International Journal of Grid Distribution Computing
Vol.7, no.3 (2014), pp.31-40
http://dx.doi.org/10.14257/ijgdc.2014.7.3.04

On the Scheduling Algorithm for Adapting to Dynamic Changes of User Task in Cloud Computing Environment

Taoshen Li[1] and Xixiang Zhang[2]

1. School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China
2. College of Information Science and Technology, Hunan University, Changsha 4100075，China
{tshli@gxu.edu.cn; 77938710@qq.com}

Abstract

Focusing on the problem that users would remove or delete the task at any time in cloud computing environment, a new scheduling algorithm for adapting to dynamic changes of user task is proposed. At first, the algorithm uses directed acyclic graph (DAG) to describe the association relation of cloud computing tasks. When a task needs to be removed, the algorithm utilizes a cost function to determine whether to cancel this task, and then updates the DAG according to its dependence on removed task. Finally, it uses a heuristic algorithm to perform scheduling. Experimental results show that this algorithm can better avoid scheduling execution of revoked tasks and improve the executive efficiency of cloud computing resource. Its performance is better than Min-Min and Max-Min in executive time span.

Keywords: *cloud computing, scheduling algorithm, cancel task, dynamic change, directed acyclic graph (DAG)*

1. Introduction

In cloud computing environment, task scheduling is a key problem in cloud computing system, which needs to consider various factors restricting the user jobs and is responsible for selecting the most suitable resources of cloud computing for user job [1, 2]. According to the scheduling problem, the scheduling strategy can be divided to independent task scheduling and dependent task scheduling. Min-Min algorithm is a good scheduling strategy for independent task scheduling [3, 4]. For dependent task scheduling, the association relation between tasks is typically represented by means of a directed acyclic graph (DAG) and a number of heuristics algorithms have been proposed to schedule the nodes [5-7]. Sakellariou, *et al.*, [5] present a novel heuristic for DAG scheduling, which is based upon solving a series of independent task scheduling problems. Aiming to the problem of task scheduling under large-scale, heterogeneous and dynamic environments in grid computing, a heuristic algorithm based on fuzzy clustering is proposed in [6]. This algorithm uses a fuzzy clustering method to realize the reasonable clustering of processor network and reduce the resource search space and overall scheduling time. In order to improve the efficiency of scheduling, Lin, *et al.*, [7] have proposed a scheduling algorithm in grid based on dynamic decisive path. This algorithm first uses the common heuristic algorithm to schedule the tasks, and rebuilds a new DAG according to current scheduling relation and weight value of nodes and edges in

ISSN: 2005-4262 IJGDC

the DAG, and then utilizes the updated DAG to optimize the task node of decision paths and non decision path on the DAG.

However, on the condition that huge user tasks are scheduled with limited resources, for the reason of huge user groups in the cloud computing, there is the leap of situations that users remove or delete the task for personal factors after committing. The executions of these useless tasks would lead to waste of cloud resource and impact on cloud computing service provider benefits. Existing scheduling algorithms cannot solve the matter well. In consideration of above problems, this paper proposes a new scheduling algorithm for adapting to dynamic changes of user task in cloud computing environment.

The remainder of this paper is organized as follows. Section 2 presents system assumptions and model definition. In Section 3, a scheduling algorithm for adapting to dynamic changes of user task in cloud computing environment is proposed. Section 4 gives instance analysis of the proposed algorithm. The experimental result and performance of the algorithm is presented in Section 5. Finally, we summarize the paper with some concluding remarks in Section 6.

2. System Assumptions and Model Definition

2.1. System Assumptions

System assumptions are as follows:

(1) The task scheduling of cloud computing is defined as a task dependency scheduling. The jobs submitted to the cloud service center by user are generally more complex and larger, and have been split into several subtasks with known data dependence. Generally, association task scheduling is defined as a directed acyclic graph $G=(V,E)$ with nodes set V and edges set E, and $|V|=n$, $|E|=m$. The weight of node $v_i \in V$ represents amount of task computation, and the weight of edge $(i,j) \in E$ represents amount of communication between node v_i and node v_j. If a node in the DAG has no parent node, calls it as the entrance node. If a node has no any child node, calls it as the exit node. While there is more entrance or exit node in the DAG, these nodes are connected to a new node which is designated as a total entrance or exit node, and the weight value of each edge that is connected to the new entrance or exit nodes is 0. Each task node needs to wait for the completion of all of its parent nodes, and then it can begin to execute.

(2) Through virtualization technology, cloud resources are provided to the users in virtual machine resources. In the system, execution state of cloud resource is known, and execution cost of each subtask in any virtual machine is also known. The amount of delay and communication between each virtual machine is set to 0. The amount of data transmission between tasks with association relationship is known, which are produced at the time of decomposing job.

(3) Task in an assigned virtual machine is executed in a non preemptive way according to FCFS scheduling principle.

2.2. System Model Definition

System model definition is as follows:

Definition 1. Suppose cloud resources are provided through the virtual machine mode, set of the virtual machine is defined as follows:

$$VM=\{ vm_1, vm_2,..., vm_m\} \quad (1)$$

Where, vm_i is ith virtual machine resource(i=1,2,…,m), m is the number of virtual machine resources.

Definition 2. The subtask set of user job is defined as follows:

$$T=\{ t_1, t_2,..., t_n\} \quad (2)$$

Where, t_i is ith subtask to be decompose (i=1,2,…,n), n is the number of subtasks to be decomposed.

Definition 3. The subtask set which is involved at the time of cancelling user job is defined as follows:

$$U=\{ u_1, u_2,..., u_s\} \quad (3)$$

Where, u is ith subtask to be involved at the time of cancelling user job (i=1,2,…,s), s is the number of subtasks to be involved.

Definition 4. The running time of subtask t_i on the virtual machine vm_j is represented as $C[i][j]$, which is the weight of each node in the DAG.

Definition 5. The amount of communication between subtask t_i and subtask t_j is represented as $T[i][l]$, which is the weight of edge (i,j) from node v_i to node v_j in the DAG.

3. Algorithm's Description

The ideas of scheduling algorithm in this paper are as follows:

(1) At first, defining task dependencies by analyzing user task T, and then generating a dependency graph between tasks and updating right value of nodes and edges in task DAG. Finally, computing task priority according to the dependency graph, inducing pilot set and finding the priority of each task node.

(2) Getting the revocation task queue U, and then obtaining association task of each subtask in the revocation task queue according to the dependency graph, and determining whether is worth revoking these tasks.

(3) Updating or deleting the task information to be revoked in the DAG, and selecting a heuristic scheduling algorithm to complete the distribution of VM resource.

The main procedures of algorithm are presented as following.

Step 1. Analyze dependencies of user task T, and obtain dependency matrix (*Task_Dep*[][]) between tasks ;

Step 2. Get the task's computation amount (*Task_Computer*) and machine parameter (*Mach_Para*), and set weight value of nodes and edges in task DAG;

Step 3. *for each* $T_i \in \boldsymbol{T}$
Calculate in-degree (*Guide*[i]) and out-degree *(Inherit[i]*) of task T_i*;*
*S*ave pilot set (*PN*[i][]) of task T_i*;*
Compute priority of task T_i*;*
end for

Step 4. Set the priority of all the independent tasks to the lowest priority;

Step 5. Get the revocation task queue U;

Step 6. *for each* $U_i \in \boldsymbol{U}$
Push U_i into *Stack*;
end for ;

Step 7. *while (Stack != null)* //obtain association task of revocation task
Temp_Task = Pop Stack // Pop up the *Stack* top
for each $T_i \in \boldsymbol{T}$
if task_dep[i][*Num*(*Temp_Task*)] is true //There is dependency between T_i and *Temp_Task*

Push T_i into *Stack*;
end if
end for
if The Computation amount of revocation task *Temp_Task* > *Cost* // *Cost* is minimum cost of Communication
Delete *Temp_Task;*
end if
end while

Step 8. Update weight value of nodes and edges in task DAG;

Step 9. Select Min-min algorithm to schedule the tasks and distribute VM resource according to priority;

Step 10. Algorithm End.

4. Instance Analysis

This section will analyzes algorithm's performance by instance scheduling. Supposed the number of tasks is 10, a DAG is randomly generated as shown in Figure 1. Where, task n_0 is initial entrance task, and task n_9 is an ending task.

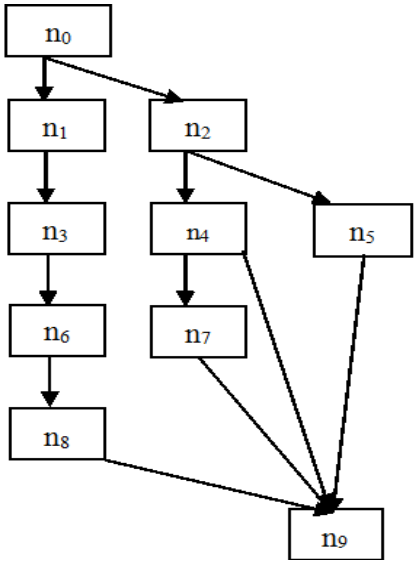

Figure 1. DAG of the Example With 10 Tasks

Its dependency matrix (*Task_Dep*[][]) is as following.

$$task\_dep = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

While the amount of data transmission from task T_i to task T_j ($T[i][j]$) is restricted to $C[i][j]$, the computation amount of task T_i on virtual machine vm_j is as following.

$C[i][j]$={{0,0,0},{60,40,20},{160,80,70},{120,80,40},{160,90,80},{80,30,10},
{70,30,10}, {160,70,40},{90,40,30},{80,40,20},{0,0,0}}

The scheduling result of using Min-Min algorithm is as shown in figure 2. Decision path generated by Min-Min algorithm is n_2, n_4, n_7, n_8, and its cost is 220.

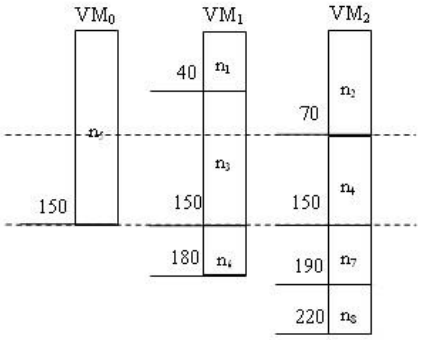

Figure 2. Gantt Charts of Min-Min Algorithm

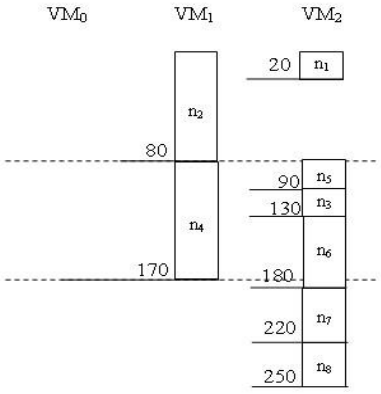

Figure 3. Gantt Charts of Max-Min Algorithm

The scheduling result of using Max-Min algorithm is as shown in Figure 3. Decision path generated by Max-Min algorithm is n_1, n_5, n_3, n_6, n_7, n_8, and its cost is 250.

When using the algorithm proposed in this paper for scheduling, it randomly gets a revocation task $U=\{n_4\}$ at first, and then finds task queue to be revoked according to calculating results. This revocation task queue is n_1, n_7. The updated DAG is as shown in Figure 4. Thus, the scheduling result of this algorithm is as shown in Figure 5. Decision path generated by the algorithm is n_1, n_5, n_3, n_6, n_8, and its cost is 170.

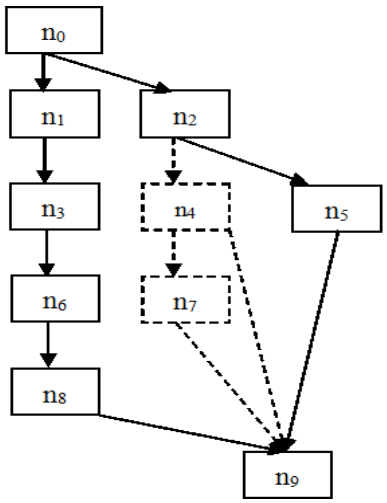

Figure 4. Updated DAG

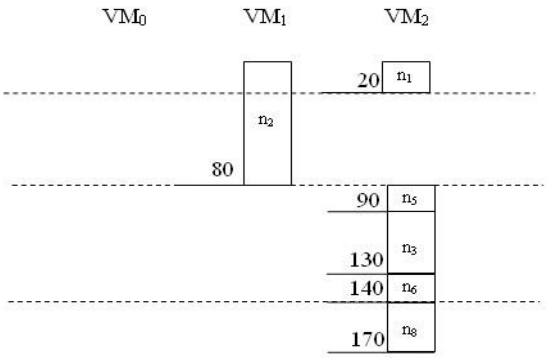

Figure 5. DAG of the Algorithm in this Paper

Obviously, because the proposed algorithm has deleted redundant tasks in scheduling queue, it can reduce the number of tasks, improve the scheduling efficiency and save the cloud resources.

5. Experimental Results and Analysis

All the experiments are performed on CloudSim platform, which is a cloud computing simulation software published by the grid laboratory of Melbourne University in Australia and the Gridbus project [8]. CloudSim can simulate various components in the cloud system, including cloud data center, host machine, virtual machine, data center broker, cloudlet, cloud information services, *etc.*,. It has good simulation effects for research and development of cloud computing [9]. In experiment, the scheduling algorithm is implemented in CloudSim by improving Data center Broker java.

The aim of simulation experiment is to verify the feasibility and effectiveness of the proposed algorithm by analyzing experimental results. In the experiment, the number of tasks is respectively 10, 15, 20, 25, 30, 35 and 40. Task parameter can be randomly generated, where the each task computing amount (*tLength*) takes a value in [10000,50000] at random, each task storage amount (*tFileSize*) randomly takes a value in [100,10000] at random and each task output communication amount (*tOutputSize*) takes a value in [100,10000] at random. The task is randomly set and generated by the task parameters shown in Table 1, and virtual resources are as shown in Table 2. In Table 2, *vpMipss* is the CPU execution rate of virtual machine, *vpSize* is the storage size of virtual machine, *vpRam* is virtual machine memory, and *vpBw* is virtual machine's bandwidth. The DAG diagram is generated randomly, and there is only one entrance task and an export task, no cyclic dependencies. The revocation cost is 1 and the number of revocation tasks is set to 10 percent.

Table 1. Task Parameter List

t	*tLength*	*tFileSize*	***tOutputSize***	***tClassType***
0	19365	4034	**300**	**1**
1	49809	432	**287**	**0**
2	30218	842	**194**	**0**
3	44157	432	**9400**	**2**
4	16754	359	**9939**	**2**
5	18336	331	**8843**	**2**
6	20045	9349	**310**	**1**
7	41493	492	**327**	**0**
8	30727	942	**403**	**0**
9	31017	5831	**294**	**0**

Table 2. Virtual Machine Resource list

vmID	*vpMipss*	*vpSize*	*vpRam*	*vpBw*	*vmComScore*	*vmStoScore*	*vmBwScore*
0	763	8300	2048	2000	7.5	8.3	3.9
1	887	2000	2048	2000	8.6	2.1	4.1
2	132	5000	2048	4000	1.3	5.4	8.0
3	209	2000	2048	4000	2.3	1.9	8.1

Figure 6 is the experimental results that the proposed algorithm compares with Min-Min algorithm and Max-Min algorithm. As seen at Figure 6, the proposed algorithm is better than Min-Min algorithm and Max-Min algorithm in execution time span.

Supposed task number is 20, virtual machine resources are 3, the revocation ratio of tasks is 10 percent, and the association relations are generated at random. Figure 7 shows the performance expense ratio saved by our algorithm, which experimental result is obtained after 15 times of experiments. Experimental results shows that proposed algorithm can better avoid revocation of useless task, and then improves the executive efficiency and saves the cloud resources. In the experiment, the average overhead ratio to save is 18.72%.

It should be noted that, the performance of the proposed algorithm is associated with the number of tasks to be removed by user and the dependence degree between tasks. When the number of tasks to be removed by user accounts for a larger proportion of the total number of tasks and the dependency between tasks is very close, the performance of the proposed algorithm can be improved significantly. When the number of task to be removed by user accounts for a smaller proportion of the total number of tasks and the dependency between tasks is more sparse, performance improvement of the proposed algorithm is not significant.

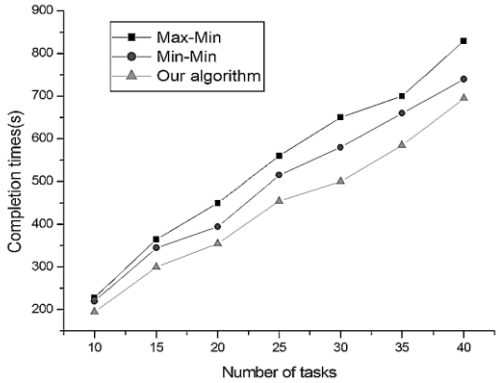

Figure 6. Comparison Between Several Scheduling Algorithms

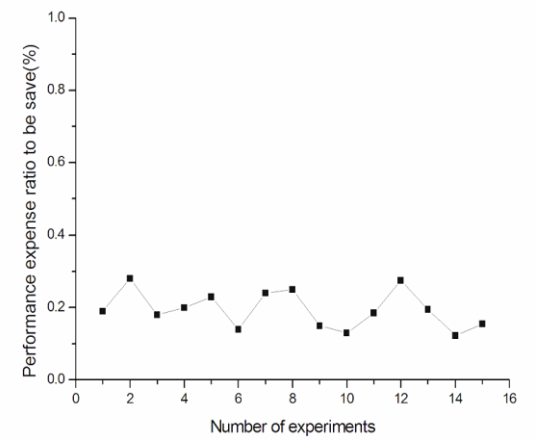

Figure 7. Performance Expense Ratio Saved by the Algorithm

6. Conclusion

Because the user group and related task is huge in the cloud computing environment, the situation that user removes useless task will exists generally, the this situation will revocation of the useless tasks will lead to waste of cloud resources. Aiming at this problem, this paper proposes a new scheduling algorithm for adapting to dynamic changes of user task. This algorithm utilizes DAG diagram to establish association relation between the cloud computing tasks. When a revocation task appears, algorithm checks every subtask related revocation task, then uses a cost function to determine whether or not need to remove. After removing each task, algorithm will updates the DAG and uses the Min-Min algorithm to perform scheduling. Experimental results show that the proposed algorithm can avoid the scheduling execution of revoked tasks and improve use ratio of cloud computing resource, while the algorithm is better than Min-Min and Max-Min in executive time span.

Acknowledgements

This work is supported by the National Natural Science Foundation of China(Grant No 61363067).

References

[1] S. Selvarani and G. S. Sadhasivam, "Improved cost-based algorithm for task scheduling in cloud computing", Proceedings of 2010 IEEE International Conference on Computational Intelligence and Computing Research, **(2010)** December 28-29, pp. 1-5, Coimbatore, India.
[2] L. Li, "An Optimistic Differentiated Service Job Scheduling System for Cloud Computing Service Users and Providers", Proceedings of 3rd International Conference on Multimedia and Ubiquitous Engineering, **(2009)** June 4-6, pp. 295-299, Qingdao, China.
[3] R. Braun and H. Siegel, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems", Journal of Parallel and Distributed Computing, vol. 6, no. 61, **(2001)**, pp. 810-837.
[4] K. Etminani and M. Naghibzadeh, "A min-min max-min selective algorithm for grid task scheduling", Proceedings of the 3rd IEEE/IFIP International Conference in central Asia on Internet, **(2007)** September 26-28, pp.1-7, Tashkent, Uzbekistan.
[5] R. Sakellariou and H. Zhao, "A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems", Proceedings of 18th International Parallel and Distributed Processing Symposium, **(2004)** April 26-30, pp.26-30, Santa Fe, NM, United states.
[6] X. Du, C. Jiang, G. Xu and Z. Ding, "A Grid DAG Scheduling Algorithm Based on Fuzzy Clustering", Journal of Software, vol. 11, no. 17, **(2006)**, pp. 2277-2288.

[7] J. Lin and H. Wu, "A New Scheduling Algorithm in Grid Based on Dynamic Decisive Path", Journal of Computer Research and Development, vol. 5, no. 45, **(2008)**, pp.841-847.
[8] N. Rodrigo, R. R. Calheiros, *et al.*, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software: Practice and Experience, vol. 1, no. 41, **(2011)**, pp. 23-50.
[9] R. Buyya, R. Ranjan and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities", Proceedings of the 2009 International Conference on High Performance Computing & Simulation, **(2009)** June 21-24, pp. 1-11, Leipzig, Germany.

Authors

Taoshen LI, he received the B.S., M.S. and PhD degrees in computer science and technology from the Central South University, Changsha, People's Republic of China, in 1982, 1989 and 2008, respectively. He is now a Professor and the Dean of School of Computer, Electronics and Information in Guangxi University, China. He is the author or coauthor of more than 200 technical papers in the areas of distributed database system, data mining, any cast routing algorithm, network and information security, wireless mesh network, and intelligent search engine. His current research interests are in the area of distributed database system, wireless mesh networks, cloud computing and intelligent processing.

Xixiang Zhang, he received the B.S. in computer science and technology from the Guangxi Normal University, China, in 2009, and the M.S. degrees in computer science and technology from the Guangxi University, China, in 2012. He is now a candidate PhD of College of Information Science and Technology in Hunan University, China. His research interests include cloud computing, speech and language information processing.