A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing

Meng Xu, Lizhen Cui, Haiyang Wang, Yanbing Bi
School of Computer Science and Technology
Shandong University
Jinan, China
xumeng_wf@163.com, clz@sdu.edu.cn, why@sdu.edu.cn, byb_126@126.com

***Abstract*—Cloud computing has gained popularity in recent times. As a cloud must provide services to many users at the same time and different users have different QoS requirements, the scheduling strategy should be developed for multiple workflows with different QoS requirements. In this paper, we introduce a Multiple QoS Constrained Scheduling Strategy of Multi-Workflows (MQMW) to address this problem. The strategy can schedule multiple workflows which are started at any time and the QoS requirements are taken into account. Experimentation shows that our strategy is able to increase the scheduling success rate significantly.**

Keywords-Cloud Computing, Scheduling, Multiple QoS Requirements, Multiple Workflows

I. INTRODUCTION

Cloud computing [1][2] has been a possible solution for providing a flexible, on demand computing infrastructure for a number of applications recently. Many companies and research institutes show great interests in cloud computing. A cloud computing environment has several key features[11]: 1) it is massively scalable, 2) can be encapsulated as an abstract entity that delivers different levels of services to customers outside the Cloud, 3) it is driven by economies of scale, and 4) the services can be dynamically configured (via virtualization or other approaches) and delivered on demand. As cloud computing has these features, the scheduling strategy for workflows on a cloud computing platform should consider the new features. First, cloud provides services for multi-users. So the scheduling strategy must cater to the different QoS requirements of different users. Second, there will be many workflow instances on the cloud platform at the same time. So the scheduler should be able to schedule multi-workflows. Last but not the least, new workflow may be started at any time.

A lot of Grid workflow management systems [3][4] have been developed to facilitate the composition and execution of workflow applications over distributed resources. Many heuristics [5][6][7] have also been proposed for workflow scheduling in order to optimize a single objective, such as minimizing execution time or budget constrained. However, more objectives need to be considered when scheduling workflows on cloud based on users'QoS requirements. Most algorithms developed for scheduling workflow focus on a single Quality of Service (QoS) parameter such as execution time or cost. However, if we consider more than one QoS parameter (e.g. execution cost and time) then the problem becomes more challenging.

To address the problem mentioned above, we introduce a scheduling strategy for multi-workflows with multiple QoS constrained for cloud computing in this paper. According to the key features of cloud and the characteristics of workflow applications, we have considered four factors which affect the total makespan and cost of workflow greatly. On the basis of these four factors, we generate a scheduling to satisfy with the users' QoS requirements and minimize the makespan and cost of workflows and increase the success rate of workflow scheduling (A success schedule is defined that the workflow is finished and meet the QoS requirements of the users).

The rest of the paper is structured as follows: Related work is discussed in Section 2. Then section 3 describes our research motivation. The scheduling strategy will be presented in Section 4. And Section 5 will show the experimental details and simulation results. Finally Section 6 concludes.

II. RELATED WORK

In this section, we will describe the related work of workflow scheduling. Scheduling in distributed systems is NP-complete in general. Some typical grid workflow scheduling algorithms are introduced in [4] by Jia Yu and Buyya. As we take a look at the scheduling algorithms they use, it is possible to see that many practical algorithms are designed for single workflow and do not consider the QoS requirements, such as the Min-Min Heuristic, the Greedy Randomized Adaptive Search Procedure algorithm [6] and the Heterogeneous Earliest-Finish-Time algorithm [8]. Jia Yu introduces a Cost-based scheduling algorithm [5] which partitions a workflow and assign deadline to every partition. However the algorithm is designed for single workflow. Though it can be used to schedule multiple workflows, the relationship of workflows is not taken into account.

Ke Liu et al. proposed a throughput maximization strategy for scheduling transaction intensive workflows [12].

978-0-7695-3747-4/09 $25.00 © 2009 IEEE
DOI 10.1109/ISPA.2009.95

But it is designed for transaction intensive workflows not for multiple workflows. The difference between transaction intensive and multiple workflows is that transaction intensive workflows are multiple instances of one workflow and multiple workflows are different workflows. In other word, multiple workflows may have completely different structures and transaction intensive workflows have the same structure. As multiple \workflows may have thoroughly different structure, this strategy would not perform well.

Zhifeng Yu and Weisong Shi [13] present a planner-guided strategy for multiple workflows. It ranks all ready tasks and decides which task should be scheduled first. However, if there are new lower rank workflows coming continuously, the higher rank task will not be scheduled to execute. In a massive scalable cloud, this situation will become true. On the other hand, this algorithm only considers the execution time, not other QoS requirements, such as cost.

In conclusion, although the algorithms mentioned above have their respective benefits in their particular application areas, none of them is particularly designed for multiple workflows with multiple QoS constrained scheduling and the key features of cloud computing are not specifically taken into consideration. Thus, the motivation of our work is to design a strategy to increase the success rate of scheduling and minimize the makespan and cost of workflows for cloud computing platform.

III. MOTIVATION

Inspired by some effective algorithms in the single workflow scheduling problem domain and the fact that cloud computing are gaining more popularity, we explore practical solutions to the real world challenges of scheduling workflow applications in a cloud environment. In this section, we discuss the motivation of our research.

There have been extensive comparative studies for static and dynamic algorithms [14], [15], [16], [17] in the context of scheduling single workflow. It is well concluded that static approaches outperform dynamic ones in most cases when resource do not vary over time. Nevertheless, the impracticality of static scheduling algorithms is recognized in researches [18]. And the dynamically configured (via virtualization or other approaches) and delivered on demand is one of the key features of cloud [11]. On the other hand, cloud is massively scalable. This leads to a fact that the services in cloud may change dynamically. In this case, the static scheduling algorithm is not a optimal choice. Moreover, all application on cloud will be used by a large number of users. And different user has different QoS requirements naturally. So a challenge for workflow systems on cloud is how to satisfy with multiple QoS requirements of different users. However, most existing scheduling algorithms are designed for scheduling of a single complicated workflow instance or only one QoS constraints rather than for multiple QoS constraints of multi-workflows. To address this problem, we need to develop new solutions.

The new algorithms must be able to support the scheduling of multiple workflows, and must be able to meet multiple QoS requirements. Therefore, the new algorithms should not only consider the completion time of each single workflow, but most importantly, the overall performance.

The overall performance can be defined in many aspects. Among them, we focus on the following aspects: (1) the scheduling success rate - it is obvious that a better system can satisfy with QoS requirements of more users; (2) the mean execution time of all workflows - the algorithms must strive to decrease the execution time of all workflows; and (3) the mean execution cost of all workflows - the algorithms should minimize the cost of workflow on the basis of meeting the QoS requirements of time.

To achieve the goals mentioned above, we propose a Multiple QoS Constrained Scheduling Strategy of Multi-Workflows (MQMW) for cloud computing which will be discussed in detail in Section 4.

IV. SCHEDULING STRATEGY

A. System Overview

Fig. 1 shows the overview of our workflow scheduling system. Users first submit workflow with their QoS requirements. The system then allocates appropriate services for processing the workflow tasks and schedules the tasks on the services according to the QoS requirements and the cloud environment.

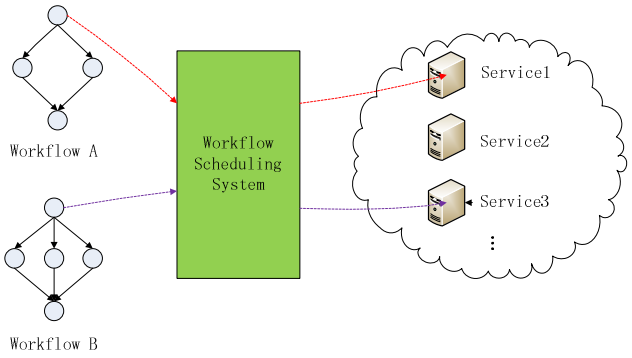

Figure 1. Overview of Scheduling Workflow

In order to schedule the workflow dynamically and optimize the resource allocation decision, the system we proposed consists of three core components: Preprocessor, Scheduler and Executor. The Preprocessor computes four attributes of the ready tasks which will be defined in section 4.2: the available service number, the covariance for time and cost, the time quota, the cost quota. In addition, the Preprocessor computes the time and cost surplus of the workflow. Then it submits the ready tasks the Scheduler queue, which is a sorted set containing all tasks from different users waiting to be scheduled. Then the Scheduler re-computes the above attributes of the tasks in the queue and then re-sorts all tasks in the queue according to the strategy which will be discussed below. The Executor selects the best service to sequential execute the tasks in the queue. When a task finishes, the Executor notifies the

Preprocessor which the task belongs to of the completion status. The collaboration is implemented by the continuous and dynamic event triggered communication among core components, as defined in Fig. 2.

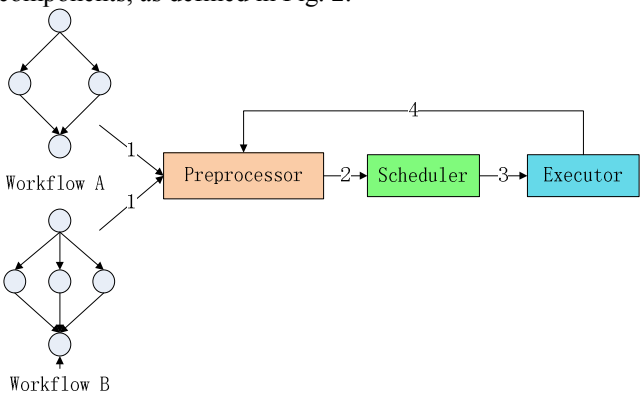

Figure 2. An overview of the scheduling strategy

1) Workflow submission: When a new workflow arrives, it is submitted to the Preprocessor. Then the Preprocessor computes the attributes of all ready tasks.

2) Preprocessing: After computing the attributes of all ready tasks within the workflow, the Preprocessor inserts the ready tasks into the queue. In the very first time, only entry tasks will be submitted. Afterwards, upon notification by the Executor of completion of a task, the Preprocessor will determine if any successor tasks become ready and submit them. The task attributes information is submitted along with the task.

3) Task scheduling: Whenever there are services available and a task is waiting in the queue, the Scheduler will re-compute all tasks currently present in the queue and sort all tasks and then repeatedly do:

a) Remove the first task in the queue;

b) Allocate the task to the service which is best fitted;

c) Insert the task into the next round queue if there are not services which can finish the task.

4) Task completion notification: When a task finishes successfully, the Executor will notify the Preprocessor of the task completion status.

B. Definition

Before we introduce our scheduling strategy, we first define some term which we will use later.

Definition 1: The Time Quota

The time limit when the task is executed.

$$TQ(t_i)=Min(T(t_i,R))+VT(t_i,R)*(QoS(time)-MinT(w))/\sum VT(t,R) \quad (1)$$

where Min(T(ti,R)) is the minimum time of the task ti executed, VT(ti,R) is the time variance which the task ti executed on all services, QoS(time) is the time attribute of QoS requirements, MinT(w) is the minimum time of the whole workflow executed, and ∑VT(t,R) is the time variance sum of all tasks in the workflow.

Definition 2: The Cost Quota

The cost limit when the task is executed.

$$CQ(t_i)=Min(C(t_i,R))+VC(t_i,R)*(QoS(cost)-MinC(w))/\sum VC(t,R) \quad (2)$$

where Min(C(ti,R)) is the minimum cost of the task ti executed, VC(ti,R) is the cost variance which the task ti executed on all services, QoS(cost) is the cost attribute of QoS requirements, MinC(w) is the minimum cost of the whole workflow executed, and ∑VC(t,R) is the cost variance sum of all tasks in the workflow.

Definition 3: The Covariance for Time and Cost

It takes different time and cost that one task is executed on different service. So we can get the covariance for time and cost covariance of the task.

Definition 4: The Time Surplus of Workflow

We call the difference between the time attribute of QoS requirements and the time which the workflow is executed if all service is dedicated as the time surplus of workflow.

Definition 5: The Cost Surplus of Workflow

We define the difference between the cost attribute of QoS requirements and the cost which the workflow is executed if all service is dedicated is the cost surplus of workflow.

C. Scheduling Algorithm

It is possible to directly applying traditional scheduling algorithms on multiple workflows scheduling. But they are developed for single workflow. It will have great practical limitation. One method is introduced in [19]. It creates a composite DAG by making the nodes which do not have any predecessors of all DAGs the immediate successors of a new common entry node, and all the exit nodes of the DAGs immediate predecessors of a new common exit node. A node does not have any predecessor because either itself is an entry node or its predecessors are executing or have finished when composition process occurs. These two extra common nodes have no computation and no communication between them and other nodes. But in reality, workflows may come at any time.

Our strategy is to sort all ready tasks according to their attributes defined in section 4.2. The consideration is:

1) The number of services in cloud is finite. And in most cases, the number of tasks waiting to be executed is larger than the service number. So a task with minimum available service number should be scheduled first. The reason is that the task would have not available services, if other tasks are scheduled first.

2) The tasks which belong to the workflow with minimum time surplus and cost surplus should be scheduled first. The reason is similar to the above.

3) The task with minimum covariance should be scheduled first. The covariance describes the strength of the correlation between time and cost. The minimum covariance means when the time decreasing a definite value, the cost will increase mostly. So the task should be scheduled first. Otherwise, we should pay more or the time would increase more.

Moreover, we need divide the overall time and cost over every task using (1) and (2). Then we get the sort algorithm shown in Fig. 3.

```
1.  T: a set of all tasks
2.  S: a set of available services
3.
4.  PROCEDURE: getReadyTasks(){
5.      while ∃t ∈T is ready do
6.          ReadyTasks← t
7.      end while
8.  }
9.
10. PROCEDURE: sortTask(ReadyTasks, q){
11.     while ∃t .ReadyTasks
12.         insertTask(t,q)
13.     end while
14. }
15.
16. PROCEDURE: insertTask(t,q){
17.     insert task t into queue q according to the strategy
    introduced above
18. }
```

Figure 3. Task Sort Algorithm

The scheduling algorithm is shown in Fig. 4.

```
1.  PROCEDURE: schedule(q,S){
2.      while q is not empty
3.          t←first task in q
4.          s←getService(t,S)
5.          schedule task t on service s
6.          q=q-{t}
7.          S=S-{r}
8.      end while
9.  }
10.
11. PROCEDURE: getService(t,S){
12.     select s.S, and the execution time and cost not
    exceed the time and cost quota of task t
13.     return s
14. }
```

Figure 4. Schedule Algorithm

V. EXPERIMENTS AND RESULTS

To evaluate the performance of the algorithm, we have developed an experimental simulator. And to simulate the algorithm with a wide range of workflow graph, we randomly generate various workflow graphs and also randomly generate a cloud environment. We compare our algorithm with the dynamic scheduling algorithm RANK_HYBD [13]. To guarantee fairness of the simulation, we conduct 250 iterations for each parameter set, and use the average of result values.

Our simulate cloud environment has 20 services. Every service can execute one task at the same time. There are 5 to 25 users to use the cloud to execute their workflows and every workflow has its own QoS requirements. We list 5 examples and the result of MQMW here as shown in Table 1. And the result of RANK_HYBD is shown in Table 2. From the Table 1 and Table 2, we can find that MQMW can meet all QoS requirements because it takes time and cost into account. On the other hand, RANK_HYBD improve the execution time of workflow wherever the cost is increased greatly.

TABLE I. MQMW RESULT AND QOS REQUIREMENTS

Workflow	MQMW Result		QoS Requirements	
	Time	*Cost*	*Time*	*Cost*
Workflow1	200	211	298	1244
Workflow2	301	540	304	1558
Workflow3	303	261	834	1196
Workflow4	310	673	1121	675
Workflow5	411	81	744	585

TABLE II. RANK_HYBD RESULT

Workflow	RANK_HYBD Result	
	Time	*Cost*
Workflow1	109	1353
Workflow2	122	1278
Workflow3	123	588
Workflow4	134.3333	1063
Workflow5	148	960

As a result, MWMQ has a higher success rate than RANK_HYBD. The reason is MQMW takes into account the QoS requirement while RANK_HYBD only considers the execution time. MWMQ always outperforms RANK_HYBD and the average success improvement rate of MQMW over RANK_HYBD increases from 69.3% to 92% when total number of concurrent workflows increases from 5 to 25 as shown in Table 1. And Fig. 5 shows the success rate of MQMW and RANK_HYBD.

TABLE III. SCHEDULING SUCCESS RATE

	MWMQ	RANK_HYBD
Average Success Rate	92%	69.3%

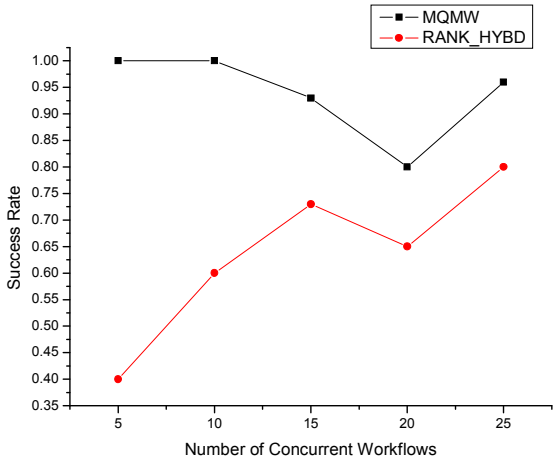

Figure 5. Success Rate of Scheduling

Fig. 6 shows the mean execution time of workflows. What we can conclude from Fig. 5 is MWMQ performs better than RANK_HYBD when the cloud environment has more workflows execute concurrently.

On the other hand, Fig. 7 shows the mean execution cost of workflows. From the graph, we can see MWMQ costs less than RANK_HYBD.

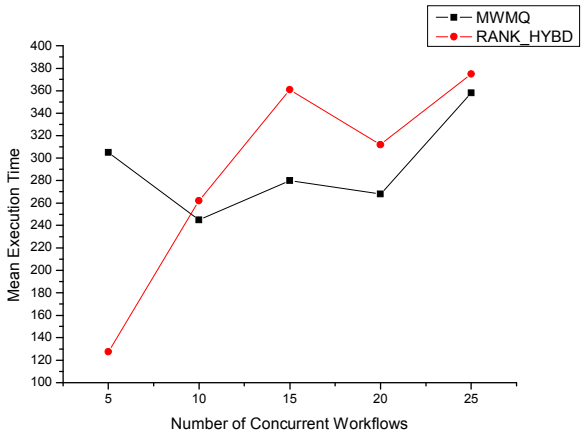

Figure 6. Mean Execution Time

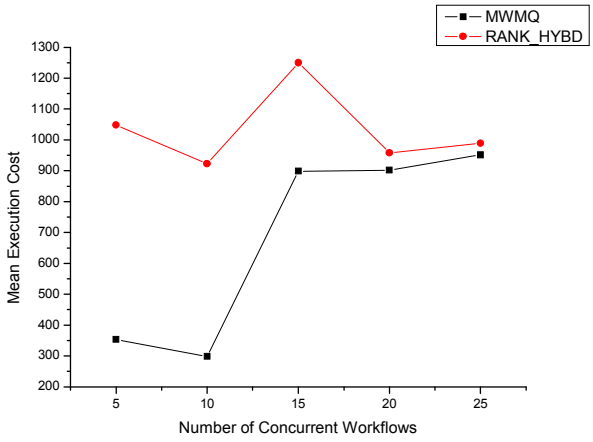

Figure 7. Mean Execution Cost

VI. Conclusion and Future Work

The workflows on cloud computing platform have multiple QoS requirements. It is a main challenge of cloud workflow system to scheduling the multiple QoS constrained workflows. However, most existing scheduling algorithms are not designed to address this problem.

To address this problem, we have proposed a multiple QoS constrained scheduling strategy of multiple workflows for cloud computing. The final comparison and simulation have practically demonstrated it can produce better scheduling results than RANK_HYBD.

In future work, we plan to conduct further evaluations with a realistic testbed. And we will study how to add more QoS constrained (reliability, availability, etc) to workflows.

Acknowledgement

The research work reported in this paper is partly supported by the National Natural Science Foundation of China under Grant No.60673130 and the Doctoral Fund of Ministry of Education of China under Grant No. 200804221031.

References

[1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic, “Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility”, Future Generation Computer Systems, Elsevier Science, Amsterdam, June 2009, Volume 25, Number 6, pp. 599-616.

[2] M. Armbrust, Above the Clouds: A Berkeley View of Cloud Computing, EECS Department, University of California, Berkeley, 2009.

[3] Deelman E. Mapping abstract complex workflows onto Grid environments. Journal of Grid Computing 2003; 1:25–39.

[4] J. Yu and R. Buyya, "Workflow Scheduling Algorithms for Grid Computing", Metaheuristics for Scheduling in Distributed Computing Environments, F. X. a. A. Abraham, ed., Springer, 2008.

[5] Jia Yu, Rajkumar Buyya and Chen Khong Tham, “Cost-based Scheduling of Scientific Workflow Applications on Utility Grids”, In 1st IEEE International Conference on e-Science and Grid Computing, Melbourne, Australia, Dec. 5-8, 2005.

[6] R. Sakellariou and H. Zhao, A Hybrid Heuristic for DAG Scheduling on Hetero-geneous Systems, The 13th Heterogeneous Computing Workshop (HCW 2004), Santa Fe, New Mexico, USA, April 26, 2004.

[7] Jia Yu and Rajkumar Buyya, “A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms”, Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing (HPDC 2006), IEEE CS Press, Los Alamitos, CA, USA, June 19-23, 2006, Paris, France.

[8] H. Topcuouglu, S. Hariri and M. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,”IEEE Transactions on Parallel and Distribution Systems, vol. 13, no. 3, pp. 260–274, 2002.

[9] G. C. Fox and D. Gannon, "Workflow in Grid Systems", Concurrency and Computation: Practice and Experience, vol. 18, no. 10, 2006, pp. 1009-1019.

[10] Mustafizur Rahman, Srikumar Venugopal and Rajkumar Buyya, “A Dynamic Critical Path Algorithm for Scheduling Scientific Workflow Applications on Global Grids”, Proc. Third IEEE International Conference on e-Science and Grid Computing, IEEE CS Press, 2007, pp. 35-42.

[11] Ian Foster, Yong Zhao, Ioan Raicu and Shiyong Lu, “Cloud Computing and Grid Computing 360-Degree Compared”, Grid Computing Environments Workshop 2008(GCE '08).

[12] Ke Liu, Jinjun Chen, Yun Yang and Hai Jin, “A throughput maximization strategy for scheduling transaction-intensive workflows on SwinDeW-G”, Concurrency and Computation: Practice and Experience, Wiley, 20(15):1807-1820, Oct. 2008.

[13] Zhifeng Yu and Weisong Shi, "A Planner-Guided Scheduling Strategy for Multiple Workflow Applications," icppw, pp.1-8, International Conference on Parallel Processing - Workshops, 2008.

[14] M. Wieczorek, R. Prodan, and T. Fahringer, “Scheduling of scientific workflows in the askalon grid environment,” SIGMOD Record, vol. 34, no. 3, pp. 56–62, 2005.

[15] M. Lopez, E. Heymann, and M. Senar, “Analysis of dynamic heuristics for workflow scheduling on grid systems,” in Proceedings of the 5th International Symposium on Parallel and Distributed Computing (ISPDC’ 06). IEEE, 2006, pp. 199–207.

[16] J. Blythe et al., “Task scheduling strategies for workflow-based applications in grids,” in Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid’05), Cardiff, UK, 2005.

[17] A. Mandal et al., “Scheduling strategies for mapping application workflows onto the grid,” in Proceeding of the 14th International Symposium on High Performance Distributed Computing (HPDC’05). IEEE, 2005, pp. 125–134.

[18] Z. Yu and W. Shi, “An adaptive rescheduling strategy for grid workflow applications,,” in Proceeding of 21st International Parallel and Distributed Processing Symposium (IPDPS’07), Long Beach, Florida, USA, March 2007.

[19] H. Zhao and R. Sakellariou, “Scheduling multiple dags onto heterogeneous systems,” in Proceedings of the 15th Heterogeneous Computing Workshop (HCW), Rhodes Island, Greece, April 2006.