International Journal of Grid Distribution Computing
Vol. 8, No.6, (2015), pp.269-280
http://dx.doi.org/10.14257/ijgdc.2015.8.6.26

Task Scheduling Algorithm based-on QoS Constrains in Cloud Computing

Yi Zhang and Baomin Xu

School of Computer and Information Technology, Beijing Jiaotong University
Beijing 100044
12281208@bjtu.edu.cn, bmxu@bjtu.edu.cn

Abstract

Based on the study of traditional min-min scheduling algorithm, the paper proposed a min-min task scheduling algorithm based on QOS constraints in cloud computing. According to the vector which is generated by QOS parameters, the algorithm processes the matching of resources and tasks, and then provides users with resources which meet their requirements. Experimental results show that the min-min task scheduling algorithm for cloud computing has better performance in such aspects as task execution time, rate of discarding task and QOS satisfaction compared with traditional min-min scheduling algorithm.

Keywords: *Cloud computing; task scheduling; quality of service*

1. Introduction

In recent years, cloud computing is becoming an increasingly important research topic. The task scheduling problem is an important problem in cloud computing [1-3]. It is directly related to the stability of cloud services, efficient use of resources, the user's satisfaction degree and operating costs. Therefore, the study of task scheduling of cloud computing is of great significance.

The nature of task scheduling is to create a mapping relationship between tasks and resources. Therefore, a cloud task scheduling strategy is how to find a suitable mapping relationship between tasks and resources and to ensure task scheduling algorithm achieve various performance indicators base on the basic needs of users task. One of the goals of cloud task scheduling is the optimal makespan, which directly affects the interests and satisfaction of the users. Therefore, it is the common goal of users and providers of cloud services to realize the optimal makespan.

Task scheduling is a NP-complete problem. About the problem, scientists from all over the world have made a lot of research work recently. There are many classical task scheduling algorithms. Such as Hadoop default FIFO scheduling algorithm [4], it has a good performance, but it can't ensure small task be executed promptly. Against this fault, Facebook developed a fair scheduling algorithm which can ensure the small task get response quickly while have a high service level for a big task [5]. Yahoo developed a computing capabilities scheduling algorithm named capacity scheduler [6]. We have put forward a fair scheduling algorithm based on Bergel model [1]. According to QOS parameter matching, our algorithm can allocate a task with the best matching resources based on the fairness principle of resource allocation.

The goal of traditional scheduling algorithm is generally to complete task with the shortest time. However, what users expect and users' behavior are complex in a cloud environment. Sometimes the user is not just simply pursuing the shortest completion time.

Cloud computing involves users, providers of cloud services, and other entities. The goal of different user's QoS may be different among security policy, cost and other

ISSN: 2005-4262 IJGDC

aspects, even conflicting. For example, resource managers hope to achieve global control on resources, to maximize the throughput of whole system. However, the user usually wants to run his task using the whole resource. In addition, tasks come from different users could tend to have shorter execution time, or to emphasize safety and reliability during execution, or to pay great attention to their own expenses. Therefore, only consider the problem of optimizing the maximum completion time is obviously not enough. In one word, how to use resource management and scheduling algorithm to coordinate the QoS requirements of different entities is what current research focus on.

Each task may contain a lot requirements of QoS, such as timeliness, reliability, safety, priority and so on. If the system has a great compute ability, various QoS requirements of the task can be met as far as possible. Otherwise, even we can get ideal makespan, there are still many problems, such as some tasks can't be finished before the deadline, the task which has a high demand of reliability may be distributed to the compute node that has a high failure rate.

Under such circumstances, we need to establish a source manage model which is based on QoS. With this model, the submitted task should provide a description of QoS requirements, the resources also should have a description of the QoS service. During the schedule, scheduling algorithm should take a full consideration of the description of task and resources, and then perform the scheduling from task to resource.

2. Resource Manage Model based on QoS

In the resource manage model based on QoS, the task should give QoS requirements. Compute nodes in cloud computing center need give QoS description of computing resources that they could provide. The scheduling algorithm try to search the global benefit optimization scheduling scheme.

2.1. Description of QoS

In order to establish resource manage model based on QoS and process task scheduling of multi QoS restriction. We need to describe and classify the QoS parameters. About this problem, B.Sabata and his partners have done some research [7]. They divide QoS description into metrics and polices. Metrics is used to measure some QoS parameters, including QoS requirements of task and QoS service which provided by compute nodes. Metrics can further be divided into performance metrics, security, and relative importance (Priority). Polices is used to describe the behavior of the system specifications. The main polices include management policy, service level, etc. The performance metrics of QoS is further divided into timeliness, precision, accuracy and other aspects. They are used to indicate the requirements of specific task in performance. The paper will divide QoS description which is used as metrics into following cases:

(1)Timeliness description: Property description related to time. The timeliness of QoS description of task includes the total completion time, start time, the latest completion time, and so on. The timeliness of QoS description of resource refers to estimate the execution time of a task on the resource. We mainly consider the latest completion time and the completion time of a task.

(2)Reliability description: Long tasks may fail due to resource failure. To restart these tasks, it will cause repetitive resource consumption and lead to a decline in performance of the system. Scheduling these kinds task according to the requirements of reliability, it could reduce the happening of this kind of situation. The reliability description of the task includes the required minimum probability of successful completion. This kind description also includes its failure rate per unit time.

(3)Priority description: Used to describe the relative importance of task. The task priority shall be specified by the people. The higher priority tasks need to be executed earlier.

(4)Security description: Task and data submitted by each user has different requirements in terms of security level. The security here includes authenticity, confidentiality, and integrity. Data confidentiality and integrity are two important metrics of data security. The confidentiality of the data is to ensure that the data is not access by other users. The integrity of the data is to ensure that the data is accurate in the process of task scheduling. In the paper, we divide security into four levels: poor, low, medium, and high.

QoS description of a strategy is the primarily service level, it works on all kinds of metrics of QoS description we mentioned above. Specific including:

(1) Hard: The QoS description with this level has strongest restraint. The description of the task must be met, otherwise the scheduling is invalid.

(2) Soft: There are some QoS description doesn't need very strong constraint. Task will obtain the biggest benefit if such descriptions are met. Otherwise, the scheduling is not considered invalid, just benefit can be affected.

(3) Best-effort: This level is used to represent some QoS description which is not being concerned. The description of the task will be met as far as possible.

In this paper, the QoS description of task or resource is described as a function:

$$q:T \cup U \rightarrow \Omega$$

where Ω is the set of all QoS. We assume that each task t_i in task set T contains QoS parameter d_i. For the jth QoS parameter, Ω_i^j can be a finite or infinite set if Ω_i^j represents the value range of QoS parameter. We can further assume q_i^j represents a possible value of the jth QoS parameter of task t_i, $q_i^j \in \Omega_i^j$. Thus, we regard value of each QoS parameter of task t_i as one-dimension, so all values of parameter $\Omega_i = \{\Omega_i^1, \Omega_i^2, ..., \Omega_i^{d_i}\}$ can form a space of d_i dimension, and $q_i = \{q_i^1, q_i^2, ..., q_i^{d_i}\}$ is one point of the space.

In this paper, we consider the following four QoS parameters: timeliness, reliability, security, and priority. Namely d_i=4.

2.2. The Benefit Function

For each type of QoS description, we introduce a concept of benefit function in order to quantify the benefit of being met by different levels. Reasonable benefit function [8] is prerequisites of whether scheduling algorithm can produce optimal scheduling scheme.

For task t_i, we define benefit function of the jth （$1 \leq j \leq di$） QoS description as U_i^j. The corresponding benefit function of four kinds of QoS metrics presented in this paper respectively are:

1) Timeliness benefit function

Timeliness benefit function represents the time benefit that being brought to task owner when the task t_i is mapped to resource r_j, namely according to the finish time of the task on the resource.

For hard, soft and best-effort three kinds of service level, the timeliness benefit function is defined as $TimeUtility_{hard}(i,j)$, $TimeUtility_{soft}(i,j)$, $TimeUtility_{best\_effort}(i,j)$, respectively:

(1) Timeliness benefit function of hard level

$$TimeUtility_{hard}(i,j) = \begin{cases} 1 & CT_{i,j} \leq D_i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where D_i is the deadline of t_i, $CT_{i,j}$ is the expected completion time of task t_i on resource r_j. The meaning of this function is that we will obtain the biggest benefit 1 if

tasks are completed within the prescribed deadline, otherwise receiving none benefits. As shown in Figure 1.

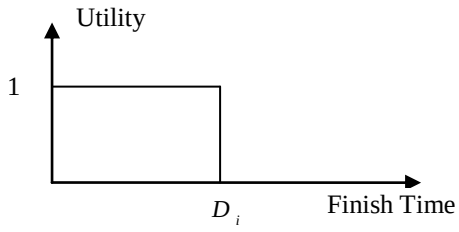

Figure 1. Hard Level Timeliness Benefit Function

(2) Timeliness benefit function of soft level

$$TimeUtilit \quad y_{soft}(i,j) = \begin{cases} 1 & CT_{i,j} \le D_i \\ \exp(-\lambda_s \cdot (CT_{i,j} - D_i)) & \text{otherwise} \end{cases} \tag{2}$$

where λ_s is a constant, and $\lambda_s>0$. The meaning of this function is that we will obtain the biggest benefit 1 if tasks are completed within the prescribed deadline. If the task completion time is exceed the deadline, it will also bring some benefits to the owner of the task. Here, when the task is not completed within the deadline, the benefits that the execution of the tasks bring to the task owner were expressed with exponential function. As shown in Figure 2, when $CT_{i,j}=D_i$, we get maximum benefit value 1; when $CT_{i,j}>D_i$, the benefit go down along with the exponential function decreasing, eventually close to zero and is not equal to zero.

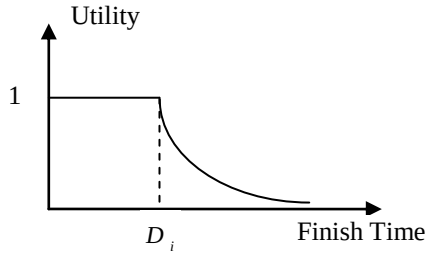

Figure 2. Soft Level Timeliness Benefit Function

(3) Best-effort Level Timeliness Benefit Function

$$TimeUtilit \quad y_{best\_effortt}(i,j) = \exp(-\lambda_b \cdot CT_{i,j}) \tag{3}$$

Where λ_b is a constant, and $\lambda_b>\lambda_s$.The meaning of this function is that if task t_i can be completed in a short time it will make much benefits; Even if the task t_i takes a long time to complete, it still will bring some benefit to task owner. Here also uses the exponential function to represent efficiency that is brought by task execution. The execution time is the independent variable of the exponential function. As shown in Fig.3. With the increase of the independent variables, efficiency go down along with the exponential function decreasing, eventually close to zero.

2) Reliability Benefit Function

Reliability benefit function represents the benefit that is brought to task owner in reliability when distribute task t_i to resource r_j. For hard, soft and best-effort three kinds of service level, the reliability benefit function is defined as RliUtilityhard(i,j), RliUtilitysoft(i,j), RliUtilitybest_effort(i,j), respectively.

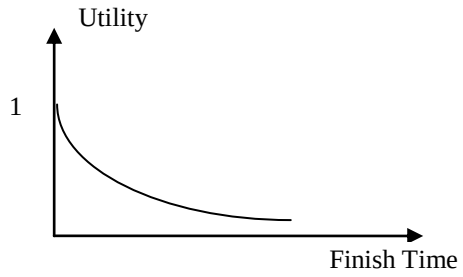

Figure 3. Best-Effort Level Timeliness Benefit Function

(1)Hard Level Reliability Benefit Function

$$RIiUtility_{hard}(i,j) = \begin{cases} 1 & RT_{i,j} \geq RD_i \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where RD_i is the least successful completion probability that task t_i required, $RT_{i,j}$ is the reliability that can be obtained when task t_i run on resource r_j. The meaning of this function is that we will get the maximum benefit 1 if the obtained reliability when task t_i run on resource r_j exceed the least successful completion probability that is fixed by the task itself; Otherwise there is no benefit. As shown in Figure 4.

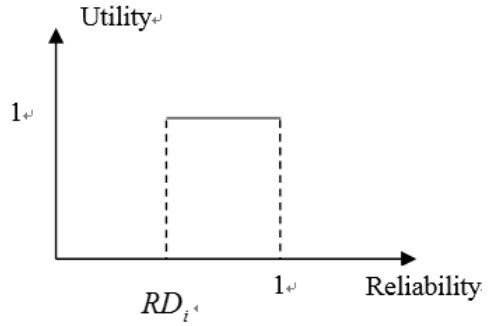

Figure 4. Hard level Reliability Benefit Function

For $RT_{i,j}$ in Equation (4), we can use the following method to calculate: assuming that the inherent failure rate of resource r_j is Fr_j. The completion time of task t_i on r_j is $CT_{i,j}$. The reliability when task t_i run on this resource can be calculated using the equation:

$$RT_{i,j} = \exp\{-CT_{i,j} \cdot Fr_j\} \quad (5)$$

(2)Soft Level Reliability Benefit Function

$$RIiUtility_{soft}(i,j) = \begin{cases} 1 & RT_{i,j} \geq RD_i \\ \dfrac{\exp(-(1-RD_i)) - \exp(-(RT_{i,j}+1-RD_i))}{\exp(-(1-RD_i)) - \exp(-1)} & \text{otherwise} \end{cases} \quad (6)$$

In Equation (6), when the value of $RT_{i,j}$ is greater than or equal to RD_i, it is shown that the reliability of the resources can satisfy the requirements of the task, the value of benefit function is 1; when the value of $RT_{i,j}$ is zero, the value of benefit function is 0. In other cases, the function shows exponential growth. As shown in Figure 5.

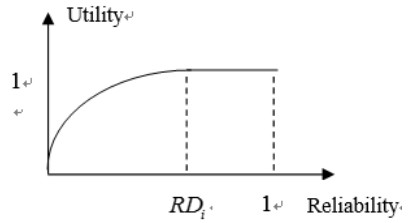

Figure 5. Soft Level Reliability Benefit Function

The meaning of this function is that we will get the maximum benefit 1 if the obtained reliability when task t_i run on resource r_j exceed the least successful completion probability that is fixed by the task itself. Otherwise there is no benefit, but we still can get a certain benefit.

(3) Best-effort level reliability benefit function

$$RliUtility_{best\_effort}(i,j) = \frac{1-\exp(-RT_{i,j})}{1-\exp(-1)} \tag{7}$$

In Equation (7), when the value of $RT_{i,j}$ is zero, the value of benefit function is zero. With the increase of the independent variables, function presents exponential growth. As shown in Figure 6. $RT_{i,j}$ eventually tend to be 1. So the value of benefit function also tend to be 1 in the end. The meaning of this function is that if the obtained reliability is high when task t_i run on resource r_j, we can get high efficiency-benefit value; Otherwise we will still get a small amount.

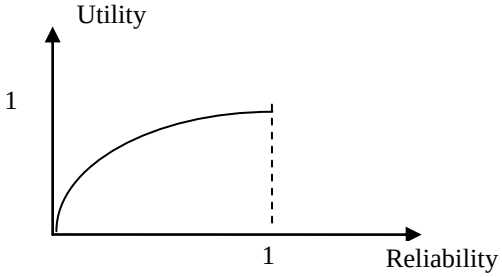

Figure 6. Best-effort Level Reliability Benefit Function

3) Security Benefit Function

Reliability benefit function represents the benefit that is brought to task owner in security when distribute task t_i to resource r_j.

Description of tasks or resources reliability is divided into four levels. To the submitted task and computing resources which provide services, they need to be given a certain level of security before the simulation of scheduling process. Assuming that the security level of task t_i and resource is ST_i and SR_j, respectively. We set the rule that if the security required by task is greater than the security provided by resource, the obtained benefit is zero when task run on this resource; otherwise the benefit is 1.The benefit function as follows:

$$SecUtility(i,j) = \begin{cases} 1 & SR_j \geq ST_i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

4) Relative Importance (Priority)

Priority is just in terms of the tasks. Each task will be set up a certain priority before simulating scheduling process, the parameter is written directly into the ultimate benefit function.

To sum up, we have established corresponding benefit function for the four kinds of QoS description. However, we can't compute the total benefit only according to benefit function of a certain kind of QoS description. This requires us to define an overall efficiency function, changing all QoS description of one task into a total benefit value which is used as the basis of our scheduling. The total benefit function ($U_{i,j}$:$\Omega_i \rightarrow \mathbb{R}$) of distributing task t_i to resource r_j as follows:

$$U_{i,j}(q_i) = \sum_{j=1}^{d_i} U_i^j(q_i^j) \cdot p_i \quad (9)$$

where p_i is the priority of task t_i.

3. Task Scheduling Algorithm based on QoS Constraints

3.1. Min-Min Algorithm

The idea of Min-Min algorithm is to map small tasks to compute node which perform fast. Min-Min algorithm is based on the MCT (Minimum Completion Time) algorithm. The Min-Min algorithm in every task mapping considers all tasks which are not allocated, however, MCT algorithm only considers a task in every task mapping.

Specifically, when task set U is not empty, the Min-Min algorithm repeatedly perform the following operations until the entire task set U is empty:

(1) For each task t_i in U, it need to calculate the different minimum completion time to assign the task to n machines. Assuming task t_i has the minimum completion time on the kth machine, the minimum time can be recorded as MinTime(i) =MCT(i,k). *MinTime* is a one dimensional array containing m elements, it represents a set which consists of the minimum completion time;

(2)Choose task which has the minimum completion time from MinTime array. Assuming the task is t_a, it need to assign the task to its corresponding host such as h_b;

(3)Delete t_a from U while need to be update MCT matrix. Namely update MCT of the rest tasks on host h_b.

(4) Repeat the above process until U is empty.

3.2. Mul-QoS-Min-Min Algorithm

In cloud environment, the task could contain a lots QoS requirements. In this case, we will combine the QoS requirements with traditional Min-Min algorithms to form a new task scheduling algorithm based on multi-QoS constraints, namely Mul-QoS-Min-Min algorithm. The pseudo code of the algorithm as follows.

```
Split task set T into three task set T1,T2,T3
Let k=1,flagi=0, 1≤i≤n.
repeat
    if (Tk≠Θ)
        for each host ti∈Tk
          for each host rj∈M
              if q(ti)<q(rj) then
     Compute Ui,j;
     flagi=1;
        else
```

$U_{i,j}$ =0;
endif
endfor
if($flag_i$=0)
Remove task t_i from T_k
Endif
Compute MaxValue[i]=max{ $U_{i,j}$ } and keeps the task-host pair(t_i, r_j)
endfor
Find the maximum of MaxValue and the corresponding task-host pair(t^*, r^*)
If exists two values MaxValue[i]= MaxValue[j]
Choosse the task-host pair(t^*, r^*) with smaller complete time
endif
Assign task t^* to host r^* and remove t^* from T_k
endif
if (T_k =Θ)
k= k+1
endif
until(k≤3)

4. Results and Discussion

The simulation experiments were conducted using CloudSim [9]. The hardware platform is Dell Studio XPS 8300, with a 3.00GHz core i5 and 4GB of memory.

In simulation experiments, according to the different number of tasks and resources, we will consider the following three scenarios:

(1) Task number is 20, resource number is 10;

(2) Task number is 40, resource number is 20;

(3) Task number is 50, resource number is 25;

This experiments will investigate the following three problems:

(1) Compare the makespan of traditional Min-Min algorithm and Mul-QoS-Min-Min algorithm;

(2) Investigate the hard task level QoS requirements satisfaction of the Min-Min algorithm and the Mul-QoS-Min-Min algorithm;

(3) Investigate the failure rate of Mul-QoS-Min-Min algorithm with the increasing number of tasks in the case that the amount of resource is 50.

(1) Experiment 1

The experiment will compare the makespan of the above two algorithms under three kinds of scenarios. In each scenario, experiment was conducted 20 times for each algorithm. The average makespan is shown in Table 1.

Table 1. The Average Makespan of Min-Min Algorithm and Mul-QoS-Min-Min Algorithm

	Min-Min	Mul-QoS-Min-Min	Performance improvements
(1)	48.30	44.83	7.2%
(2)	67.23	66.85	0.57%
(3)	97.95	95.62	2.38%

Seen from Table 1, the makespan of Mul-QoS-Min-Min algorithm which is put forward in this paper is better than the traditional Min-Min algorithm.

(2) Experiment 2

The experiment will investigate the satisfaction demand of hard level task QoS requirements of both Min-Min algorithm and Mul-QoS -Min-Min algorithm.

In each scenario, some experiments were conducted for each algorithm. Here, these tasks with regard to the timeliness of the hard level requirements as the research object. Namely, these experiments mainly investigate the satisfaction of these tasks demand for timeliness of the two algorithms, respectively. The results are shown in Table 2, Table 3, and Table 4.

Table 2. Task Scheduling on Hard Level Timeliness for Scenario 1

Task id	Deadline for the task	Min-Min	Mul-QoS-Min-Min algorithm
10	31.5238	44	27
16	19.9525	25	16

Table 3. Task Scheduling on Hard Level Timeliness for Scenario 2

Task id	Deadline for the task	Min-Min	Mul-QoS-Min-Min
0	60.3605	46	34
20	34.2928	39	27
23	43.5842	26	21
28	44.7331	48	17
34	62.3290	51	54
39	63.8019	38	26

Table 4. Task Scheduling on Hard Level Timeliness for Scenario 3

Task id	deadline for the task	Min-Min	Mul-QoS-Min-Min
16	12.0621	16	6
31	40.5121	34	20
47	37.9252	16	15

Seen from Table 2, Table 3, and Table 4, the satisfaction of Mul-QoS-Min-Min is 100% for hard level timeliness task, however the satisfaction of Min-Min is 0, 66.7%, and 66.7% at three different scenarios.

(3) Experiment 3

The experiment will investigate the failure rate of Mul-QoS-Min-Min and Min-Min. We consider the discarding task of Mul-QoS-Min-Min and Min-Min under a number of different resources and tasks. When the number of resources is 50 and the Task number change within 0-500, the results of experiment are shown in Figure 7. It can be seen that Mul-QoS-Min-Min has less failure rate than Min-Min.

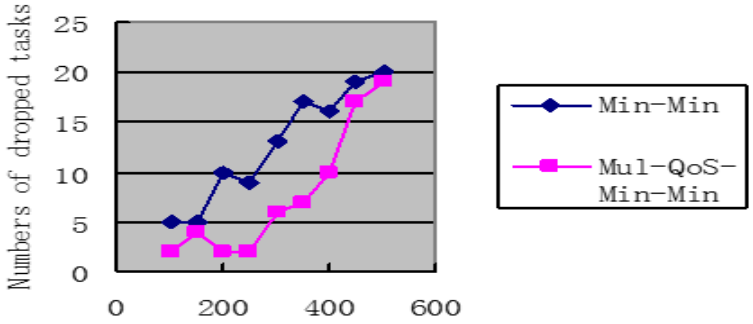

Figure 7. The Failure Rate of the Two Algorithms

5. Conclusions

In this paper, we proposed a resource management model and a new task scheduling algorithm based on QoS constraints according to this model. Compared with the traditional Min-Min algorithm, the new algorithm not only regard makespan as the only optimizing goal of scheduling, but also give full consideration to the user's QoS requirements. In one word, our algorithm can more truly reflect the actual situation of task scheduling and meet various QoS requirements of tasks.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC 61370060), and the Fundamental Research Funds for the Central Universities (2015JBM035).

References

[1] Xu, Baomin, Chunyan Zhao, Enzhao Hu, and Bin Hu. Job scheduling algorithm based on Berger model in cloud environment. Advances in Engineering Software 42.7 (2011): 419-425.
[2] Xu Baomin, Ni Xuguang. Development Trend and Key Technical Progress of Cloud Computing. Bulletin of Chinese Academy of Sciences 30.2(2015):170-180.
[3] Xu, Baomin, Ning Wang, and Chunyan Li. A cloud computing infrastructure on heterogeneous computing resources. Journal of Computers 6.8 (2011): 1789-1796.
[4] Xia, Yang, Lei WANG, Qiang ZHAO, and Gongxuan ZHANG. Research on Job scheduling algorithm in Hadoop. Journal of Computational Information Systems 7, no. 16 (2011): 5769-5775.
[5] http://www.valleytalk.org/wp-content/uploads/2013/03/fair_scheduler_design_doc.pdf.
[6] http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html
[7] Sabata, Bikash, Saurav Chatterjee, Michael Davis, Jaroslaw J. Sydir, and Thomas F. Lawrence. "Taxonomy for QoS specifications." In Object-Oriented Real-Time Dependable Systems, 1997. Proceedings. Third International Workshop on, pp. 100-107. IEEE, 1997.
[8] Golconda, Kavitha S., Fusun Ozguner, and Atakan Dogan. A comparison of static QoS-based scheduling heuristics for a meta-task with multiple QoS dimensions in heterogeneous computing. In Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, p. 106. IEEE, 2004.
[9] Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 41, no. 1 (2011): 23-50.

Authors

Yi Zhang is an undergraduate student at the school of computer and information technology, Beijing Jiaotong University. Her current research interests include cloud computing and big data processing.

Dr. Baomin Xu is an associate professor in School of Computer and Information Technology, Beijing Jiaotong University, China. His research interests include Large Scale Data Analysis Based-on Cloud Computing, and Complex Network (Link Prediction, Community Detection).Until now he has published more than 50 research papers in referred journals and proceedings.