Available online at www.ijpe-online.com

**International Journal of Performability Engineering**

vol. 15, no. 6, June 2019, pp. 1580-1590
DOI: 10.23940/ijpe.19.06.p9.15801590

# Adaptive Job-Scheduling Algorithm based on Queuing Theory in a Hybrid Cloud Environment

Yanpei Liu[a,*], Xiaoni Chen[a], Ying Hu[a], and Qiang Cai[b]

[a]*School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou, 450002, China*
[b]*Beijing Key Laboratory of Big Data Technology for Food Safety, Beijing Technology and Business University, Beijing, 102488, China*

**Abstract**

To resolve the problem of unreasonable resource allocations caused by the continuous arrival of different types of jobs in a hybrid cloud environment, an adaptive job-scheduling algorithm based on queuing theory is proposed. This paper analyses job load types, and the jobs are classified according to the logistic regression method. A resource utility is used to classify the nodes in a private cloud cluster by considering the heterogeneity of the private cloud resources. Based on the job classification and the resource classification, a queuing model is established, and an adaptive genetic algorithm is used to manage the job queue's arrival rate that becomes the basis of the resource allocation. The proposed algorithm is compared with some existed similar algorithms to verify its performance in terms of job response times and throughput.

*Keywords*: hybrid cloud; heterogeneous resources; queuing theory; job-scheduling

(Submitted on March 20, 2019; Revised on April 3, 2019; Accepted on June 5, 2019)



## 1. Introduction

The volume of data used in large data applications is increasing explosively, and the data volumes produced in the fields of finance, weather forecasting, and data mining are as high as the PB level. Enterprises usually use a hybrid cloud model for large data processing projects. When dealing with data-intensive computations, a task is executed in the private cloud, when possible, to ensure the utilization of private cloud resources. Once private cloud resources are insufficient, public cloud resources are then allocated at minimum costs with constraints that allow for flexible expansion of resources. Therefore, research on adaptive online scheduling methods for different types of large data applications in heterogeneous hybrid cloud environments is of high theoretical value and has practical application significance.

However, the scheduling of large data applications in hybrid cloud environments still faces many challenges [1-3]: (1) The diversity of jobs submitted by users: The resource requirements and data volumes of large data services may vary in different time periods, and different users may submit different types of large data jobs. (2) The heterogeneity of hybrid cloud resources: Presently, enterprise IaaS private cloud platforms are heterogeneous not only in the physical machine but also in the design of the private cloud architecture; virtualization heterogeneity is common. Public cloud resources are more diversified, and the location of public cloud resources differs from the reputation value of public cloud service providers. (3) QoS heterogeneity: The various big data services have different QoS requirements.

At present, job-level scheduling in the hybrid cloud environment has produced noteworthy research results. The research in [4] proposed a fault-tolerant scheduling strategy in Hadoop, which optimizes completion times, maximizes resource utilization, and minimizes the task failure rate. In [5], a new Hadoop scheduling strategy, COSHH, was presented, and it considers heterogeneity at both the application and cluster levels. The algorithm classifies jobs according to job resource requirements. It then submits jobs to the queue based on resource execution efficiency as determined by the specific job resource requirements, thereby improving the average execution time, scheduling time, fairness, and locality. The authors of [6] considered the limited number of I/O ports on the client side and proposed a scheduling strategy based on

* Corresponding author.
*E-mail address*: liuyanpei@zzuli.edu.cn

the limited number of I/O ports to solve the problem of I/O traffic congestion and shorten the job response time. The authors of [7] proposed a data-locality-based scheduling method, which estimates the completion time of a job for an unexecuted job in a cluster. It then compares the estimated time with the time required to transfer resource blocks to find candidate nodes for job-scheduling and ensure data locality and reduce job execution time. The authors of [8] proposed a Hadoop scheduling algorithm for dynamic resource redistribution based on a fair scheduler. This algorithm allows jobs to use resources from other resource pools and improves the job execution efficiency and Hadoop scheduling.

The above algorithms have job-level scheduling that has been studied in depth, and the job-level scheduling performances have been improved to a certain extent. However, due to the development of large data applications and the promotion of the hybrid cloud model, most job-scheduling algorithms seldom employ job type matching with the corresponding resource types, affecting job response times and cluster job throughput. For this reason, this paper classifies user-submitted jobs as either IO-intensive jobs or CPU-intensive jobs according to their load characteristics and classifies cluster resources as either strong IO resource pools or strong CPU resource pools. Then, it establishes a job queuing structure according to the job types and resource types. This paper applies the job arrival probability model from queuing theory to select job queues and uses the improved adaptive genetic algorithm to calculate the job queue arrival rate. Finally, an adaptive job-scheduling algorithm based on queuing theory is proposed to decrease the average job response time and increase the throughput of the private cloud cluster.

The rest of this paper is structured as follows. Section 2 introduces the method of job classification. Section 3 presents a job queuing model based on job types and heterogeneous resources. Section 4 presents the implementation of a job adaptive scheduling algorithm based on queuing theory. Section 5 experimentally verifies the feasibility of the proposed algorithm. Section 6 gives the paper's conclusions.

## 2. Job Classification based on Logistic Regression

In this paper, jobs submitted to a hybrid cloud are classified, and jobs with different load types are scheduled corresponding to resource pools to reduce job response times and achieve load balancing of cluster nodes. According to literature [9], Map tasks are divided into IO intensive tasks and CPU intensive tasks according to the workload, namely, $\rho = MOD / MID$. $MTCT$ represents the completion time of Map tasks. $DIOR$ expresses the I/O rate of the disk, and $n$ denotes the number of tasks. IO-intensive tasks and CPU-intensive tasks are defined as Equations (1) and (2).

$$\frac{n \times (MID + MOD)}{MTCT} = \frac{n \times (1 + \rho)}{MTCT} \geq DIOR \tag{1}$$

$$\frac{n \times (MID + MOD)}{MTCT} = \frac{n \times (1 + \rho)}{MTCT} < DIOR \tag{2}$$

In this paper, we need to classify jobs before they are executed and then schedule them to the appropriate resource pool according to the type of jobs. At this time, the execution times of Map and $\rho$ are unknown, so Equations (1) and (2) cannot be directly applied to classify jobs. According to the job history information, the principal component analysis (PCA) method can be used to analyze the two factors affecting the IO-intensive and CPU-intensive job classification, namely, the amount of job data ( *DataSize* ) and the computational complexity ( *Complexity* ). This paper applies logistic regression [10] to classify jobs and then uses Equations (1) and (2) to check the types of jobs. The formula for calculating the sigmoid function of logistic regression is given by Equations (3) and (4).

$$\sigma(z) = 1/(1 + e^{-z}) \tag{3}$$

$$z = \omega_0 Datasize + \omega_1 Complexity \tag{4}$$

The value of $h_w(x)$ represents the probability of $y=1$. Therefore, the probability of classification results for Categories 1 and 0 is determined by Equations (5) and (6).

$$P(y = 1|x, w) = h_w(x) \tag{5}$$

$$P(y = 0|x, w) = 1 - h_w(x) \tag{6}$$

If $M$ is the number of samples, the log-likelihood function is determined by Equation (7).

$$f(w) = \sum_{i=1}^{M} [y^{(i)} log \left(h_w\left(x^{(i)}\right)\right) + (1 - y(i)) log\ (1 - h_w\left(x^{(i)}\right))] \tag{7}$$

The main factors of the job classification are regression coefficients $w_0$ and $w_1$ of the classification function. In this paper, the gradient rise method is used to determine the optimal regression coefficient. If the gradient is marked as $\nabla$, then the gradient of the function $f(w)$ is expressed as Equation (8).

$$\nabla f(w) = \begin{pmatrix} \frac{\partial f(w)}{\partial w_0} \\ \frac{\partial f(w)}{\partial w_1} \end{pmatrix} \tag{8}$$

The moving step is $\alpha$, and the iteration formula of the gradient rise is Equation (9).

$$w \coloneqq w + \alpha \nabla f(w), \quad w = [w_0, w_1]^T \tag{9}$$

First, the regression coefficient for the historical job trainer is used to determine the corresponding job type for the sigmoid function value, and then the sigmoid function of the job to be classified is calculated to get the classification results.

## 3. Job Queuing Model based on Job Type and Heterogeneous Resources

### *3.1. Resource Classification based on Utility Ratio of Heterogeneous Resources*

Computational and storage resources in the private cloud are mostly heterogeneous resources. Physical factors affecting job execution efficiency in a heterogeneous private cloud mainly include disk IO performance, bandwidth, memory, and CPU speed. *MIPS* represents the CPU performance, and memory mainly includes capacity, frequency, and timing performance indicators. This paper primarily considers the memory capacity, which is expressed in *Memory*. The main performance evaluation index of the storage devices is *IOPS*. The bandwidth affects the job transmission rate.

The computing resources of Hadoop's DataNode node are represented by CPU performance, and its storage resources are represented by IO performance. This paper defines a node whose CPU performance is better than the IO performance; in these cases, the node is the resource node of the CPU of the main resource and is classified into the strong CPU resource pool. Conversely, it is classified into a strong IO resource pool. $S_{ir}$ represents the average service time of type *r* jobs in *i* resources. $r = \{IO, CPU\}$ are IO-intensive jobs and CPU-intensive jobs, and $r = \{IO, CPU\}$ expresses the IO time and CPU time of the jobs. $S_{IO,IO}$ represents the average service time of IO-intensive jobs in IO resources. It can be shown that the matrix of the average service time of *r* type jobs in *i* resources can be expressed as Equation (10).

$$[S_{ir}] = \begin{bmatrix} S_{IO,\,IO}, S_{IO,\,CPU} \\ S_{CPU,\,IO}, S_{CPU,\,CPU} \end{bmatrix} \tag{10}$$

$\gamma_{ik}$ represents the physical performance impact factor of resource node *k*, and its calculation formula is expressed as Equations (11) and (12).

$$\gamma_{IO,\,k} = \beta_1 \times IOPS_k \tag{11}$$

$$\gamma_{CPU,\,k} = \beta_2 \times MIPS_k \tag{12}$$

In Equations (11) and (12), $\beta_1$ and $\beta_2$ represent the IO performance weights and the CPU performance weights of resource node *k*, respectively.

Literature [11] proposed a utility calculation for the heterogeneous resources and chose the resources with minimum response times. This paper improves its method, calculates the IO and CPU resource performance utility ratio of nodes,

judges the resource type of the nodes, and classifies node resources. The resource utility ratio of compute node $k$ is expressed as $\ell_{IO,k}$ and $\ell_{CPU,k}$ , and the calculation formula is given by Equations (13) and (14).

$$\ell_{IO,\,k} = log\frac{S_{IO,\,IO} \times \gamma_{IO,\,k}}{S_{CPU,\,IO} \times \gamma_{CPU,\,k}} / log\frac{S_{IO,\,IO} \times S_{CPU,\,CPU}}{S_{IO,\,CPU} \times S_{CPU,\,IO}} \tag{13}$$

$$\ell_{CPU,\,k} = 1 - \ell_{IO,\,k} \tag{14}$$

The resource utility ratio of nodes is calculated, and the resources are classified according to the resource utility ratio.

### *3.2. The Selection Basis of a Job Queue based on a Queuing Model*

This paper assumes that users submit jobs independently and in accordance with a Poisson distribution, that is, the arrival time interval of jobs conforms to a negative exponential distribution. The function $f(t)$ is introduced to calculate the relationship between time and the number of job submissions. Assuming that the number of job submissions is $N(t)$ for time *t*, we get the following:

(1) $N(t) \geq 0$;

(2) $N(t)$ is a positive integer;

(3) $s < t$ , then $N(s) \leq N(t)$ and $N(t) - N(s)$ denotes the number of jobs submitted in $(s,t]$ time;

In the interval of time $t$ , the random process $\{N(t),\ t \geq 0\}$ is a counting process. The number $N(t)$ of job submissions in any time *t* obeys a Poisson distribution with parameter $\lambda t > 0,$ and the arrival time interval of jobs obeys a negative exponential distribution.

$$f\big(N(t)\big) = P\{N(t+t') - N(t') = k\} = e^{-\lambda t} \times \frac{(\lambda t)^k}{k!},\ \ k = 0, 1, 2, \cdots \tag{15}$$

In Equation (15), $k$ is the total number of job submissions in a given period of time, and $\lambda = N(\Delta t)/\Delta t$ denotes the number of job submissions per unit time, i.e., the job arrival rate. In this paper, the process of submitting multiple types of jobs to heterogeneous private cloud clusters and waiting for cluster resources to serve jobs is established as a mathematical model, that is, a queuing model. The simplified model is shown in Figure 1.

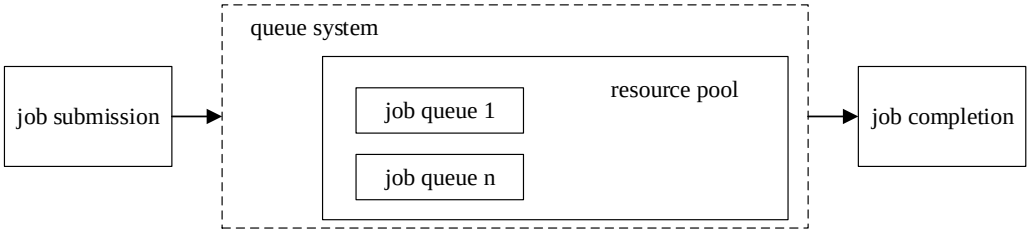


Figure 1. Simplified queuing model for job-scheduling

The three basic components of the queuing system are as follows:

(1) Job submission is also an input process in queuing theory. Jobs arrive in the system by a Poisson distribution process.

(2) After multi-type jobs enter different queues, they queue up according to certain rules and wait for cluster resources to serve them. This process conforms to the queuing rules of queuing theory.

(3) The number of job queues, service modes, and service times in heterogeneous resource pools are the core parts of queuing theory.

The average service time is defined as the average job execution time in this paper. Assuming the distribution function of the job execution time is $F(t)$, $u$ is the average service rate of job, that is, the average execution time of a job. Then, the average execution time of the job can be expressed as Equation (16).

$$1/u = \int tdF(t) \tag{16}$$

The job queue of a private cloud resource pool can run $Nes$ jobs at the same time. In a M/M/Nes/$\infty$ queuing system, the condition for system stability is shown by Equation (17).

$$\rho_j < \frac{\lambda_j}{u \times Nes_j} < 1 \tag{17}$$

In Equation (17), $\rho_j$ is the utilization rate of the heterogeneous resource pool system and $\lambda_j$ denotes the arrival rate for job queue *j*. To satisfy this condition, the stability probability of each M/M/c queue is expressed as Equation (18).

$$P_0 = \left( \sum_{k=0}^{Nes_j^{-1}} \frac{a_j^k}{k!} + \frac{a_j^{Vs_j}}{Nes_j!\,(1-\rho_j)} \right)^{-1}, \quad a_j = Nes_j \times \rho_j \tag{18}$$

$$e^z = \sum_{n=0}^{\infty} \frac{z^n}{n!}$$

and

$$\sum_{k=0}^{Nes_j^{-1}} \frac{a_j^k}{k!} \approx e^{a_j}$$

can be approximately deduced as Equation (19).

$$P_0 \approx \left( e^{a_j} + \frac{a_j^{Nes_j}}{Nes_j!\,(1-\rho_j)} \right)^{-1} \tag{19}$$

The probability of newly arrived jobs entering the private cloud resource pool is $P_j$. According to the Erlang formula, we can obtain Equation (20).

$$P_j = 1 - \sum_{k=0}^{Nes_j^{-1}} P_k = P_0 \left( \frac{a_j^{Nes_j}}{Nes_j!\,(1-\rho_j)} \right) \tag{20}$$

The formula for calculating the average number of jobs $Q_{avg}$ reaching the job queue *j* is expressed as Equation (21).

$$Q_{avg} = \frac{\rho_j}{1-\rho_j} P_j \tag{21}$$

The waiting time is $T_j^w$, the service time is $T_j^s$, and the response time of each job queue is $T_j^r$. The formula is shown as Equation (22).

$$T_j^r = T_j^w + T_j^s = \frac{Q_{avg}+Q_{init}}{\lambda_j} + \frac{1}{\mu} \tag{22}$$

In Equation (22), $Q_{init}$ represents the initial number of jobs in the queue, $N_{rp}$ is the number of jobs in the resource pool, and the formula for calculating the total response time of the resource pool is given by Equation (23).

$$T_{total}^r = \sum_{j=1}^{N_{rp}} P_j T_j^r = \sum_{j=1}^{N_{rp}} \frac{\lambda}{\lambda_j} (\frac{Q_{avg} + Q_{init}}{\lambda_j} + \frac{1}{\mu}) \tag{23}$$

### *3.3. Job Queue Selection based on Improved Adaptive Genetic Algorithm*

The basic genetic algorithm has the disadvantages of a slow convergence speed, tendency to collapse into a local optimal solution, and long running times. Therefore, this paper, based on the job queue load, improves the initial individuals of the genetic algorithm, reduces the search area, and decreases the learning time of the genetic algorithm.

The job queue load has four factors: the initial job number of the job queue, the resource utilization rate of the job queue, the job data volume, and the job computing complexity. At present, the Hadoop YARN resource allocation unit mainly includes the CPU virtual core number and the memory. The utilization rate of the $i^{th}$ queue resource is the sum of the CPU virtual core number and the memory utilization rate.

$$Ru_i = \delta_1 \frac{\sum_{i=1}^{k} availVCPU_i}{\sum_{i=1}^{k} VCPU_i} + \delta_2 \frac{\sum_{i=1}^{k} availMem_i}{\sum_{i=1}^{k} Mem_i} \tag{24}$$

In Equation (24), $availVCPU_i$ and $VCPU_i$ respectively represent the number of CPU virtual cores available and the total number of CPU virtual cores, while $availMem_i$ and $Mem_i$ respectively represent available memory and total memory. The resource allocation unit is $container = \{VCPU, Mem\}$. *Mem* is the memory size of a resource unit measured in GBs. *VCPU* is the core number of resource units. *Mem* is the memory size of a resource unit. We can then obtain Equation (25).

$$\delta_1/\delta_2 = VCPU/Mem \tag{25}$$

$\eta_1^{(k)}$, $\eta_2^{(k)}$, $\eta_3^{(k)}$, and $\eta_4^{(k)}$ represent the initial number of jobs, resource utilization, job data volume, and efficiency coefficient of the job computation complexity of the first job queue load. The formulas are as follows:

$$\eta_1^{(k)} = \frac{InitQ_{max} - InitQ_k}{InitQ_{max} - InitQ_{min} + 1} \tag{26}$$

$$\eta_2^{(k)} = \frac{Ru_{max} - Ru_k}{Ru_{max} - Ru_{min} + 1} \tag{27}$$

$$\eta_3^{(k)} = \frac{DataSize_{max} - DataSize_k}{DataSize_{max} - DataSize_{min} + 1} \tag{28}$$

$$\eta_4^{(k)} = \frac{Complexity_{max} - Complexity_k}{Complexity_{max} - Complexity_{min} + 1} \tag{29}$$

In Equation (26), $InitQ_{max}$ is the maximum initial job number in the job queue, $InitQ_{min}$ represents the minimum initial job number in the job queue, and $InitQ_k$ denotes the initial queue number in queue *k*. In Equation (27), $Ru_{max}$ is the maximum resource utilization of the job queue, $Ru_{min}$ denotes the minimum resource utilization of the job queue, and $Ru_k$ represents the resource utilization of the $k^{th}$ job queue. In Equation (28), $DataSize_{max}$ is the maximum data volume of the job queue, $Datasize_{min}$ denotes the minimum data volume of the job queue, and $DataSize_k$ represents the data volume of the $k^{th}$ job queue. In Equation (29), $Complexity_{max}$ is the maximum computational complexity of the job queue, $Complexity_{min}$ denotes the minimum computational complexity of the job queue, and $Complexity_k$ represents the computational complexity

of the $k^{th}$ job queue. The formula for calculating the total efficiency coefficient of the $k^{th}$ job queue is given by Equation (30).

$$\eta^{(k)} = \sqrt[4]{\eta_1{}^{(k)}\eta_2{}^{(k)}\eta_3{}^{(k)}\eta_4{}^{(k)}} \tag{30}$$

Therefore, the probability formula for job queue selection based on the initial queue load is expressed as Equation (31).

$$p_k = \eta^{(k)} / \sum_{k=1}^{n} \eta^{(k)} \tag{31}$$

The initial arrival rate of the job queue is described by Equation (32).

$$\lambda_k = \eta_k / \sum_{k=1}^{n} \eta_k \times \lambda \tag{32}$$

The greater the similarity between individuals, the greater the difference of fitness values between individuals. Therefore, the larger the crossover probability the smaller the mutation probability of the population. In contrast, the smaller the crossover probability, the larger the mutation probability of the population. According to this principle, the similarity coefficient is defined to reflect the differences between individuals. According to probability theory, the expected $EX$ is used to inverse the average fitness of the population, and the variance $DX$ is used to reflect the discrete degree of individual fitness.

$$EX = (f_1 + f_2 + \cdots + f_G)/G \tag{33}$$

$$DX = \sum_{i=1}^{G} (f_i - EX)^2 / G \tag{34}$$

In Equations (33) and (34), $G$ is the population size and $f_i$ denotes the fitness value of the $i^{th}$ individual, the calculation formula of which is given by Equation (35).

$$f_i = 1/T_{total}^{r} \tag{35}$$

The formula for calculating the similarity coefficient is shown as Equation (36).

$$factor = (EX + 1)/\sqrt{DX} \tag{36}$$

In this paper, the adaptive formulas of crossover probability and mutation probability are improved based on the logistic function.

$$p_c = 1/(1 + e^{-\frac{G_1}{factor}}) - G_2 \tag{37}$$

$$p_m = G_3/G_4(1 + e^{\frac{1}{factor}}) \tag{38}$$

In Equations (37) and (38), $G_1$, $G_2$, $G_3$, and $G_4$ are all constant.

## 4. An Adaptive Job-Scheduling Algorithm based on Queuing Theory

In this paper, the core pseudocode of the adaptive job-scheduling algorithm based on queuing theory is proposed as follows:

**Algorithm 1**: An adaptive job-scheduling algorithm based on queuing theory

Input：Job[1…k]

Output：HashMap<Job[1…k],Queue>

1: Cluster nodes are divided into strong IO resource pool and strong CPU resource pool.
2: Assume that the strong IO resource pool has $r$ job queues $QueueIO = \{QueueIO_1, \cdots, QueueIO_r\}$, the strong IO resource pool has $t$ job queues, $QueueCPU = \{QueueCPU_1, \cdots, QueueCPU_t\}$
3: Calculate the job arrival rate and probability of arrival for each job queue
4: **fo**r all jobs **do**
5: job class
6: **if** O intensive job
7: submit to the strong IO resource pool and select the job queue according to the arrival probability.
8: **else** CPU intensive job
9: submit to the strong CPU resource pool and select the job queue according to the arrival probability.
10: **end for**

## 5. Experiments

### *5.1. Experimental Environment*

The hybrid cloud environment in this experiment is composed of a private cloud and a public cloud. The private cloud is composed of six physical units, while different types of virtual machines of aliyun are rented as the public cloud experimental environment. The experiment in this paper is carried out on the open source Hadoop distributed platform. The operating system is based on Linux version Ubuntu 14.04, and the Hadoop version is hadoop-2.7.1. The hybrid cloud environment is shown in Figure 2.

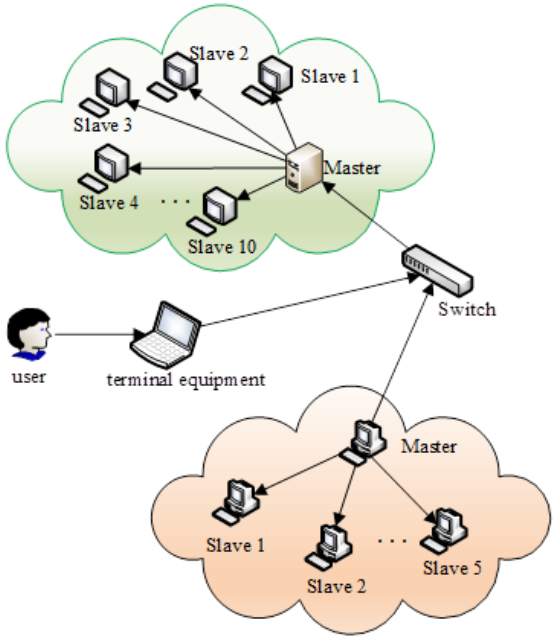


Figure 2. Architecture and network settings of hybrid cloud cluster

### *5.2. Comparison and Analysis of Algorithms*

To evaluate the average job response time and the cluster job throughput of the job adaptive scheduling algorithm based on queuing theory, the proposed algorithm is compared with FIFO, Fair, and COSHH [5].

#### *5.2.1. The Effect of Job Number on Algorithmic Performance*

The arrival rate of the job queue remains unchanged. After all jobs in the cluster are finished, different numbers of jobs are submitted to evaluate the impact of the job number on the performance of the algorithm. The types of jobs consist of six kinds of benchmark test programmes, and the data quantities are 2G, 10G, 23G, and 64G. There were random combinations of job types and data volumes, and the time interval of submitting jobs were subject to a negative exponential distribution.

In this paper, 20, 50, 100, and 180 jobs were submitted to the initialization cluster, and ten groups of FIFO, Fair, COSHH, and the proposed algorithm were run to obtain the average job response time and cluster throughput. The experimental results are shown in Figures 3 and 4.

From the experimental results of Figures 3 and 4, it can be seen that when the number of jobs is 25, the difference in the average job response time and the job throughput between COSHH and the algorithm proposed in this paper is very small. When the number of jobs is 150, the job response time of the proposed algorithm is 71.9%, 64.36%, and 51.09% lower than FIFO, Fair, and COSHH, respectively. At the same time, the throughput of this algorithm is 87.56%, 84.01%, and 41.38% higher than FIFO, Fair, and COSHH, respectively. When the number of jobs is small, the type of jobs and the amount of data are homogenous and not heterogeneous. The algorithm in this paper will be in the learning stage, and the resource utilization of each resource pool will be unbalanced. However, with an increase in the number of randomly submitted jobs, uniform job types, and queuing system stability, the performance of the proposed algorithm will be stable and better than that of the other three scheduling algorithms.

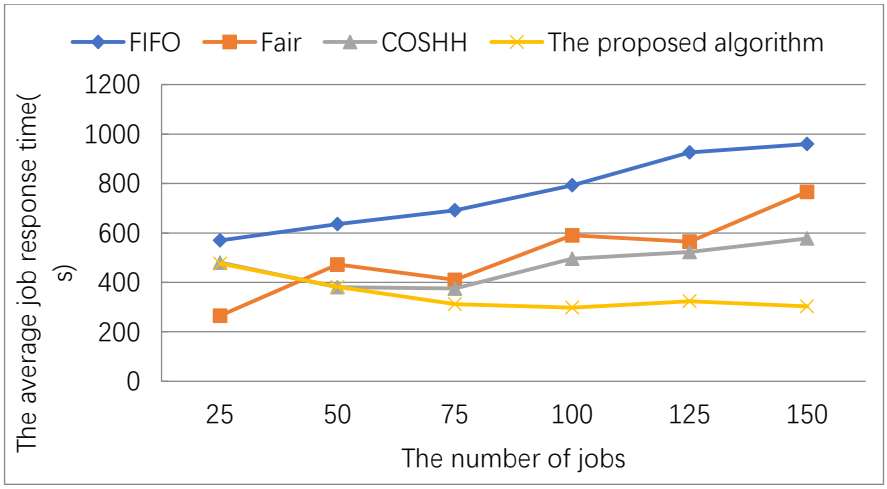


Figure 3. The effect of job number on response time

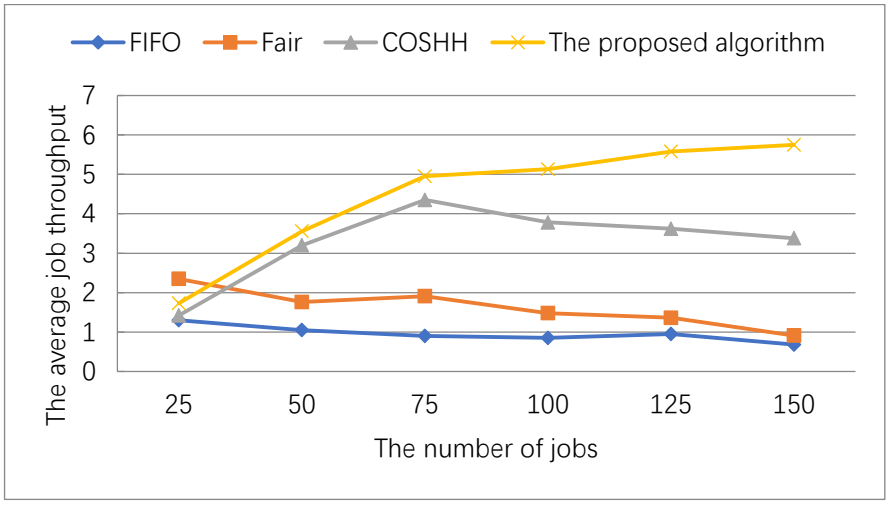


Figure 4. The effect of job number on throughput

*5.2.2. The Effect of Job Arrival Rate on Algorithmic Performance*

This paper sets different job arrival rates for the experiments and submits 150 jobs to run the algorithm proposed in this paper. The types of jobs consist of six kinds of benchmark test programmes, and the data quantities are 2G, 10G, 23G, and 64G. There were random combinations of job types and data volumes, and the time interval of submitting the jobs was subject to a negative exponential distribution. The same experimental method ran on ten groups of FIFO, Fair, and COSHH algorithms to obtain the average job response time and the cluster's job throughput. The experimental results are shown in Figures 5 and 6.

Figures 5 and 6 show that when the job arrival rate is 0.5, the average job response time of Fair is lower than FIFO, and the throughput is higher than FIFO. The performance of Fair and FIFO are better than that of COSHH and the proposed algorithm. With an increase in the job arrival rate, the performance of FIFO and Fair decreases. The performance of COSHH and the algorithm in this paper increases first and then decreases. When the job arrival rate is 2, the performance of the algorithm in this paper achieves the best performance.

## 6. Conclusions

In view of the job characteristics in the hybrid cloud environment and considering the types of large data jobs submitted by users and the diversity of the job requirements, the jobs submitted by users are divided into IO intensive jobs and CPU intensive jobs according to their load characteristics, and the cluster resources are classified into a strong IO resource pool and a strong CPU resource pool. The job queue structure is then established according to the job type and resource type. The job arrival probability model from queuing theory is used to select the job queues. The improved adaptive genetic algorithm is used to calculate the job queue arrival rate and determine the job queue arrival probability. An adaptive job-scheduling algorithm based on queuing theory for a hybrid cloud is proposed. Finally, the proposed job adaptive scheduling algorithm based on queuing theory is compared with FIFO, Fair, and COSHH. The experimental results show that the proposed algorithm performs well in terms of job response time and throughput.

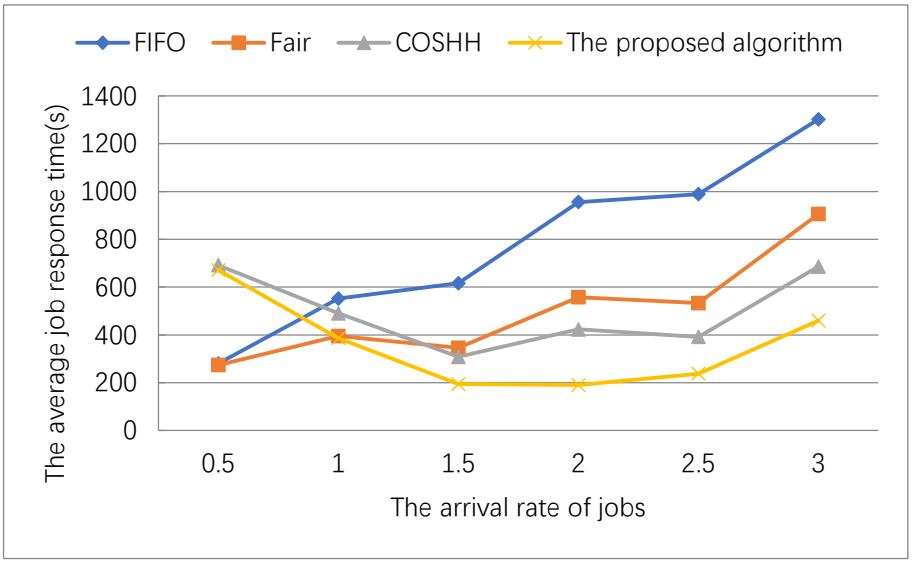


Figure 5. The effect of job arrival rate on response time

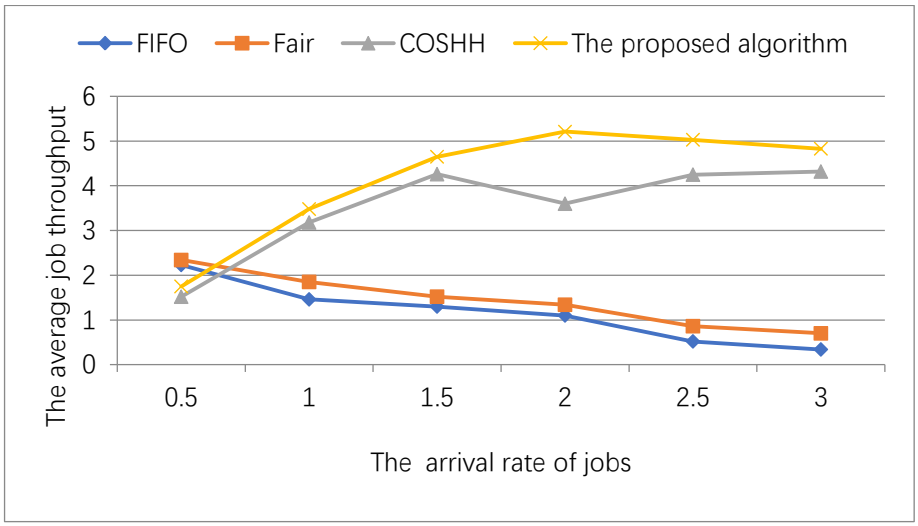


Figure 6. The effect of job arrival rate on throughput

**Acknowledgments**

The authors thank the editor and the anonymous reviewers for their helpful comments and suggestions. The work was supported by the National Natural Science Foundation (No. 61802353), Henan Provincial Department of Science and Technology (No. 192102210270), Open Issues of Beijing City Key Laboratory (No. BKBD-2017KF08), and Dr. Fund of Zhengzhou University of Light Industry.

**References**

1. J. G. Chen, K. L. Li, Z. Tang, and K. Bilal, "A Parallel Random Forest Algorithm for Big Data in a Spark Cloud Computing Environment," *IEEE Transactions on Parallel & Distributed Systems*, Vol. 28, No. 4, pp. 919-933, 2017
2. C. Napoli, G. Pappalardo, and E. Tramontana, "A Cloud-Distributed GPU Architecture for Pattern Identification in Segmented Detectors Big-Data Surveys," *Computer Journal*, Vol. 59, No. 3, pp. 338-352, 2018
3. C. W. Yang, Q. Y. Huang, and Z. L. Li, "Big Data and Cloud Computing: Innovation Opportunities and Challenges," *International Journal of Digital Earth*, Vol. 10, No. 1, pp. 13-53, 2017
4. Y. H. Moon and C. H. Youn, "Multihybrid Job Scheduling for Fault-Tolerant Distributed Computing in Policy-Constrained Resource Networks," *Computer Networks*, Vol. 82, No. 3, pp. 81-95, 2015
5. A. Rasooli and D. G. Down, "COSHH: A Classification and Optimization based Scheduler for Heterogeneous Hadoop Systems," *Future Generation Computer Systems*, Vol. 36, No. 3, pp. 1-15, 2014
6. W. Chongdarakul, P. Sophatsathit, and C. Lursinsap, "Efficient Task Scheduling based on Theoretical Scheduling Pattern Constrained on Single I/O Port Collision Avoidance," *Simulation Modelling Practice and Theory*, Vol. 56, No. 67, pp. 171-190, 2016
7. M. Khan, Y. Liu, and M. Li, "Data Locality in Hadoop Cluster Systems," in *Proceedings of 2014 International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 720-724, 2014
8. W. Tian, G. Luo, and L. Tian, "On Dynamic Job Ordering and Slot Configurations for Minimizing the Makespan of Multiple MapReduce Jobs," *IEEE Transactions on Service Computing*, Vol. 9, No. 1, pp. 1-6, 2016
9. C. T. Chen, L. J. Hung, and S. Y. Hsieh, "Heterogeneous Job Allocation Scheduler for Hadoop MapReduce using Dynamic Grouping Integrated Neighboring Search," *IEEE Transactions on Cloud Computing*, pp. 1-14, 2017
10. O. Komori, S. Eguchi, and S. Ikeda, "An Asymmetric Logistic Regression Model for Ecological Data," *Methods in Ecology & Evolution*, Vol. 7, No. 2, pp. 249-260, 2016
11. S. Spicuglia and L. Y. Chen, "On Load Balancing: A Mix-Aware Algorithm for Heterogeneous Systems," in *Proceedings of 2013 International Conference on PERFORMANCE Engineering*, pp. 71-76, 2013

**Yanpei Liu** is currently a teacher in the School of Computer and Communication Engineering at Zhengzhou University of Light Industry. She received her Ph.D. in computer science and technology from Wuhan University of Technology. She received her M.S. degree from Nanchang Hangkong University in 2009 and her B.S. degree from Luoyang Normal University in 2006. Her research interests include cloud computing and distributed computing.

**Xiaoni Chen** is currently a postgraduate student in the School of Computer and Communication Engineering at Zhengzhou University of Light Industry. She received her B.S. degree from Zhengzhou University of Light Industry in 2017. Her research interests include cloud computing and data mining.

**Ying Hu** is currently a teacher in the School of Computer and Communication Engineering at Zhengzhou University of Light Industry. She received her Ph.D. from Zhengzhou University in 2016. Her research interests include computer networks and distributed computing.

**Qiang Cai** is currently a professor in the School of Computer and Information Engineering at Beijing Technology and Business University. His research interests are web databases, distributed computing, and heterogeneous systems integration.