2012 IEEE International Conference on Information Science and Technology
Wuhan, Hubei, China; March 23-25, 2012

A QoS–Awared Scientific Workflow Scheduling Schema in Cloud Computing

Cong Wan, Cuirong Wang, and Jianxun Pei

Abstract—**Now it is a trend to deploy the applications in the cloud computing environment. Most of scientific workflow (SWF) needs to process a large amount of data and requires parallel computing to reduce response time. SWF under grid computing environment have been already studied. As a business model, cloud computing is more scalable and cost-effective. In this paper, we propose a schedule schema that enable user to change the processing time by adjusting the budget and to change the price by adjusting the request response time in the SWF.**

I. INTRODUCTION

RECENTLY years, cloud computing has attracted more and more attention. It provides computing resources or services as pay-as-you-go price model [1]. Cloud computing has three service models: (1) Cloud Software as a Service (SaaS), the provider's applications running on a cloud infrastructure and accessible from various client devices by consumers. (2) Cloud Platform as a Service (PaaS), consumers deploy their applications onto the cloud infrastructure using programming languages and tools supported by the provider. (3) Cloud Infrastructure as a Service (IaaS), consumers rent processing, storage, networks, and other fundamental computing resources to deploy and run arbitrary software, which can include operating systems and applications [2].

Many Industry and academia published their products. For example, see [3]-[10]. According to different situation, Consumers can select the appropriate service model and product.

With the development of increasingly complex scientific applications and experiments (e.g. Biological Engineering, Nuclear simulation, earthquake engineering, climate modeling, federated computing for high-energy physics, etc.), scientists need to combine the thousands of applications and process large amounts of data which could be up to several hundred GB or more. As an important assistant tool for scientists, scientific workflow (SWF) gains more and more attractions in both research and development domains.

Compared with other environment, there are many advantages to execute workflow under cloud computing environment: (1) Resource in cloud is Scalable, as the data size expanding, user only need to request more Resource. (2) As long as the budget allows , the type and number of resource to requested are determined by user. (3) Reduce the cost, user make a purchase only if they need to execute their applications. (4) Not all the resource are necessary though out the whole execution of workflow, user could request resource or release them on demand. (5)The number of resource can be considered as infinite, but the budget is limited.

Cong Wan and Cuirong Wang are with College of Information Science and Engineering, Northeastern University, Shenyang 110044, China (e-mail: 10000cong@163.com).
Jianxun Pei is with Jilin Meteorological Information and Technical Support Center, Changchun, China.

In the past few years, there have been many workflow task schedule algorithms base on grid computing [7]-[9]. To some extent, these algorithms improve the efficiency of the workflow process and shorten the response time. However, as a business model, cloud computing charges on the usage of resource. Generally speaking, the response time and cost in the pay-as-you-go model are two typical constraints as well as two major QoS attributes. Very little work considered the scheduling algorithm of dependent Task Graphs in the cloud computing environment.

While response time is a constraint, user request resource with high process capability and use parallel computing as much as possible in order to finish the job on time. While response time is fulfilled, another target is to save money. In case of no limit for response time or budget is limited, from economic perspective, saving money would be the first target. We propose algorithms to address this issue.

The rest of the paper is organized as follows. Section II briefly reviews related work. Section III formulates the problem. Section IV describes the algorithm. Section V shows the experimental result. Conclusions and future work are presented in Section VI.

II. RELATED WORK

Rajkumar Buyya and his team have studied market-based resource allocation in grid, cluster and cloud. However, they did not point out the relationship between such attributes and which kind of application should be evaluated by such attributes. Some researchers focus on QoS in special applications under cloud computing environment.

III. PROBLEM FORMULATION

We formulate scientific workflow as a directed acyclic graph (DAG). DAG= {V, E} represent the workflow under a cloud computing environment, which the vertices of the graph denote a set of tasks and the edges of the graph denote a set of relationships of dependency. For each $(t_i, t_j) \in V$, that t_j must be executed after the end of the execution of t_i. The task computation time is described as the weight of vertex in the graph.

Processing tasks on different type of instances have different response time and cost. We use *Instance$_i$* to denote an instance of type *i*. Generally speaking, the instance with higher process capability is more expensive. With the growth of process capability, the price increase. However, the relationship between process capability and price is not linear. In order to describe the processing capability of instance, we use *pc$_i$* denote the workload that *Instance$_i$* could process in a time unit. We use *price$_i$* represent for the cost of renting *Instance$_i$* in a time unit. The Capital utilization of instance type *i*, denote by *Cui* is defined by follow:

$$cu_i = \frac{pc_i}{price_i} \tag{1}$$

We define the cheapest instance type i which cui is the largest as *InsCh*. *InsCh* can process more work at a low price, on the other hand, it lead to a long response time. The instance with largest *pc$_i$* is defined as *InsQuk*. It can process more work in a time unit.

The following variables are defined to describe a task t_i:

W_i: the workload of this task.

M_i: the maximum instance number that could be assigned to the task at the same time. It may be limited by cloud provider or task. For example, Amazon EC2 limits the maximum number of instance to a small number. In addition, parallel computing cannot be used by some algorithms. In that case, m should by 1.

R_i: resources that signed to task ti, which consist of a set of instance.

X_i: the decision vector $X_i=[x_{i1},x_{i2}\ldots x_{ik}]$, x_{ik} means the number of instance type *k* that user request multiplied by time unit. For example, we rent 2 instances of type k for 3 time unit, then x_{ik} is 6.

APT_i: the actual processing time of the task.

$$APT_i = \frac{\sum_{j=1}^{k} x_{ij}}{M_i} \tag{2}$$

P_i: the actual cost of instances renting for processing task.

$$P_i = \sum_{j=1}^{k} price_j * x_{ij} \tag{3}$$

RT_i: the requested response time.

$QukTime_i$: the shortest processing time of $task_i$;

$$QukTime_i = \frac{W_i}{M_i * pc_{InsQuk}} \tag{4}$$

$Budget_i$: the maximum cost that user can accept for processing the task.

$LesC_i$: the least cost for processing $task_i$;

$$LesC_i = \frac{price_{InsCh} * W_i}{pc_{InsCh}} \tag{5}$$

IV. UNITS SCHEDULING ALGORITHMS

The problem of scheduling a weighted directed acyclic graph to a set of homogeneous processors to minimize the completion time has been extensively studied. However, given cloud computing environment, resources are delivered as a standard interface and can be seem as infinite, so it's simple to solve the minimize time problem. At the same time, budget is a new constraint.

A. A single task

As a single task, dependency relationships do not need to be considered. Given a response time limitation RT_i, the target is to save money under the premise of finishing in time. The mathematical model shown in (6). Given a budget limitation $Budget_i$, the target is to save time without deficit. The mathematical model shown in (7).

It's a typical Integer linear programming (ILP) problem. Generally speaking, it will cost a long time to solve such problems. A popular way is to use heuristic algorithm, but here we use branch and bound method given actual m and k is small. For example, there are only three types in Amazon EC2 Spot Instances. We name this phase as step one.

$$\begin{aligned}
&\min imize: \\
&\sum_{i=1}^{n} x_i\, price_i \\
&S.T: \\
&0 \le x_i \le \frac{W}{pc_i} \\
&\sum_{i=1}^{n} x_i * pc_i \ge W \\
&\sum_{i=1}^{n} x_i \le M * RT
\end{aligned} \tag{6}$$

$$\begin{aligned}
&\min imize: \\
&\frac{\sum_{i=1}^{n} x_i}{M} \\
&subjectto: \\
&0 \le x_i \le \frac{W}{pc_i} \\
&\sum_{i=1}^{n} x_i * price_i \le Budget \\
&\sum_{i=1}^{n} x_i * pc_i \ge W
\end{aligned} \tag{7}$$

B. A sequence of task

Typically, a scientific workflow can be formulated as a DAG. Each task may be processed by different algorithm or application. Each vertex is a task and the weight is determined by the actual processing time of the task APT_i. Here we consider the task execution time only. First of all, we consider a special case that each vertex has one parent node and one child node at most. For each task can be started immediately after the execution of the parent node. All the vertexes are on the critical path. As the scientific workflow must be finished before the requested response time RT, the following is algorithm description:

```
Algorithm -1
  Input: RT;$pc_k$;$W_i$;$M_i$
  For each task $t_i$ do
    Update $QukTime_i$& $P_i$& $X_i$
    Input: RT;$pc_k$;$W_i$;$M_i$
  For each task $t_i$ do
    Update $QukTime_i$& $P_i$& $X_i$
    Update $APT_i$= $QukTime_i$
  End For
  Calculate TimeTotalleast = sum $QukTime_i$
            RestTime = RT-TimeTotalleast
  For each task $t_i$ do
    $RT_i$= $QukTime_i$+ RestTime
    Update $APT_i$ & $P_i$& $X_i$
    RestTime= RestTime-( $APT_i$-$QukTime_i$)
    If RestTime<=0
    Break;
  End For
  Ptotal=sum $p_i$
  Ttotal=sum $APT_i$
  End
```

As the budget is limited to Budget, the algorithm is:

```
Algorithm -2
  Input: Budget;$pc_k$;$W_i$;$M_i$
  For each task $t_i$ do
    Update $LesC_i$ & $APT_i$ & $X_i$
     Update $P_i$= $LesC_i$
  End for
  Calculate CostTotalleast = sum $LesC_i$
  RestBudget = Budget - CostTotalleast
  For each task $t_i$ do
    $Budget_i$ = $P_i$ + RestBudget
    Update $APT_i$ & $P_i$& $X_i$
    RestBudget = RestBudget -( $P_i$ - $LesC_i$)
    If RestBudget <=0
    Break;
  End For
  Ptotal=sum $p_i$
  Ttotal=sum $APT_i$
  End
```

C. DAG

As a common DAG, each vertex denotes a task and the weight denotes the task processing time. Given response time is limited, we present a schedule algorithm. First, assign the InsQuk instance to each task and calculate the actual processing time APT of each task as weight of vertex. Compute the EST and LST of each task and pick up a critical path. Assign new instances to task on the critical path using algorithm -1and update APT on the critical path, then update the EST and LST of each task. According to the topological sort, assign new instances to tasks using algorithm in step one with a response time limitation RT_i= LST_i- EST_i+ APT_i from back to front, then update LST of parent nodes.

```
Algorithm -3
  Input:RT;$W_1$,$W_2$…$W_k$;$M_1$,$M_2$..$M_k$
  For each task $t_i$ do
    Update $QukTime_i$& $P_i$& $X_i$
    Update $APT_i$= QukTimei
  End for
  For each task $t_i$ do
    Update $EST_i$ &$LST_i$
  End for
  Choose a critical path update $APT_i$& Pi& $X_i$ using Algorithm -1
  For each task $t_i$
  End for
  i=k
  While $_i$>0
   If $EST_i$ <> $LST_i$
    $RT_i$= $APT_i$ + $LST_i$- $EST_i$
    Update $APT_i$ & $P_i$& $X_i$
    Update $LST_i$
   End If
    i++;
  End While
  Ptotal=sum $p_i$
  Ttotal=sum $APT_i$
  End
```

The budget is limited to Budget. First, assign the InsCh instance to each task and calculate the actual processing time APT of each task as weight of vertex. The remaining money RM= Budget-sum pi. We can use RM to shorten an actual processing time of task. We assign new instances to task that on the critical path with maximum weight using algorithm in step one and update EST and LST. Next, choose the task that on the critical path with maximum weight except for those have been scheduled until RM is used up.

```
Algorithm -4
  Input:RT;W1,W2…Wk;M1,M2..Mk
  For each task ti
    Update LesCi & APTi & Xi
    Update Pi = LesCi
  End for
  RM= Budget-(sum pi)
  For each task ti
    Update ESTi &LSTi
  End For
  Scheduled={}
```

```
 While RM>0
   Index=0;max=0;
   For each task ti
     If ESTi ==LSTi&ti Scheduled
       If APTi>max
         Index=I;max= APTi
       End If
     End If
   End For
   Schedule taskindex using algorithm in step one
   Update RM
 End While
```

V. Experimental Results

The tasks in the workflow can be data analysis, image processing, video encoding or other. Here we use a video encoding application to show the processing capability of different type of instance. The Java Audio Video Encoder (JAVE) library is Java wrapper on the ffmpeg project. Developers can take advantage of JAVE to transcode audio and video files from a format to another. It can separate and transcode audio and video tracks, resize videos, changing their sizes and proportions and so on. We use it transcode a MGP video with the size of 9.66MB into FLV format under different instance types. We define such workload as 1 unit. As most cloud providers we adopt hour as a time interval. If any instance usage is less one hour, it would be still charged as one hour. We assume that any migration does not cause any cost. Table 1 and table 2 show the attributes of three instance types.

We have 100GB of data that equivalent to 100GB/9.66MB= 102400MB/9.66MB =106000 unit workload and assume the maximum instance number is 5. Fig. 1 shows the response time and cost using different instances separately.

Fig. 2 shows the cost with the request response time in step one. The cost will be reducing with the response time increasing. Fig. 3 shows the different type of instances number requested with different response time.

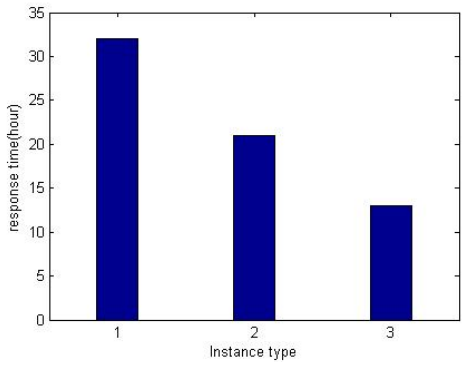
TABLE I
THE CONFIGURATIONS OF THREE TYPES OF INSTANCE

Instance type	CPU	Memory	Price
1	1GHZ*1	1GB	4cent
2	1GHZ*2	2GB	8cent
3	1GHZ*4	4GB	16cent

TABLE II
THE PROCESSING CAPABILITY OF THREE TYPES OF INSTANCE

Instance type	Time Spend on 1 Unit Workload	Standard Deviation	Processing Capability
1	5397ms	257ms	667uw
2	3420ms	131ms	1052uw
3	2100ms	300ms	1714uw

ms = millisecond, uw = unit workload



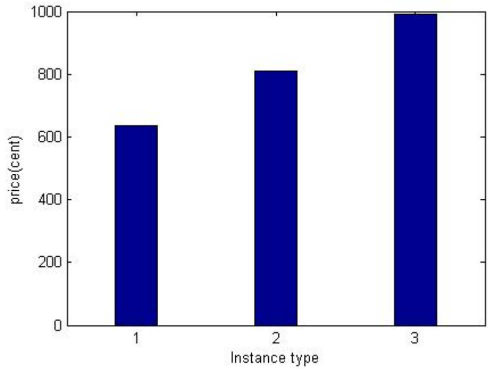


Fig. 1. Response time and cost using different instance separately

Fig. 4 shows the processing time with the limited budget in step one. The processing time will be reducing with the budget increasing. Fig. 5 shows the different types instances number requested with different budget.

These data illustrate the relationship between processing time and cost. We can reduce response time by increasing budget, or vice versa.

We describe a scientific workflow as Fig. 6. It composed of invoke activities, sequence activities and flow activity. Tasks have a random workload between 50GB and 300GB and a random M between 2 and 5. Tasks attributes as table III.

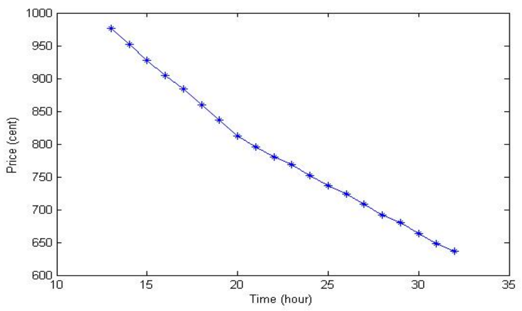


Fig. 2. The processing time with the limited budget

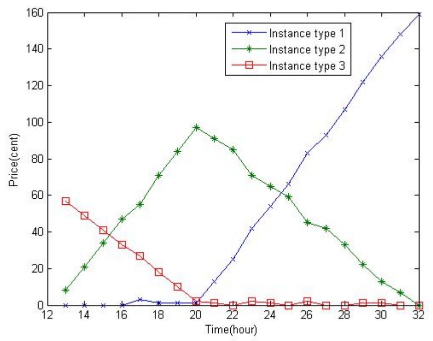


Fig. 3. The different type of instances number requested with different response time

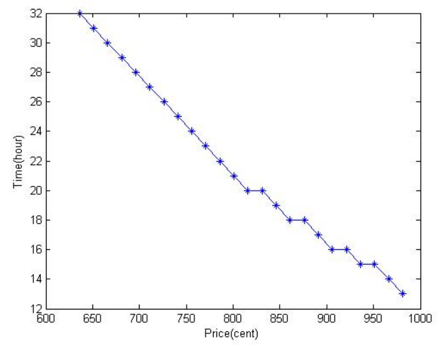


Fig. 4. The processing time with the limited budget

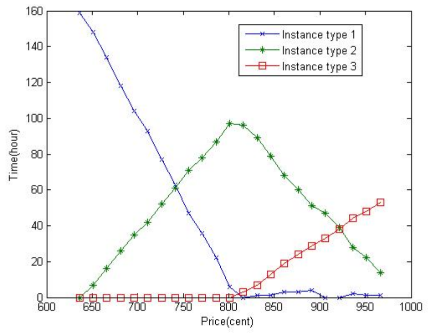


Fig. 5. The different type instances number requested with different budget

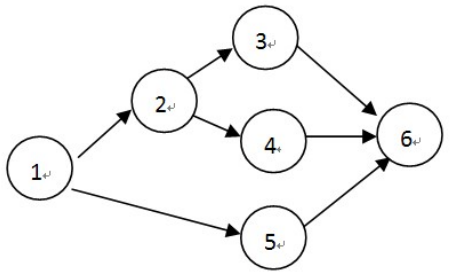


Fig. 6. Workflow directed acyclic graph

It can be concluded that user can improve the response time by increase budget or save money by waiting a longer time though these algorithms in scientific workflow.

A TTRIBUTES OF TASKS

Task number	Workload	M
1	200	3
2	150	2
3	150	5
4	250	5
5	300	4
6	50	3

Fig. 7 shows the cost with the request response time. Fig. 8 shows the processing time with the limited budget.

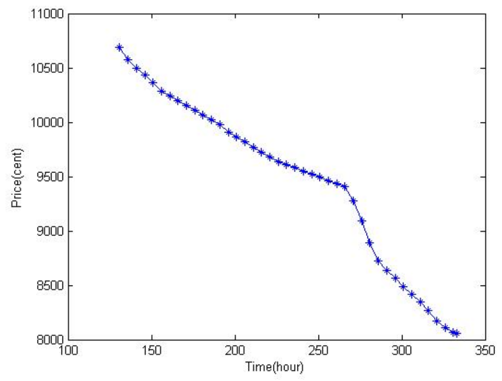


Fig. 7. The cost with the request response time

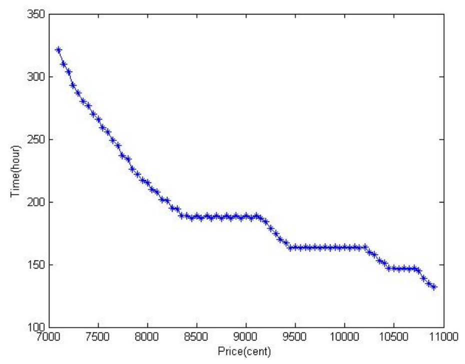


Fig. 8. The processing time with the limited budget

VI. CONCLUSIONS AND PERSPECTIVES

In this paper, we investigated the scientific workflow scheduling schema in cloud computing environment. Our simulated experimental results showed the schedule algorithm reducing the cost with a response time limitation and reducing the processing time with a budget limitation effectively in scientific workflow.

Our future work will concern the following problem: time complexity of algorithm in one task is relatively high. It will be cost a long time given more types of instance. We will solve such problem with heuristic algorithm.

REFERENCES

[1] C.S. Yeo and R. Buyya,"Pricing for utility-driven resource management and allocation in clusters," *International Journal of High Performance Computing Applications*, vol.21, no.4, pp. 405-418, 2007.

[2] Z. Peng and Y. Zheng, " A QoS-aware system for mobile cloud computing," *Conference on Cloud Computing and Intelligence Systems,* Beijing, 2011, pp 518-522.

[3] Triana, Available: http: //www.trianacode.org/index.html

[4] Taverna, Available: http: //www.taverna.org.uk/

[5] GridFlow, Available: http: //gridflow.ca/

[6] Java Cog Kit, Available: http: //www.globus.org/cog/java/

[7] Kepler, Available: https: //kepler-project.org/

[8] W. Shi and W. M. Zheng, "The balanced dynamic critical path scheduling algorithm of Dependent Task Graphs," *Chinese J. Computers*, vol. 24, no. 9, pp. 992-997, 2001.

[9] Y. K. Kwok and I. Ahmad, "Benchmarking the task graph scheduling algorithms," In *Proc. 12th International Parallel Processing Symposium*, Orlando, Florida, USA, 1998, pp. 531-537

[10] Java Audio Video Encoder, Available: http: //www.sauronsoftware.it/ projects/ja-ve/manual.php.