A Dynamic Scheduling Workflow Algorithm Based On Critical Path

Zhu Tongyue¹,a, Yang Wenhui²,*

¹School of Information Science and Technology, Chengdu University of Technology, Chengdu, Sichuan, China
²* School of Information Science and Technology, Chengdu University of Technology, Chengdu, Sichuan, China

aemail: zhutongyue@stu.cdut.edu.cn, *email: ywhui@cdut.edu.cn

Abstract. With the explosion in the size of scientific workflows, local workflows can no longer satisfy existing computing needs, and cloud computing platforms have become the first choice for scientific workflows. Compared with traditional local computing, cloud computing platform not only needs to consider the loss of scheduling time and transmission time, but also involves the cost of cloud resources. Therefore, whether the scheduling scheme is reasonable becomes the decisive factor of workflow efficiency. Aiming at how to achieve efficient scheduling under cost constraints, a dynamic scheduling algorithm based on critical path is proposed. The algorithm uses the optimized Dijkstra algorithm to classify the task nodes, guarantees the completion time of key nodes in the scheduling process, adjusts the priority of nodes dynamically, and selects the best resources according to the loss weight. Experiments show that the optimization rate increases with the increase of the number of task nodes. The algorithm proposed in this paper is suitable for large-scale workflow operation and can effectively reduce the scheduling time.

1. Introduction

The amount of data has expanded rapidly, and the business needs for processing large-scale data calculations are becoming more and more important. In order to achieve a certain business goal, the use of computers to transfer information between multiple participants and the introduction of scientific workflow has become an indispensable link in business process automation. Cloud computing is suitable for processing large-scale data due to its easy-to-obtain, low-cost, and high-efficiency characteristics. Scholars at home and abroad have done in-depth research on how to use cloud resources to plan a reasonable task scheduling plan[1-7]. HEFT algorithm[8] is a basic static scheduling algorithm that does not consider the cost and schedules all tasks to be executed on the virtual machine that can complete it earliest. Literature [9] proposed a multi-stage workflow task scheduling algorithm based on the Hybrid algorithm, and divided the solution process into three stages: workflow classification, task priority determination, and optimal virtual machine selection. Literature [10] divides the workflow into multiple stages for data-intensive workflows, and completes the task scheduling stage by stage through the allocation algorithm according to the estimated execution time and transmission time. Literature [11] conducted predictive modeling on the price of Preemptive virtual machines, obtained the lowest cost bid strategy, integrated execution time limit, and configured workflow instances.

For users, the cost of using cloud resources has become their core concern. Obtaining the optimal scheduling plan and reducing the scheduling duration within the budget are the problems that this article
needs to solve. The most typical algorithm is the LOSS algorithm [12] proposed by Rizos Sakellariou and Henan Zhao. Firstly, the task is arranged on the resource with the highest configuration, and then the virtual machine with the lowest loss weight is selected to perform the calculation according to the loss weight. Literature [13] proposed the HCBS algorithm and a greedy algorithm on an amortized basis, calculated the slack cost between the user's budget and the worst execution cost, allocated resources according to the value function, completed the task assignment, and updated the slack cost. Literature [14] proposed the MSLBL algorithm based on HCBS. Firstly, the slack cost of the task is calculated. After the task scheduling is completed, the slack cost of the task is added to the budget of the next task, and then the task is mapped to the virtual machine, using the earliest time first algorithm. Literature [15] proposed the ESBL algorithm based on the MSLBL algorithm, submitted key node tasks to key virtual machines for execution without exceeding the budget cost, reduced communication time and communication loss, and further reduced the length of task set scheduling.

The above algorithm proposes innovative ideas in terms of budget division, and selects the earliest completed machine under the budget standard. However, the above algorithm allocates all slack costs to the next task, which is prone to uneven budget allocation, and does not take into account the dynamic adjustment of scheduling priority. This paper proposes a dynamic scheduling algorithm based on the critical path, which divides the slack cost equally according to the proportion of the remaining task execution time, and selects the resource with the lowest loss to perform the calculation according to the loss weight.

2. Materials and Methods

2.1 Workflow scheduling model design

Scientific workflow is composed of a series of sub-tasks and dependencies between tasks, usually displayed in the form of directed acyclic graph. In this paper, the scientific workflow is modeled as $G=<T,D>$, $T$ represents the set of subtasks that need to be executed in the workflow $\{T_{1}, T_{2}, T_{3},..., T_{n}\}$ and $D$ represents a set of dependencies between any two connected subtasks $\{D_{1}, D_{2}, D_{3},..., D_{n}\}$. $D_{i,j}$ represents the communication time required from the virtual machine where Task $i$ is located to the virtual machine where Task $j$ is located. This article assumes that the subtask task is the smallest unit of scientific workflow and cannot be divided again. In DAG, there is only one entry node and one exit node. The task node without the predecessor parent node is defined as the entry node Task$_{enter}$, and the task node without the successor child node is defined as the exit node Task$_{exit}$.

2.1.1 Resource model

The resource center is composed of virtual machine resources with different computing capabilities and computing costs, and the execution time of each task on different virtual machines is different. Each task selects resources from the resource center to perform calculations for the task according to the scheduling algorithm through the scheduling system.

$VM_{i}=<mips_{i}, price_{i}, Time_{i}>$  \hspace{1cm} (1)

$mips_{i}$ represents the computing power of the virtual machine $VM_{i}$, which is determined by the configuration of the virtual machine's I/O speed, CPU performance, and hard disk storage. $Price_{i}$ represents the cost of using the virtual machine $VM_{i}$ during the billing period. $Time_{i}$ represents the time for Task$_{i}$ to perform calculations on the virtual machine $VM_{i}$.

$\frac{Tasksize_{i}}{Mips_{i}} = Time_{i}$ \hspace{1cm} while $Filesize_{i} < diskvm_{i}$  \hspace{1cm} (2)

2.1.2 Task model

$Task_{i}=<Tasksize_{i}, Filesize_{i}, Budget_{i}, AverageTime_{i}, Status_{i}>$  \hspace{1cm} (3)

Task$_{i}$ represents the amount of tasks executed by Task$_{i}$, which is a key basis for affecting task execution time. Filesize$_{i}$ indicates that the amount of data in Task$_{i}$ data transmission process is related to
the task transmission time. Budgeti represents Taski budget division, dividing the remaining budget according to the ratio of Tasksizei and Filesizei, occupying the remaining workflow data tasks. The Budgeti calculation formula is as follows:

\[ \text{Budget}_i = \frac{\text{Tasksizei} + \text{Filesizei}}{\sum_{i} \text{Tasksizei} + \text{Filesizei}} \times \text{Cost}_{\text{residue}} \quad (4) \]

Averagetimei represents the average value of the execution time of Taski on each virtual machine, and the calculation formula is as follows:

\[ \text{Averagetime}_i = \frac{\sum_{\text{VM}_{\text{num}}} \text{Time}_{\text{Taski}}^\text{VM}_i}{\text{VM}_{\text{num}}} \quad (5) \]

Statusi represents the execution status of Taski. This article sets the task execution status as the following three: Scheduled, NotScheduled, Delayed.

### 2.1.3 Time model
Taski is the predecessor node of Taskj, and Taskk is the successor node of Taskj. Taski needs to execute calculation after the execution of the predecessor node Taskj is completed. Transferi represents the transfer time required from Taski to Taskj, which is the workflow construction D_j in workflow modeling.

The latest task start time LST(j) is calculated as the minimum value of the latest start time of the successor node Task, and the time difference between Task and Task minus the average execution time of Taskj. The calculation formula of LST(j) is as follows:

\[ \text{LST}(j) = \text{Min}(\text{LST}(n) - \text{Transfer}_i^j) - \text{Averagetime}_j \quad (6) \]

The earliest task start time EST(j) is calculated as the sum of the earliest completion time of the predecessor node Taski and the average execution time of Taski. The calculation formula of EST(j) is as follows:

\[ \text{EST}(j) = \text{EST}(i) + \text{Averagetime}_i \quad (7) \]

The earliest task completion time EFT(j) is calculated as the sum of the earliest completion time of the precursor node Taski and the average transmission time from Taski to Taskj. The calculation formula of EFT(j) is as follows:

\[ \text{EFT}(j) = \text{EFT}(i) + \text{Avg}(\text{Transfer}_i^j) \quad (8) \]

### 2.1.4 Price model
The lowest budget Costmin is the budget needed to ensure the completion of all tasks with the lowest price of resources. In the HEFT scheduling algorithm, the cost required for the system to configure the resource with the shortest completion time for each task node is the highest budget Costmax. In the experiment, the value \( \theta \in (0,1) \) of the budget tightness factor \( \theta \) is customized to control the tightness of the overall budget Costall on the basis of meeting the minimum budget. Costall calculation formula is as follows:

\[ \text{Costall} = \text{Cost}_{\text{min}} + \theta (\text{Cost}_{\text{max}} - \text{Cost}_{\text{min}}) \quad (9) \]

After each task scheduling is completed, the task budget division is re-planned according to the actual expenses. Costresidue calculation formula is as follows:

\[ \text{Cost}_{\text{residue}} = \text{Cost}_{\text{all}} - \text{Cost}_{\text{actual}} \quad (10) \]

### 2.1.5 Workflow scheduling priority
Before executing the scheduling, it is necessary to arrange the scheduling sequence for the tasks in advance, from the exit node Taskexit upward recursive calculation until the entry node Taskenter. If Taski is the exit node Taskexit, the average execution time of Taskexit is the priority value of the node. Considering that the task nodes with more subsequent nodes have a greater impact on the workflow, the influence of the number of subsequent nodes is considered when calculating the priority.

\[ \text{Rank}(i) = \begin{cases} \text{Averagetime}_i & \text{while parent(Taski)} \in \mathcal{O} \\ \text{Averagetime}_i + \max(\text{transfer}_i^j + \text{Rank}(j)) + \beta \times \text{child}(\text{Task}_i) & \end{cases} \quad (11) \]
The urgent value Urgency(j) is the latest start time of the successor node Taskj minus the earliest end time of the predecessor node Taski. If the urgent value is a negative number, it means that Taskj has been delayed. The calculation formula of Urgency(j) is as follows:

\[ \text{Urgency}(j) = \text{LST}(j) - \text{EFT}(i) \]  

2.1.6 Loss weight calculation

The loss weight calculation method is improved according to the LOSS algorithm. In terms of time loss, in addition to the task execution time, the waiting time of the virtual machine and the task transmission time are also considered, and the actual completion time of the task shall prevail. The HEFT algorithm is used to calculate the fastest completion time and the amount required for the task to select the best quality resource to complete the task scheduling regardless of the cost. During scheduling, resources with lower loss weights are preferentially selected to perform calculations.

\[ \text{LossWeight}(\text{Task}_i, \text{VM}_n) = \frac{\text{Time}(i) + \text{averageTime}(i) + \text{wait}(vmn) + \text{transfer}(i,j) - \text{time(heft)}}{\text{Cost}_{\text{HEFT}} - \text{Cost}_{\text{actual}}} \]  

2.1.7 Algorithm optimization rate

The final algorithm scheduling duration uses HEFT algorithm as the comparison group and compares the ratio of the difference between the current algorithm and the HEFT algorithm scheduling time occupying the total HEFT algorithm scheduling time. The calculation formula for optimization rate Optimization is as follows:

\[ \text{Optimization} = \frac{\text{MapScan}(\text{CPDS}) - \text{MapScan}(\text{HEFT})}{\text{MapScan}(\text{HEFT})} \]  

2.2 Algorithm design

The CPDS algorithm proposes to classify task nodes based on the critical path and the shortest path, and divide the budget according to the proportion of tasks in the planning stage. Priority is given to the calculation of key nodes during scheduling, and the resource with the earliest completion time is selected when the remaining budget meets the execution conditions; for non-critical nodes, the resource with the lowest loss weight is selected to perform calculations within the budget. In the scheduling process, the budget division and scheduling sequence of each node are dynamically adjusted according to the execution status of the task node.

The algorithm execution process is as follows:
1. Calculate the execution time of each task on different configuration resources
2. Arrange the rank values of task nodes in descending order to form a scheduling queue, and use the HEFT algorithm to calculate the execution time and cost of each task on the optimal virtual machine
3. Use the optimized dijkstra algorithm to get the shortest path and critical path
4. Divide the budget for tasks based on the proportion of tasks
5. Assign virtual machines to tasks according to the scheduling sequence, divide the budget of the remaining tasks again after each scheduling is completed, and adjust the calculation sequence if there is a delay in the scheduling of tasks
6. Calculate the total execution time and total budget

CPDS algorithm

Input: Workflow task set G=<T,D>, resource center VM set, Costall
Output: Time(G), makespan(G)
1. Recursively calculate the task Rank value, arrange the Rank value in descending order to get the task scheduling queue Q (G)
2. Calculate the EFT (Taski, VMn) of all tasks on each virtual machine
3. Use the HEFT algorithm to record the optimal EFT of the task and the Cost that can meet the conditions
4. Use ClassificationTask() to classify task nodes
5. Get key node set A (Task) and non-key node set B (Task)
6. Divide the budget for task nodes according to the proportion of tasks
7. for i=0 to Tasknum do{
8. if (Taski ∈ A) {
9. Calculate the cost of the EFT minimum virtual machine
10. Calculate Costresuide and Costmin of remaining nodes
11. while(Costresuide<Costmin) {
12. The virtual machine configuration is degraded
13. The virtual machine with the lowest loss weight is selected
14. Calculate Costresuide and Costmin of remaining nodes
15. }
16. else if(Taski ∈ B) {
17. Calculate the loss weight of the node Taski LossWeight (Taski, VMn)
18. Sort LossWeight in descending order
19. Select the virtual machine with the smallest loss weight under the budget
20. }
21. Calculate remaining nodes Urgency(i)
22. if(Urgency<0) {
23. Adjust the priority dynamically and adjust the scheduling sequence
24. }
25. Calculate Costresuide and allocate the budget for the remaining tasks
26. }
27. output Time(G) and makespan(G)

Algorithm 2 Adjust the scheduling sequence dynamically

In the planning stage, calculating the Rank value of each task node recursively from the exit node, and arranging them in descending order to obtain the scheduling order of the task nodes. In the scheduling process, the task priority is adjusted according to the actual situation of task execution, and the Urgency value of the task node is calculated. If there is a negative value, the task node status is delayed, and the remaining nodes need to be reconstructed to rearrange the scheduling sequence. The priority of the node with a negative urgency level is increased to the highest, and the remaining nodes are still arranged in descending order of Rank value.

Algorithm 3 Task classification algorithm

Critical nodes and non-critical nodes are classified according to the critical path and the shortest path. When the critical path and the shortest path overlap, the critical task node is the node of the non-shortest path on the critical path. In the scheduling process, if there is a delayed node, it will also be added to the key node.

The method of finding the shortest path adopts the optimized dijkstra algorithm to traverse all subsequent nodes of the scheduled node, calculate the path length, and add the calculated node to the calculated queue. Sorting the path lengths of all nodes in the calculated queue in ascending order, selecting the head node to write in the scheduling queue, and updating the unscheduled queue and the calculated queue.

Find the shortest path algorithm

1. N : the set of scheduled nodes
2. M : the set of unscheduled nodes
3. S : the calculated node queue and record the path length
4. While (N(ChildNode) != NULL) {
5. for i=0 to Tasknum {
6. if (Task ∈ N) {
7. Continue;
8. }
9. else if (Task ∈ S) {
10. Continue;
11. }
12. }
13. Compute node execution computing time and communication time between two tasks
14. Write the node to the sort storage in the set S;
15. Select the first point of set S and write it in the scheduled set N;
16. Update M and delete this node in M
17. Update S and delete this node in S
18. }
19. }
20. }

3. Results & Discussion

3.1 Examples
This article verifies the feasibility of the CPDS algorithm through a specific scheduling example. Figure 1 is a schematic diagram of a typical scientific workflow. There are 11 task nodes in total, and there are dependencies between nodes. When any two connected task nodes are assigned to the same virtual machine for execution, the communication time on the boundary can be ignored. The DAG model is shown in Figure 1.

![DAG example diagram](image)

Figure 1 DAG example diagram

The resource center provides a resource set VM={VM1, VM2, VM3, VM4} for scientific workflow. When the storage capacity of the virtual machine hard disk does not meet the task node execution requirements, the task cannot be executed on this virtual machine. The cost of transferring data between any virtual machines is negligible. The execution cost of each task node on different resources is shown in Table 1, and the execution time is shown in Table 2.

| Task | VM1 | VM2 | VM3 | VM4 |
|------|-----|-----|-----|-----|
| 0    | 40  | 48  | 43  | 32  |
| 1    | 53  | 60  | 56  | 41  |
| 2    | 44  | 50  | -   | 28  |
| 3    | 48  | 64  | 49  | -   |
| 4    | 46  | 72  | 43  | 46  |
| 5    | 97  | 72  | 83  | 34  |
| 6    | 67  | 76  | 78  | 78  |
| 7    | 73  | -   | 75  | -   |
| 8    | 22  | 24  | 19  | -   |
| 9    | 19  | 34  | 66  | -   |
| 10   | -   | 35  | 50  | -   |

Table 1 Execution Cost Table

| Task | VM1 | VM2 | VM3 | VM4 |
|------|-----|-----|-----|-----|
| 0    | 43  | 60  | 45  | 71  |
| 1    | 65  | 71  | 65  | 94  |
| 2    | 94  | 94  | 94  | 80  |
| 3    | 80  | 80  | 80  | 83  |
| 4    | 83  | 83  | 83  | 38  |
| 5    | 38  | 38  | 38  | 36  |
| 6    | 36  | 36  | 36  | -   |
| 7    | 15  | 15  | 15  | 50  |
| 8    | 15  | 15  | 15  | 50  |
| 9    | 15  | 15  | 15  | 50  |
| 10   | 15  | 15  | 15  | 50  |

Table 2 Execution Time Table
According to the average execution time of each task node and the communication time between nodes, the Rank value of the task node and the priority order of task node execution are obtained from the recursive calculation from the exit node as shown in Table 3. The scheduling sequence obtained by descending order is \{Task0, Task1, Task3, Task4, Task2, Task5, Task7, Task6, Task8, Task9, Task10\}, the key nodes of the execution task classification algorithm are \{Task0, Task4, Task7, Task10\}

| Task0 | Task1 | Task2 | Task3 | Task4 | Task5 | Task6 | Task7 | Task8 | Task9 | Task10 |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|
| Rank  | 347.25| 289.5 | 274.92| 286   | 285.75| 222.25| 195.34| 211   | 104   | 90     | 57     |
| priority | 1    | 2     | 5     | 3     | 4     | 6     | 8     | 7     | 9     | 10     | 11     |
| Key   | √     | √     | √     | √     | √     | √     | √     | √     | √     | √      |

Table 4 Budget Division Table

| Task0 | Task1 | Task2 | Task3 | Task4 | Task5 | Task6 | Task7 | Task8 | Task9 | Task1 |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Proportion | 7.08 | 8.65 | 6.96 | 9.78 | 9.31 | 15.37 | 11.15 | 13.87 | 5.55 | 3.94 | 8.32 |
| Budget | 36.67 | 44.8 | 36   | 50.66| 48.22| 79.61 | 57.75 | 71.84 | 28.74| 20.4 | 43    |

Without limiting the budget, the optimal virtual machine scheduling duration using the HEFT algorithm is 303, and the total cost is 569. Therefore, the highest cost is set at 569, and the lowest cost is the sum of the lowest costs of all tasks at 465. In this experiment, the budget factor is 0.5, and the total budget is 517. In the planning stage, the total budget is divided according to the proportion of the average execution time of the task. Whenever a task scheduling is completed, the task budget is re-planned according to the actual remaining budget. Figure 2 is an example diagram of CPDS algorithm scheduling, and Figure 3 is an example diagram of ESBL algorithm scheduling.

The total length of ESBL scheduling under the budget of 517 is 356, and the total cost is 514. The CPDS algorithm has a total scheduling length of 324 under a budget of 517 and a total cost of 514. In the case of the same cost, the CPDS scheduling time is reduced by 32, and the optimization rate is 10.56%.
3.2 Result analysis

The experiment was run on a machine configured with Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz and 8.00 GB RAM, the operating system was Windows10, and the Workflowsim simulation platform is used to simulate the CPDS algorithm. The relevant data of the scientific workflow in the experiment uses the workflow data randomly generated by Workflowsim, including the dependencies between tasks, the calculation amount of task nodes, and the resource center provides the optional resource configuration for the workflow and the corresponding execution time, execution cost and so on.

In order to verify the effect of the experiment, firstly, carrying out simulation experiments on scientific workflows of different scales. The number of control task nodes is from 10 to 1000, and the fixed budget factor is 0.5. Every 100 tasks are divided into an interval, and 100 scientific workflows are randomly generated in each stage. The calculation results are averaged as the experimental results of the current interval. The experimental results are shown in Figure 4.

![Figure 4 Comparison of Workflow Algorithm Results at Different Scales](image)

From the data in the figure, it can be concluded that the CPDS algorithm has an optimization rate of 8.32%-15.44% compared to the ESBL algorithm. As the number of task nodes increases, the optimization rate also gradually increases, up to 15.44%. Therefore, the CPDS algorithm is more suitable for complex workflow operations with more task nodes.

Then performing simulation experiments according to the different budget tightness, controlling the budget factors as 0.3, 0.5, 0.8, and the number of tasks ranges from [10,100]. Every twenty tasks are divided into an interval, and 100 scientific workflows are randomly generated in each interval. The calculation results are averaged as the experimental results of the current interval. The experimental results are shown in Figure 5.

![Figure 5 Comparison of Different Budget Factors](image)

From the data in the figure, it can be concluded that when the budget factor is high, that is, when the budget is relatively sufficient, the optimization rates of the two algorithms are not much different. In the case of a low budget factor, the scheduling duration proposed in this paper, which is obtained by the method of prioritizing the critical path and considering the loss weight, is more suitable for the optimal scheduling algorithm.
4. Conclusion
Aiming at how to efficiently schedule task workflows under cost constraints, this paper proposes a scheduling algorithm based on the critical path, which selects virtual machines according to the loss weight and adjusts the scheduling priority dynamically. The feasibility of the method is verified through specific experiments, and the effectiveness of the method is verified by comparing the results of simulation experiments. In future research, factors such as the cost of task transmission loss and the virtual machine settlement cycle will be considered, and how to reduce budget costs will be further studied.

References
[1] Jia, Z. H, Pan L , Liu X, Li X. J. (2021). A novel cloud workflow scheduling algorithm based on stable matching game theory. The Journal of Supercomputing. 6: 1-28.
[2] Ahmad W , Alam, B, Atman, A. (2021). An energy-efficient big data workflow scheduling algorithm under budget constraints for heterogeneous cloud environment. The Journal of Supercomputing.
[3] Raman N, Wahab, A. B, Chandrasekaran S (2021). Computation of workflow scheduling using backpropagation neural network in cloud computing: a virtual machine placement approach. The Journal of Supercomputing, 10.
[4] Lz A, Lz A, As B (2020). Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments - scattedirect. Information Sciences. 531:31-46.
[5] Ismayilov G, Topcuoglu HR (2020). Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. Future Generation Computer Systems, 102:307-322.
[6] Gao T, Wu CQ, Hou AQ, et al. (2019) Minimizing financial cost of scientific workflows under deadline constraint multi-cloud environments. Proc of the 34th ACM/SIGAPP Symposium Applied Computing. 114-121.
[7] Yu J, BUYYA R, Than C K.A. (2005) cost-based scheduling of scientific workflow application on utility grids The 1st International Conference on E-Science and Grid Computing. Washington, DC:IEEE Computer Society. 140-147.
[8] Topcuoglu H, Hariri S, Wu M. (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems. 13(3): 260-274.
[9] Zhang Y, Ma F. (2020) A Multi-Stage Workflow Task Scheduling Algorithm in Cloud Environment . Computer Applications and Software. 37(11):178-184.
[10] Chen J, Liu X. (2020) Data-intensive workflow scheduling based on phase division in cloud environment . Journal of Nanjing University of Posts and Telecommunications(Natural Science Edition),40(4):103-110.
[11] Liao J, Sun Q, Yang H, et al. (2020) Configuration and scheduling mechanism of spot instances meeting the execution time limit of workflow. 42(11):1957-1964.
[12] Sakellariou R, Zhao H, Tsiakkour J.E, et al. (2005)Scheduling workflows with budget constraints. Pisa,Italy:University of Pisa Dipartimentodi Informatica. 347-357.
[13] Arabnejad H, Barbosa JG. (2014) A budget constrained scheduling algorithm for workflow applications. Journal of Grid Computing. 12(4):665-678.
[14] Chen W, Xie G, Li R, et al. (2017) Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems. J. Future Generation Computer Systems. 74: 1-11.
[15] Zhang L, Wang L, Xiao M, et al. (2020) Efficient Work Flow Scheduling Algorithm Under Cost Budget Constraint in Heterogeneous Cloud Systems. Journal of Chinese Computer Systems, 6(6):1182-1187.