2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing

Execution Time Prediction Using Rough Set Theory in Hybrid Cloud

Chih-Tien Fan
Dept. of Comp. Sci.,
National Chiao Tung University,
1001, University Road,
Hsinchu, 300, Taiwan,
R.O.C.
s10086024.cs00g@nctu.edu.tw

Yue-Shan Chang
Dept. of CSIE,
National Taipei University
151, University Road,
Sanhsia Distinct, New Taipei City, 237, Taiwan,
R.O.C.
ysc@mail.ntpu.edu.tw

Wei-Jen Wang,
Dept. of CSIE,
National Central University
300, Jhongda Rd., Jhongli City, Taoyuan County 320,
Taiwan, R.O.C.
wjwang@csie.ncu.edu.tw

Shyan-Ming Yuan
Dept. of Comp. Sci.,
National Chiao Tung University
1001, University Road,
Hsinchu, 300, Taiwan,
R.O.C.
smyuan@gmail.com

Abstract* – Execution time prediction is an important issue in cloud computing. Predicting the execution time fast and accurately not only can help users to schedule jobs smarter, but also maximize the throughput and minimize the resource consumption of cloud platform. While hybrid cloud provides methods to federate multiple cloud platforms, different cloud platforms have different resource attributes, which will increase the difficulties to predict a job's execution time. In this paper, we exploit Rough Set Theory (RST), which is a well-known prediction technique that uses the historical data, to predict the execution time of jobs. The evaluation presents that RST can utilize the accuracy of the execution time, while the decision can be made in a short period of time.

***Keywords**: Execution Time Prediction, Rough Set Theory, Rough Sets, History Based Approach, Hybrid Cloud, Public Cloud, Private Cloud.*

I. INTRODUCTION

In the past decades, Cloud computing [3] has become a hot research field. Clouds can be roughly classified into three types: public, private, and hybrid. Public cloud is a computing resource that allows multiple users to run their jobs, while the resource providers charge as much as each user consume their resource. Private cloud is a private computing resource that only limited users can run their applications on it, while the owner needs to maintain the physical machines and software themselves. The hybrid cloud [1] allows private cloud coexists and federates with public clouds. The private cloud provides static resource while the public cloud provides the resource as a transparent zone.

Public cloud, such as Amazon EC2 [4], uses *instance* to manage and charge to their costumers. Each instance is assigned to have difference attributes (E.g., the CPU type, memory size, etc,). The instance types and their attributes are easy to be obtained from the providers, hence users can choose different types of instance according to their application needs.

In private cloud, users can utilized their computing resource by using *Virtualization Technology* [7-9], or sometimes named *Virtual Machine* (VM). VM is controlled by VM Manager (VMM or called Hypervisor) and it provides several APIs that allow users to control a VM. VMM, such as Xen [9] or Entropy [7], also provides APIs that allow users to customize each VM's attributes.

Scheduling (or task assignment) is also an important issue in cloud computing. Its purpose is to ensure all jobs can run smoothly while all the physical computing resources are utilized. Common known Scheduling methods for distributed system are Round-Robin Scheduling (RRS), Random Scheduling (RS), Size-Based Scheduling (SBS), and Dynamic Scheduling (DS) [12]. For RRS and RS, all jobs are treated as the same entity. In other words, no other informations about a job is needed before scheduler is activated; except that the RRS will assign the job in a cyclic fashion, and the RS will randomly select instance to perform the job. In the contract, SBS and DS are job-attribute-based schedulers. In SBS, a job is assigned according to its size (which is defined by user). Every job's size within the specific range will be assigned to a specific instance. In DS, a job is assigned to the instance that has the least amount of outstanding work left to do.

In job-attribute-based scheduler, the execution time of a job is one of the attribute that can be used [10, 11]. By using the execution time attribute, the scheduler can tell users when the job will be executed and terminated. Especially in the cloud environment, user can adjust the need of the resource according to it.

But predicting execution time accurately is not easy, especially in the hybrid cloud environment. In a simple cloud (or cluster), the owner will buy large amount of machines (usually with the same specifications) to reduce the cost. Users can build an predictor to predict execution time just based on the job's attributes; there is no hardware-related attributes need to be concerned. But in hybrid cloud, thing becomes complicated. Difference cloud may have difference types of instances that can be chosen from. Also, cloud provides methods that the user can provision the resources according to their needs. Building predictors for each cloud environment is time consuming and inefficient. Therefore, the attributes of the instance should be also taken into accounts.

In this paper, we utilized the Rough Set Theory (RST) to predict a job's execution time in the hybrid cloud cloud environment. RST is a well-known prediction technique that uses the historical data to predict the attribute value of an object. We propose an execution time prediction algorithm based on RST to schedule jobs in hybrid cloud environment. The evaluation can show that the RST can be utilized to accurately predict the execution time increasingly.

The reminder of the paper is organized as follows. Section 2 will briefly introduce our previous work. The Rough Set Theory as well as the algorithm will be introduced in Section 3. Section 4 shows the implementations of the prototype and explanations of the implementations. Related work is shown in Section 5. Section 6 will be the conclusions and future work.

II. Agent-based Service Migration Framework

Our previous work [1] uses the agent platform system to realize the hybrid cloud prototype model. It allows jobs migrating between different clouds. Methods of scheduling jobs in cloud environments and adjusting computing resource is also presented in this framework. The system model is shown as Figure 1.

The framework has 6 main components. Each component is described as follows.

- **System Monitoring Agent (SyMA)** – The agent collects the information of the cloud environment by receiving the *heartbeat* sent periodically by the *CAA*s. After receiving the message, the updated information is being stored to the database in the policy, which allows other agents to access.

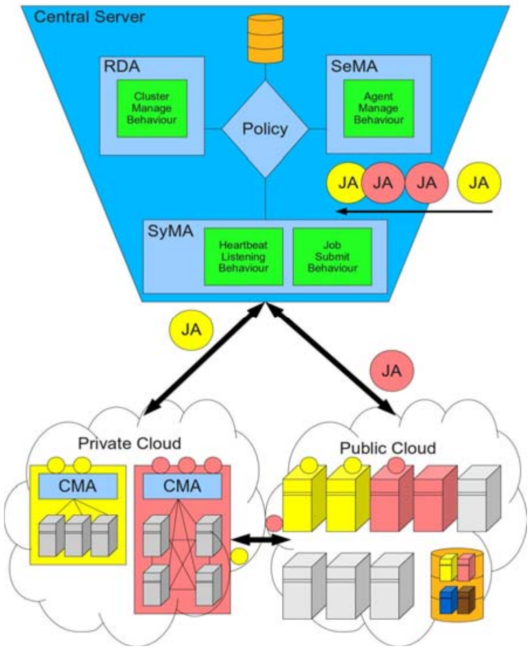


Figure 1: System model of Agent-based Service Migration Framework.

- **Service Migration Agent (SeMA)** – The *RDA* manages the *JA*s' location. When the new job is submitted, the *SeMA* will assign a location in the cloud that allows the job to be executed on. Also, if some cluster is overloading, *SeMA* will notify some *JA*s on the overloaded cluster to migrate to some other cluster, to balance the load.
- **Reconfiguration Decision Agent (RDA)** – The *RDA* is used to reconfigure and adjust the cloud environment. If the whole system is overloaded, new cluster can be initiated on the public cloud to balance the system. On the other hand, If the loading of the system is not heavy, some clusters can be turned off to save energy consumption.
- **Policy** – The policy is an abstract type. It allows users to define their own policy on 1) how the job agent is being dispatched or managed, and 2) how the public cloud resource is being dynamically adjusted. The *SeMA* and *RDA* will periodically check the policy. The policy also simplify the way how the *SyMA*, *SeMA*, and *RDA* to access the system database, which have the information of the cloud status. The policy also maintains a global queue in a FCFS fashion.
- **Cluster Management Agent (CMA)** – Each Cluster (or computing resource instance) is accompanied with one *CMA*. The *CMA* has 2 functions. One is scheduling the jobs locally in a FCFS fashion, so that there is only one job is executing on the cluster. The other function of the *CMA* is reporting the status of the cluster. Each *JA* will send *heartbeats* of their current status periodically to *CMA*. *CMA* will collect the information and send it via *heartbeats* to *SeMA*.
- **Job Agent (JA)** – The *JA* is an agent that will encapsulate a job, the job can be migrated along with the JA. *JA* also executes and monitors the job on the cluster. The *JA* also report the job status to the *CMA* periodically. After the job is terminated, *JA* can bring the results back to the private cloud.

III. RST-based Prediction

Rough Set Theory (RST) was first introduced by Pawlak in the early 1980s [2]. It has been used in many research area, such as data mining [5] and knowledge discovery [6]. Also, it provide methods for selecting needed features, feature extraction, data reduction, decision rule generation, and pattern extraction. The main benefit of using RST is that no preliminary or additional information is needed about the data (e.g., probability distribution or statistics about the data). In this paper, RST is used to predict a job's execution time on different types of instances in hybrid cloud.

A. RST Learning Procedure

Record No.	Processor Speed (GHz)	Memory Size (GB)	Input Data Size (MB)	Execution Time (sec.)
1	2	1	6	300
2	3	2	2	66
3	2	1	6	300
4	2	2	6	300
5	1	1	8	800
6	1	1	4	400
7	3	2	2	66
8	1	1	8	800

Figure 2: Example of Information Table.

Record No.	Processor Speed (a_1)	Memory Size (a_2)	Input Data Size (a_3)	Execution Time (d)
1	2	1	3	2
2	3	2	2	1
3	2	1	3	2
4	2	2	3	2
5	1	1	4	4
6	1	1	2	2
7	3	2	1	1
8	1	1	4	4

Figure 3: Information table after Discretizing.

RST makes predictions based on an *information table*; an example is shown as Figure 2. The table is consists of *records* that can be obtains from previous execution. Each record is consists of some *attributes* that characterize the record. Attributes can be categorized into two groups: *Condition Attributes* and *Decision Attributes*. Condition Attributes are attributes that can be measure beforehand (e.g., executing environments). On the other hand, *Decision Attributes* can be obtained only after it finished (e.g., execution time of a program), it is also the attribute that we want to predict. In Figure 2, Processor speed, memory size, and input data size are condition attributes; execution time of each record is decision attribute.

According to the applications of RST [2, 5, 6], the learning procedure can be summarized as five steps. For more detailed information of RST, please refer to [2, 6].

1. First, all the attributes, including condition attributes and decision attributes, should be defined. After that, historical data should be obtained to form an information table. Figure 2 is the example of the information table. The *Processor Speed (a_1)*, *Memory Size (a_2)*, and *Input Data Size (a_3)* are Condition Attributes. The *Execution Time (d)* is the decision attribute.
2. To make the RST prediction stronger, discretizing each record should be used to eliminate the noise. Example after discretizing is shown as Figure 3. The a_1 is discretized into 3 groups with the width of 1 GHz. The a_2 is discretized into 2 groups with the width of 1 GB. The a_3 is discretized into 4 groups with the width of 2 MB. The d is discretized into 4 groups with the width of 200 seconds.
3. *D-Reduct* is being computed. *D-Reduct* is the subset of condition attributes that shows the essential part of the information system. By using *D-Reduct*, all objects in the information system is discernible. A discernibility function can be used to evaluate the *D-Reduct* via the discernibility matrix. The discernibility matrix of the example is shown as Figure 4. In each element, m_{ij}, of the matrix, the differences between record i and j are shown. For example, there have no discernibility between record 1, 3, and 7 because they have the same decision value. On the other hand, record 1 and 5 has the different execution time and the attributes that can discern them are a_1 and a_3. After that, a simplification on the formulation can be made by Boolean algebra with the absorption rule. The formula is written as (1)

#	1	2	3	4	5	6	7	8
1	-							
2	$a_1a_2a_3$	-						
3	-	$a_1a_2a_3$	-					
4	-	a_1a_3	-	-				
5	a_1a_3	$a_1a_2a_3$	a_1a_3	$a_1a_2a_3$	-			
6	-	a_1a_2	-	-	a_3	-		
7	$a_1a_2a_3$	-	$a_1a_2a_3$	a_1a_3	$a_1a_2a_3$	$a_1a_2a_3$	-	
8	a_1a_3	$a_1a_2a_3$	a_1a_3	$a_1a_2a_3$	-	a_3	$a_1a_2a_3$	-

Figure 4: The discernibility matrix.

Record No.	Processor Speed (a_1)	Input Data Size (a_3)	Execution Time (d)
1	2	-	2
2	3	-	1
3	2	-	2
4	2	-	2
5	-	4	4
6	-	2	2
7	3	-	1
8	-	4	4

Figure 5: The decision table.

$$\begin{aligned} f_A(D)= &\\ &(a_1+a_2+a_3)(a_1+a_3)(a_1+a_2+a_3)(a_1+a_3)\\ &(a_1+a_2+a_3)(a_1+a_3)(a_1+a_2+a_3)(a_1+a_2)(a_1+a_2+a_3)\\ &(a_1+a_3)(a_1+a_2+a_3)(a_1+a_3)\\ &(a_1+a_2+a_3)(a_1+a_3)(a_1+a_2+a_3)\\ &(a_3)(a_1+a_2+a_3)\\ &(a_1+a_2+a_3)(a_3)\\ &(a_1+a_2+a_3)\\ =&(a_1+a_2+a_3)(a_1+a_2)(a_1+a_3)(a_3)\\ =&a_1a_3+a_2a_3 \end{aligned} \tag{1}$$

The result of (1) shows that either $\{a_1, a_3\}$ or $\{a_2, a_3\}$ can be the *D-Reduct* of the example. In the following example, $\{a_1, a_3\}$ is used as the *D-Reduct* attributes.

4. The last step is to generate the decision rules, which can eliminate unnecessary values of condition attributes in the decision table. The decision rules can be evaluated by using the relative discernibility function, which is similar to the discernibility function in procedure 3. For example, the relative discernibility function of record 1 is written as (2), which shows that either a_1 or a_3 can be eliminated. The final decision table is shown as Figure 5.

$$f_1(D) = (a_1+a_3)(a_1+a_3)(a_1+a_3)(a_1+a_3) = a_1+a_3 \quad (2)$$

B. RST-based Prediction Procedure

After the decision rule is generated, the prediction can be started. The summarized prediction procedure is described as followed.

1. First, obtain the attributes of the object that is going to predict. Discretize the attributes that belongs to *D-Reduct* as step 2 in the learning procedure section. For example, a new coming job *e* is going to be executed on a processor speed of 3 GHz, with memory size 1 GB and with an input data size of 8 MB. We can discretize them to {3, 1, 4}, and the first and the third value are used to predict.
2. Find the records from the decision table that is similar to the new job by comparing the discretized value. In this example, the record 2, 5, 7, and 8 are selected because they are similar to *e*.
3. Calculate the prediction result. If some past jobs are being selected to be similar to object *e*, then the result would be the average of the past jobs' decision value. In this example, we will predict that the *e* will execute (66+800+66+800)/4=433 seconds. If no similar jobs are selected in previous step, it means that the job *e* is the new type of job to the information system. Other approach should be used to do the prediction. Also, the result of this job should be feedback to the system.

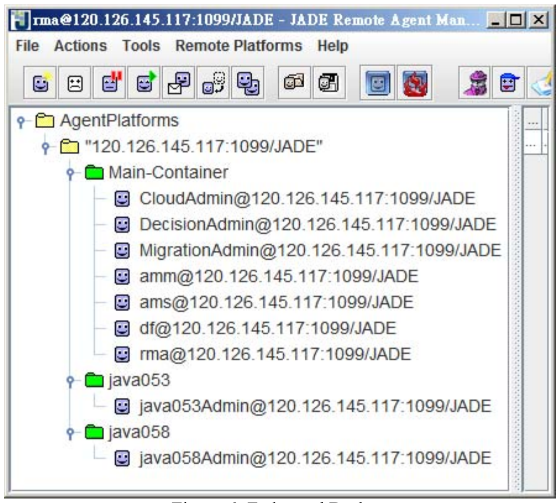
IV. Evaluation

We prototype the system using the agent platform JADE v4.0 based on our previous work described in Section 2 and [1]. The RST algorithm is implemented in policy component, and the Service Migration Agent (SeMA) will perform the task of predicting the execution time of a job. The condition attributes recorded by the system are: *the name of the application*, *the input data size*, *the number of the CPU*, *CPU clock rate*, and *the memory of the VM.* Currently all VMs are set up with the same amount of resources, that is, only the name of the application and the input data size changes during the evaluation. More variety of VMs will be put into future work. Figure 6 shows the prototyping of the system using JADE platform with some running VMs.

Figure 7 is the screen shot of 4 jobs running on four difference instances. Each green folder is named as *container* in Jade, while each instance form a container. Within the instance, CMAs help the system monitoring and controls the JAs on each instance.

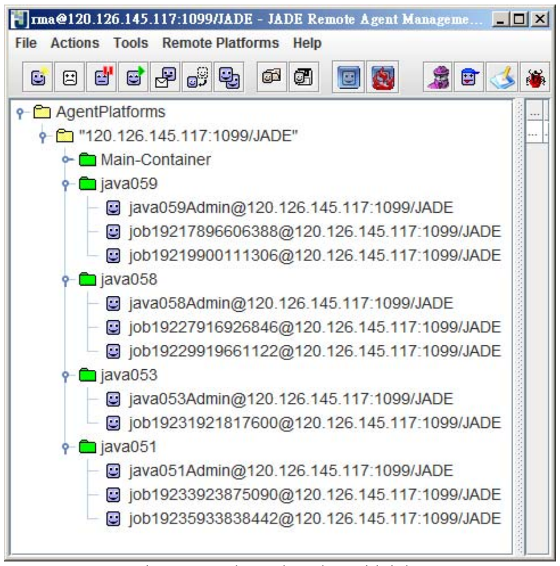
Currently, two jobs are submitted to the system. The jobs' detailed information is described as follows.

- **Compute π** – π is the ratio of a circle's circumference to its diameter, and it is an irrational number. The algorithm first draws a quarter circle inside a square. Next, with a given input, *n*, *(n+1)²* points are spread equally onto the square. By calculating the ratio between the points inside the quarter circle and the square, approximation of π/4 can be generated, thus the number of π can be computed. The job's time complexity is $O(n^2)$, where *n* is treated as the size of the job.
- **Area Approximation** – This program evaluates the lower approximation of the area between an equation and the x-axis within the interval, that is, $\lim_{a \to n} \sum_{x=0}^{a-1} \frac{1}{a} f(x)$. *n* is given by the user, while $f(x)$ is an decision equation. All calculation is double precision. The job's time complexity is O(n), where *n* is treated as the size of the job.



```
rma@120.126.145.117:1099/JADE - JADE Remote Agent Man...
File  Actions  Tools  Remote Platforms  Help
AgentPlatforms
  "120.126.145.117:1099/JADE"
    Main-Container
      CloudAdmin@120.126.145.117:1099/JADE
      DecisionAdmin@120.126.145.117:1099/JADE
      MigrationAdmin@120.126.145.117:1099/JADE
      amm@120.126.145.117:1099/JADE
      ams@120.126.145.117:1099/JADE
      df@120.126.145.117:1099/JADE
      rma@120.126.145.117:1099/JADE
    java053
      java053Admin@120.126.145.117:1099/JADE
    java058
      java058Admin@120.126.145.117:1099/JADE
```

Figure 6: Federated Broker.



```
rma@120.126.145.117:1099/JADE - JADE Remote Agent Manageme...
File  Actions  Tools  Remote Platforms  Help
AgentPlatforms
  "120.126.145.117:1099/JADE"
    Main-Container
    java059
      java059Admin@120.126.145.117:1099/JADE
      job19217896606388@120.126.145.117:1099/JADE
      job19219900111306@120.126.145.117:1099/JADE
    java058
      java058Admin@120.126.145.117:1099/JADE
      job19227916926846@120.126.145.117:1099/JADE
      job19229919661122@120.126.145.117:1099/JADE
    java053
      java053Admin@120.126.145.117:1099/JADE
      job19231921817600@120.126.145.117:1099/JADE
    java051
      java051Admin@120.126.145.117:1099/JADE
      job19233923875090@120.126.145.117:1099/JADE
      job19235933838442@120.126.145.117:1099/JADE
```

Figure 7: Federated Broker with jobs.

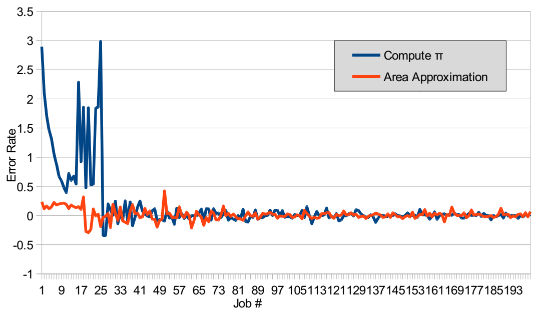


Figure 8: The Error Rate

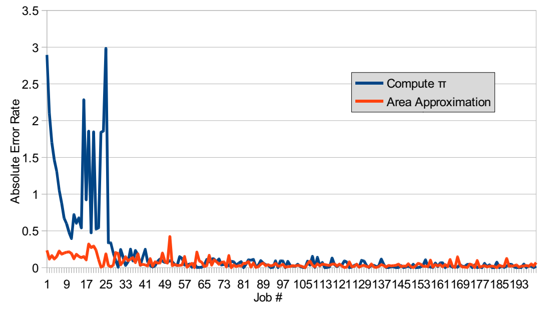


Figure 9, The Absolute Error Rate

A. *Execution Time Error Rate*

The Error Rate of the system is shown as (3).

$$\varepsilon = \frac{p - \tau}{\tau} \qquad (3)$$

where τ is the exact execution time, and p is the predicted execution time. The more the error rate ε is close to 0, the more accuracy the system is. Figure 8 is the error rate of 200 jobs.

The error rate can shows that how much relative bias does the predictor made to the actual execution time. If the error rate is positive, it means that the predictor is over predicted; in other words, the error is negative shows that it is under predicted. As show in Figure 8, the error rate of the first few jobs is relative high. It is due to the lack of the historical data that can be used to predict the job. When a job cannot be predicted, a static execution time value is applied to it. After the execution has terminated, each job's execution result will be fed back to RST as a record of the information table immediately. The more the historical data are stored, the more accurate the prediction will be.

Figure 9 shows the Absolute Error Rate. The absolute error rate shows how much improvement has the prediction made. The higher the absolute error is, the more improvement is needed. The average absolute error rates for 2 kinds of jobs with 200 submissions are 0.2008 and 0.0615. The total average values seem to be large, but if the data are binned with the width of 25 jobs, the accuracy is very impressive. As shown in Figure 10, group 1 (the first 25 jobs) has the high average values of 1.2494 and 0.7123, which is due to insufficient of historical data. On the other hand, group 8 (the last 25 jobs) has an impressive average values of 0.0178 and 0.0296.

Figure 9 and 10 also depicts one fact: It is not wise

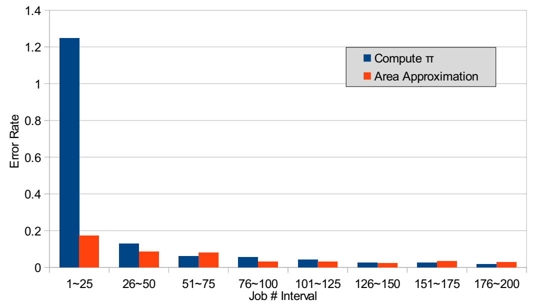


Figure 10: Average Absolute Error Rate that splits every 25 jobs

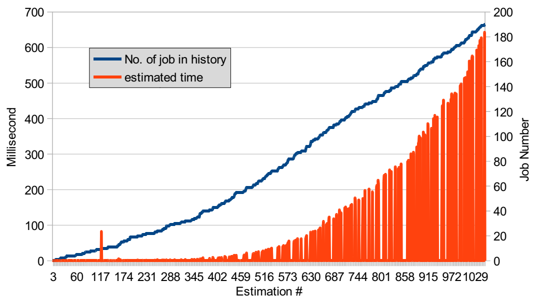


Figure 11: Prediction Time

to build one predictor for one cloud environment. Building one predictor needs to take 25 to 50 past jobs to achieve an error rate under 0.1. Users should build one global predictor that can be used in hybrid cloud environment, other cloud's result can be applied to the new clouds.

B. *Time Taken to Predict a Job*

Figure 11 is the prediction time of each iteration and the historical job used to predict the job. Every job may be predicted more than once, which is due to our Hybrid Cloud Framework described in Section 2. Our framework will not dispatch the job if the resource in hybrid cloud is not enough. If more computing resource is available, the computing resource will be added to hybrid cloud; on the contrary, the job will be kept in the queue until there is enough resource.

As each record is added into RST predictor, re-evaluation is needed to update the decision table. It is easy to be noticed that the more the history records are in the information table, the more time is needed to predict a job. The largest prediction time is 642.91 *ms* with 190 jobs, which is acceptable. When there is no new record to be updated, the prediction time taken can be less than 1 *ms*. It is easy to be noticed that the generating the decision rule needs much more time than just predicting the value. To reduce the time of predicting, periodically updating the decision rules can be considered. As shown in Figure 9 and 10, the error rate has been significantly reduced after the 50th of the job. Users can select a static period of time to update the decision rule, so that the total predicting time can be reduced, while the error rate is low.

V. Related Work

Execution time prediction has been a big research area [10, 11, 13–15]. Job prediction helps the users to know when the job will finished and how the jobs can be scheduled. The accuracy and the time needed to predict a job are important. Accuracy helps better scheduling decision while the time taken to predict helps making decision faster.

The work done by Selvi *et al*. [13] is very similar to our research problem. The authors use the RST to estimate the job's execution time in grid environment. The missing values of a job's requirements are indicated that it can affect the data analysis, and it is solved by using RST to find the common records. But the author did not mention the job's properties in the implementation, as well as the attributes used in prediction. In our work, the attributes of a job is predefined and is easily to be obtained.

Norinha Nassif *et al*. proposed a PredCase [14], which is a Case-Based Reasoning (CBR) paradigm, to predict the completion time of a job. The system tries to find the similarity between the new job and past jobs with the job attributes by performing four main steps: Retrieve, Reuse, Revise, and Retain. The main concept of estimating job is similar, but using different strategy compared to RST. The author did not mention how much time is needed to estimate a job and how to manipulate the job when there is no similar job.

Kiran *et al*. suggested an execution time prediction model [15] by using a combination of static analysis, analytical benchmarking, and compiler-based approach. The job is written as script, and each submitted job go through four layers, Application Selector, File Parser, Code Evaluation Engine, and Predictor Engine. Although the accuracy is very remarkable, but there exist risks of transferring codes between hybrid clouds. Instead of sending script of codes, in our framework, the compiled code is transferred, and the statistical analysis is performed by RST.

VI. Conclusion

In this paper, we utilized the RST to predict the execution time in hybrid cloud. The predictor was embedded into our previous framework, and worked smoothly in the hybrid cloud environment. The implementation result shows that RST-based predictor can predict the execution time of a job with error rate under 0.1 when the number of historical job is over 50. When more records available, the error rate can drop under 0.03. The time taken to predict a job is also remarkable, less than 1 second with 190 historical records to perform a full prediction. The system can aid users to schedule their jobs faster and more accurate.

In future work, we will add more variety of VMs and jobs to test the system.

Acknowledgment

This work was partially supported by the National Science Council of Republic of China under Grant NO. 100-2221-E-305-013.

Reference

[1] Fan, C.T.; Wang, W.J.; Chang, Y.S., "Agent-based Service Migration Framework in Hybrid Cloud," 2011 IEEE 13th International Conference on High Performance Computing and Communications (HPCC), pp. 887-892.

[2] Pawlak, Z.; Grazymala-Busse, J.; Slowinski, R.; Ziarko, W., "Rough Sets," Communications of the ACM, Vol. 38, Issue 11, Nov. 1995, pp. 88-95.

[3] Pallis, G., "Cloud Computing: The New Frontier of Internet Computing," IEEE Internet Computing, Vol. 14, No. 5, 2010, pp. 70-73.

[4] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/

[5] Krishnaswamy, S.; Loke, S.W.; Zaslavsky, A., "A hybrid model for improving response time in distributed data mining," IEEE Transactions on System, Man, and Cybernetics, Part B: Cybernetics, Vol. 34, Iss. 6, 2004, pp. 2466-2479.

[6] Hu, X., "Knowledge discovery in databases: an attribute-oriented rough set approach," ph.D thesis, Regina University, 1995.

[7] Hermenier, F.; Lorca, X.; Menaud, J.M.; Muller, G.; and J. Lawall, "Entropy: a Consolidation Manager for Clusters," VEE'09, pp. 41-50

[8] Yazir, Y.O.; Matthews, C.; Farahbod, R.; Neville, S.; Guitouni, A.; Ganti, S.; and Coady, Y., "Dynamic Resource Allocation in Computing Cloud Using Distributed Multiple Criteria Decision Analysis," 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 91-98.

[9] "Welcome to xen.org, home of Xen® hypervisor, the powerful open source industry standard for virtualization.," http://www.xen.org

[10] A.B. Downey, "Predicting queue times on space-sharing parallel computers," Parallel Processing Symposium, 1997. Proceedings., 11th International, pp. 209-218.

[11] Smith, W.; Taylor, V.; and I. Foster, "Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance," Job Scheduling Strategies for Parallel Processing, 1999, pp. 202-219.

[12] Harchol-Balter, M.; Crovella, M.; and C.D. Murta, "On Choosing a Task Assignment Policy for a Distributed Server System," Journal of Parallel and Distributed Computer, Vol. 59, Iss. 2, Nov. 1999, pp. 204-228.

[13] Selvi, S.T.; Kumari, M.S.S.; Prabavathi, K.; and G. Kannan, "Estimating Job Execution time and Handling Missing Job Requirements Using Rough Set in Grid Scheduling," International Conference on Computer Design and Applications (ICCDA), 2010, Vol. 4, pp. 295-299.

[14] Noronha Nassif, L.; Marcos Nogueira, J.; Vinicius de Andrade, F.; Ahmed, M.; Karmouch, A.; Impey, R., "Job Completion Prediction in Grid Using Distributed Case-based Reasoning," The 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005, pp. 249-254.

[15] Kiran, M.; Hashim, A.H.A.; Kuan, L.M.;Jiun, Y.Y., "Execution Time Prediction of Imperative Paradigm Tasks for Grid Scheduling Optimization," International Journal of Computer Science and Network Security (IJCSNS), Vol. 9, No. 2, Feb 2009, pp. 155-163.