO scheduling algorithm and *AsQ* algorithm [25]. Many experiments validate the effectiveness of our *TSBP* algorithm. Each experiment has been executed for 20 times and average values of the performance metrics have been calculated to ensure the validity of experimental results. How a variety of parameters impact the effectiveness of the task scheduling algorithms in hybrid clouds is investigated by varying different settings in each experimental group.

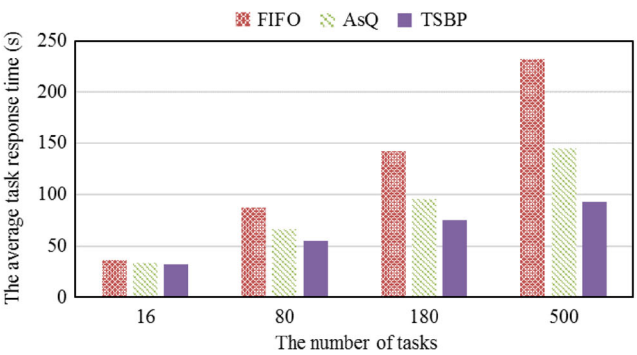
5.2.2.1 Impacts of different numbers of tasks The settings in this subsection are as follows: the number of tasks varies from 16 to 500 where the size of each task is 128 MB, and the deadline of each job is infinite. As observed from Fig. 9, both average task waiting time and average task response time act as increasing functions of the number of tasks. Our *TSBP* algorithm performs better than FIFO scheduling algorithm and *AsQ* algorithm when the number of tasks is large enough. The reason is that more computation resources are required to execute more tasks. Figure 9a shows that, our proposed *TSBP* algorithm compared with FIFO scheduling algorithm and *AsQ* algorithm reduces the average task waiting time up to 35.05 and 17.99%, respectively. Figure 9b represents that our *TSBP* algorithm leads to a 60.13% reduction over FIFO scheduling algorithm and a 35.81% reduction over *AsQ* algorithm in terms of the average response time when the number of tasks is 500.

Then, how the number of tasks affects the monetary cost is investigated. The deadline constraint is set to 60 s. As shown in Fig. 10, the costs of both *TSBP* algorithm and *AsQ* algorithm are 0 when the number of tasks is 16. All tasks can be processed in private cloud within the deadline constraint. When the number of tasks is 500, *TSB* algorithm compared with FIFO scheduling algorithm and *AsQ* algorithm reduces the cost up to 78.96 and 43.43%, respectively. Our *TSBP* algorithm can save more monetary cost under the same deadline constraint.

5.2.2.2 Impacts of different deadlines Finally, Fig. 11 shows how the deadline constraint affects the monetary cost by varying the value from 40 to 220 s. The number of tasks is 180. The cost shows as a decreasing function of different deadlines among all algorithms. When the deadline constraint is set to 140 s, the cost of our proposed *TSBP* algorithm is 0. In order to obtain the same results, the deadline constraint of AsQ algorithm should be set to 180 s



(a) The average task waiting time



(b) The average task response time

Fig. 9 Impacts of different numbers of tasks

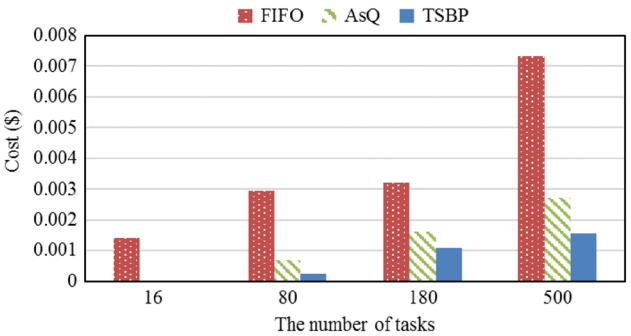


Fig. 10 Impacts of different numbers of tasks on the cost

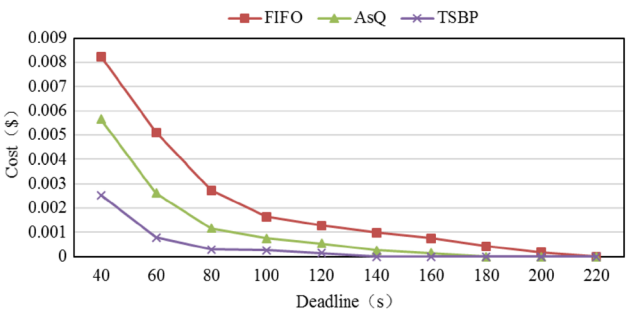


Fig. 11 Impacts of different deadlines on the cost

at least. Since our *TSBP* algorithm can make full use of the private cloud, the cost of *TSBP* algorithm is the cheapest under the same deadline constraint.

5.3 Experiment summary

In this paper, our proposed *LBJS* algorithm is compared with *SRPT* algorithm, *SWAG* algorithm and *RR* algorithm in terms of the average job waiting time, average job response time and system throughput. Many experiments show that our proposed *LBJS* algorithm can reduce the average job waiting time and average job response time. In additional, our *LBJS* algorithm can improve the system throughput of the private cloud.

Our proposed *TSBP* algorithm is compared with FIFO scheduling algorithm and *AsQ* algorithm in terms of the average waiting time, average response time and the monetary cost. Experiments show that our proposed *TSBP* algorithm can reduce the average task waiting time and average task response time significantly. Moreover, our *TSB* algorithm can reduce total monetary costs of hybrid clouds.

6 Conclusion

In this paper, a cost-aware job scheduling approach based on queueing theory in hybrid clouds is proposed. In the private cloud, the job scheduling problem in the private cloud is modeled as a queueing model. A GA is applied to achieve optimal queues for jobs to improve the utilization rate of the private cloud. Then, the task execution time is predicted by BP neural network. The max–min strategy is given to tackle the scheduling problem according to the prediction results in hybrid clouds. Experiments show that our cost-aware job scheduling algorithm can reduce the average job waiting time and average job response time in the private cloud. In additional, our proposed job scheduling algorithm can improve the system throughput of the private cloud. It also can reduce the average task waiting time, average task response time and total costs in hybrid clouds.

Acknowledgements The authors thank the editors and the anonymous reviewers for their helpful comments and suggestions. The work was supported by the National Natural Science Foundation (NSF) under Grants (Nos. 61672397, 61873341, 61472294), Application Foundation Frontier Project of WuHan (No. 2018010401011290), the Fundamental Research Funds for the Central Universities (WUT No. 2017-YB-029). Jiangsu Key Laboratory of Meteorological Observation and Information Processing, Nanjing University of Information Science and Technology (No. KDXS1804). Any opinions, findings, and conclusions are those of the author and do not necessarily reflect the views of the above agencies.

References

1. Ro, C.: Modeling and analysis of memory virtualization in cloud computing. Clust. Comput. **18**(1), 177–185 (2015)
2. Kaur, T., Chana, I.: Energy aware scheduling of deadline-constrained tasks in cloud computing. Clust. Comput. **19**(2), 679–698 (2016)
3. Kailasam, S., Gnanasambandam, N., Dharanipragada, J., et al.: Optimizing ordered throughput using autonomic cloud bursting schedulers. IEEE Trans. Softw. Eng. **39**(11), 1564–1581 (2013)
4. Guo, T., Sharma, U., Shenoy, P., et al.: Cost-aware cloud bursting for enterprise applications. ACM Trans. Internet Technol. **13**(3), 1–24 (2014)
5. Chopra, N., Singh, S.: Deadline and cost based workflow scheduling in hybrid cloud. In: 2013 2nd International Conference on Advances in Computing, Communications and Informatics, pp. 840–846. IEEE (2013)
6. TaoBao website. https://www.taobao.com/
7. Amazon website. https://www.amazon.com/
8. Toosi, A.N., Sinnott, R.O., Buyya, R.: Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka. Future Gener. Comput. Syst. **79**, 765–775 (2018)
9. Loreti, D., Ciampolini, A.: A hybrid cloud infrastructure for big data applications. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications, pp. 1713–1718. IEEE (2015)
10. Acs, S., Kozlovszky, M., Kacsuk, P.: A novel cloud bursting technique. In: 2014 9th IEEE International Symposium on Applied Computational Intelligence and Informatics, pp. 135–138. IEEE (2014)
11. Farahabady, M.R.H., Lee, Y.C., Zomaya, A.Y.: Pareto-optimal cloud bursting. IEEE Trans. Parallel Distrib. Syst. **25**(10), 2670–2682 (2014)
12. Farokhi, S., Jamshidi, P., Lakew, E.B., et al.: A hybrid cloud controller for vertical memory elasticity: a control-theoretic approach. Future Gener. Comput. Syst. **65**, 57–72 (2016)
13. Clemente-Castelló, F.J., Nicolae, B., Rafique, M.M., et al.: Evaluation of data locality strategies for hybrid cloud bursting of iterative MapReduce. In: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 181–185. IEEE (2017)
14. Lee, Y.C., Lian, B.: Cloud bursting scheduler for cost efficiency. In: 2017 IEEE 10th IEEE International Conference on Cloud Computing, pp. 774–777. IEEE (2017)
15. Charrada, F.B., Tata, S.: An efficient algorithm for the bursting of service-based applications in hybrid Clouds. IEEE Trans. Serv. Comput. **9**(3), 357–367 (2016)
16. Zhang, Y., Sun, J., Wu, Z.: An heuristic for Bag-of-Tasks scheduling problems with resource demands and budget constraints to minimize makespan on hybrid clouds. In: 2017 5th International Conference on Advanced Cloud and Big Data, pp. 39–44. IEEE (2017)
17. Daniel, D., Raviraj, P.: Distributed hybrid cloud for profit driven content provisioning using user requirements and content popularity. Clust. Comput. **20**(1), 525–538 (2017)
18. Li, C., Tang, J., Luo, Y.: Distributed QoS-aware scheduling optimization for resource-intensive mobile application in hybrid cloud. Clust. Comput. (2017). https://doi.org/10.1007/s10586-017-1171-2
19. Zhu, J., Li, X., Ruiz, R., et al.: Scheduling stochastic multi-stage jobs to elastic hybrid cloud resources. IEEE Trans. Parallel Distrib. Syst. (2018). https://doi.org/10.1109/TPDS.2018.2793254
20. Zuo, L., Shu, L., Dong, S., et al.: A multi-objective hybrid cloud resource scheduling method based on deadline and cost constraints. IEEE Access **5**, 22067–22080 (2017)
21. Champati, J.P., Liang, B.: One-restart algorithm for scheduling and offloading in a hybrid cloud. In: 2015 23rd IEEE International Symposium on Quality of Service, pp. 31–40. IEEE (2015)
22. http://snap.stanford.edu/data/index.html
23. Zhang, S., Pan, L., Liu, S., et al.: Profit based two-step job scheduling in clouds. Lect. Notes Comput. Sci. **9659**, 481–492 (2016)
24. Hung, C.C., Golubchik, L., Yu, M.: Scheduling jobs across geo-distributed datacenters. In: 2015 6th ACM Symposium on Cloud Computing, pp. 111–124. ACM (2015)
25. 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)

Chunlin Li is a Professor of Computer Science in Wuhan University of Technology. She received the ME in Computer Science from Wuhan Transportation University in 2000, and PhD in Computer Software and Theory from Huazhong University of Science and Technology in 2003. Her research interests include cloud computing and distributed computing.

Jianhang Tang received his BS Degree in Applied Mathematics from South-Central University for Nationalities in 2013 and MS Degree in Applied Statistics from Lanzhou University in 2015. He is a PhD Student in School of Computer Science and Technology from Wuhan University of Technology. His research interests include cloud computing and big data.

Youlong Luo is a Vice Professor of Management at Wuhan University of Technology. He received his MS in Telecommunication and System from Wuhan University of Technology in 2003 and his PhD in Finance from Wuhan University of Technology in 2012. His research interests include cloud computing and electronic commerce.