Reduct Algorithm Based Execution Times Prediction in Knowledge Discovery Cloud Computing Environment

Kun Gao, Qin Wang, and Lifeng Xi
Computer and Information Technology College, Zhejiang Wanli University, China

Abstract: *Cloud environment is a complex system which includes the matching between computation resources and data resources. Efficient predicting services execution time is a key component of successful tasks scheduling and resource allocation in cloud computing environment. In this paper, we propose a framework for supporting knowledge discovery application running in cloud environment as well as a holistic approach to predict the application execution times. We use rough sets theory to determine a reduct and then compute the execution time prediction. The spirit of the algorithm, which we proposed comes from the number of attributes within a given discernibility matrix. We also propose to join dynamic data related to the performances of various knowledge discovery services in the cloud computing environment for supporting the prediction. This information can be joined as additional metadata stored in cloud environment. Experimental result verifies that the proposed algorithm in this paper supply a general solution for the problem of web service execution time prediction in cloud environment.*

Keywords: *Distributed computing, cloud computing, knowledge discovery, rough set.*

Received April 5, 2012; accepted January 17, 2013; published online April 4, 2013

1. Introduction

Cloud computing is an Internet based super computing model. A cloud computing environment can be made up of tens of thousands of computer nodes [15]. Therefore, cloud computing has powerful computational ability to simulate nuclear explosions, predict climate change and market trend so much so that make you experience the computing power with 10 trillion times per second. According to the requirement of clients, they can employ the computational resource in cloud environment by means of computers, laptops, cell phones and so on [8].

The network which provides resources is called as "Cloud Environment". The narrow concept of cloud computing refers to the delivery and usage patterns of Information System (IS) infrastructure. From the user perspective, the resources in Cloud Environment could be scalable infinitely, accessible anytime anywhere, used on demand whenever and charged according to the amount of use [14]. The broad concept of cloud computing refers to the delivery and usage of services. This service can be the Information Technology, software, applications based on Internet and other services [7].

Cloud computing is the joint development result of distributed computing, parallel computing and grid computing [2, 4]. We can also consider that cloud computing is the commercial implementation of these computer science concepts. Basic principles of cloud computing is that the computation is spread on a large number of the distributed nodes rather than the local computer or just only the remote central server. Cloud computing allows enterprises allocate the resource to the requisite applications and access the services and storage system according to its need [12]. This is a revolutionary change; it's like that the old single generator model change to a centralized power supply mode in power plant. It means that the computing power can also be used as a commodity circulation, like gas, electricity and water, accessible conveniently and cost lowly. The main difference is that cloud application is transmitted via the internet. Cloud computing blueprint is already apparent: in the future, only need a laptop or a mobile phone, everything what we need to do can be done via Internet including supercomputing applications. From this perspective, the end user is the real owner of cloud computing [11].

Cloud environment is an efficient solution for both computing intensive and data intensive applications, so it is a natural platform for deploying a high performance knowledge discovery application. In this paper, we propose a distributed knowledge discovery cloud environment called as Knowledge Discovery Cloud environment (KDC), which specially execute knowledge discovery application in distributed environment.

A key issue of scheduling knowledge discovery applications and resource allocation is the ability to accurately predict the execution times of an

application. Such technique can improve the performance of scheduling algorithms and help predict queue times in cloud computing environments. An efficient cloud computing environment is a complex system which includes the matching between computation resources and data resources. For example, the KDC environment provides a specialized broker for distributed knowledge discovery computation: given a knowledge discovery service in cloud environment, the broker will perform resource allocation and tasks scheduling, construct actual implementation strategy and return the outcomes to the client. The implementation plan must be developed to meet certain conditions, such as cost efficient, preference, cloud environment algorithm and so on. Once the implementation plan is complete, it will be delivered to cloud resource management service to perform. Obviously, many diverse implementation solutions can be developed, and the resource allocation and execution management services will select the most cost effective solution. In its decision making process, this service has to accurately predict applications execution times so as to help predict queue times and improve the performance of allocation and scheduling algorithms. Unfortunately, the execution times depend on many factors: data size, specific mining parameters provided by users and actual status of the cloud environment etc. Furthermore, the mutual relations between the data source will essentially impact the response times of knowledge discovery applications. Therefore, to predict the execution time is very difficult.

The basic principle of proposed execution time prediction algorithms is that similar services have the similar features. So, we can construct a database that includes some attributes of knowledge discovery applications together with their respective execution time. For predicting execution times of a knowledge discovery service, firstly we can recognize the similar services in the above database, and then calculate a statistical prediction of these recognized services, finally this statistical value can be as the predicted execution time. Due to the lack of the dynamic attribute of resource availability, we propose to build a database about performance so as to best predict the performance in dynamic environment. This component regards monitoring information about previous implementation of the various knowledge discovery services on particular data sets.

The essential problem with this method is the definition of similarity. There are many standards to judge whether or not services are similar. For example, we can consider two services are similar because they come from the same account and the same network node or because they have the same requirement and run on the same data set. Therefore, we have to research efficient method to recognize the similar services. Such method should have the ability to correctly select the features of services that best decide similarity.

In this paper, we propose a holistic method to prediction that employ reduct algorithm to decide a similarity template and then calculate the execution times prediction based on the recognized similar services. The rest of this paper is organized as follows: We introduce related works and point out the limitations of some previous works in section 2. The suitability of rough set to predict cloud services execution times is discussed in section 3. In section 4, we recall necessary rough set notions used in this paper, and then present our reduct algorithm and services execution times prediction algorithm. In section 5, we present the KDC environment in which the distributed knowledge discovery application can be executed. In section 6, we conduct experiments to evaluate the proposed algorithm. Finally, in section 7, we conclude this paper.

2. Related Works

Some scientists proposed to make use of similarity templates of jobs characteristics to make out similar jobs recorded in the log [1, 3, 10]. A similarity template is a set of characteristics that we employ to compare jobs so as to decide whether or not jobs are similar. Therefore, some scientists choose the queue name as the feature to decide similarity in history records from parallel and distributed workloads. They thought that jobs allocated to the same queue were similar. In other study, scientists employ some templates for the same history records, including client, task name, number of computers and age.

Selecting similarity templates by hand have the following limitations: on one hand, recognizing the features that best decide similarity isn't permanently feasible; on the other hand, it's not universal: even though a specific set of features might be suitable for one field, it's not always suitable to other fields.

In [9], authors employ genetic algorithms and greedy search techniques to automated definite and search for templates. They obtain improved prediction accuracy by means of this method.

In this paper, we propose a novel reduct algorithm which bases on rough set theory to work out the question of automatically choosing features that best define similarity. In contrast to [9], our method determines a reduct as template, instead of using greedy and genetic algorithms and obtains the higher accuracy. Rough sets theory offers a robust theory for recognizing templates automatically. The whole procedure of recognizing similarity templates and matching services to similar services is based on rough sets theory. Rough set theory supplies a suitable solution with a robust mathematical foundation.

3. Rough Set Based Execution Times Prediction

Rough set theory supply a robust theoretical foundation to us for deciding the features that best define similarity since it can efficient handle uncertainty problem in data. Rough set theory only needs the connotation of the data itself, no external additional information. The history of data can be described as an IS in which every instance are the history tasks whose execution times and other attributes have been stored. The attributes in the IS are features of history tasks. In accordance with the terms of rough set theory, tasks execution time is the decision attribute, accordingly, the condition attributes are composed of the other attributes. Rough set model is intuitive and easy to draw causal relationship between attributes, make it easier to determine the dependencies between the stored attributes and the tasks execution times. Therefore, we can recognize the similar task according to the condition attributes that are dependency and importance in deciding the execution time. So, some attributes which strongly impact the execution time can constitute a fine similarity template. Having changed the question of tasks execution time as a IS question, then we study the basis notions that are appropriate in deciding the similarity template. The function of similarity template in task execution time prediction is to recognize a set of features on the foundation of which we use to compare tasks.

A similarity template should be composed of the higher significant attributes that decide the execution time without any redundant attributes. A reduct set should be composed of the minimal number of condition properties which have the same discriminating ability as the whole IS. Accordingly, the similarity template is same as a reduct set which consists of the important contribution attributes. Obviously, if there are too many properties in the template, it is not able to better identify similarities. For example, if n features are included in the similarity template, two tasks are similar only because they are recognized according to all n features. However, this method will largely restrict our capacity to detect similar tasks since not all attributes are certainly related in deciding the execution time. On the contrary, this method will cause mistakes, since tasks that are similarities very much may be measured dissimilar even though they are different in one feature which had little impact on the execution time.

Detecting a reduct set is equivalent to feature selection question. All reduct set of a data set may be got by building discernibility function. Many studies about the reduct have been done: In [5], authors present an algorithm which use significant of properties as heuristics; in [13], authors used powerful equivalence to facilitate discernibility function. More studies show that detecting minimal reduct set is NP hard problem and no general approach. This problem is still open in the related field.

Rough sets theory has very apposite and proper constructs for recognizing the attributes that best define similarity for predicting tasks execution time. A similarity template should contain properties which largely impact the execution time and remove the inappreciable factors. This makes sure that the standards with which we compare tasks for similarity have an important impact on deciding execution time. Therefore, tasks that have the same features regarding these standards will have similar execution time.

4. Heuristic Reduct Algorithm and Service Execution Time Prediction Algorithm

In this section, we first review important rough set concepts [6] used in this section, then present the reduct algorithm and task execution time prediction algorithm.

4.1. Related Rough Set Concepts

- *Information System:* An information system is an ordered pair:

$$IS = (U, C \cup D, C \cap D = \varphi) \quad (1)$$

  Where U is a non-empty finite set of instance called the universe, the elements of the U are called objects or instances. C is a non-empty, finite set of conditional attributes such that $\alpha: U \rightarrow V_\alpha$ for every $\alpha \in C$. The set V_α is called the value set of α and D is a decision attribute.
- *Indiscernibility Relation:* Let $IS=(U,C \cup D)$ be an information system, every subset $E \subseteq C$ defines an equivalence relation $IND(E)$, called an indiscernibility relation, defined as:

$$IND(E) = \{(i, j) \in U \times U : \alpha(i) = \alpha(j)\} \quad (2)$$

- *Positive Region:* Given an information system, $IS=(U,C \cup D)$, let $I \subseteq U$ be a set of objects and $J \subseteq C$ a selected set of attributes. The lower approximation of I with respect to J is:

$$B_*(I) = \{i \in U : [i]_B \subseteq I\} \quad (3)$$

  The upper approximation of X with respect to B is:

$$B^*(I) = \{i \in U : [i]_B \cap I \neq \varphi\} \quad (4)$$

  The positive region of decision D with respect to B is:

$$POSITIVE_B(D) = U\{B^*(I) : I \in U / IND(D)\} \quad (5)$$

- *Reduct:* For any $B \subseteq C$, if $POSITIVE_B(D) = POSITIVE_{B-\{\alpha\}}(D)$ then attribute α is dispensable.

For any a set of attributes $B' \subseteq B$, *if* all attributes $\alpha \in B\text{-}B$ are dispensable, and $POSITIVE_B(D) = POSITIVE_{B'}(D)$, then B' is a ruduc of B'.

In *IS*, there exist numerous reduct sets. Detecting minimal reduct and all reduct sets are NP hard problem and no general approach. For the sake of finding a proper reduct, it is necessary to introduce the concept of discernibility matrix and discernibility function.

- *Discernibility Matrix and Function:* Let *is* be an *IS* with *m* instance. The discernibility matrix of *is* is a symmetric $m \times n'$ matrix with f_{ij} as given below:

$$f_{ij} = \{ \alpha \in is \mid \alpha(y_i) \neq \alpha(y_j) \} \quad for\ i, j = 1, \ldots, n \tag{6}$$

A discernibility function f_q for an *IS* *is* is a Boolean function of *m* Boolean variables $c_1^*, \ldots, c_m^*$ (corresponding to the attribute $c_1, \ldots, c_m$) defined as follows:

$$f_s(c_1^*, \ldots, c_m^*) = \wedge \{ \vee f_{ij}^* \mid 1 \leq j \leq i \leq m, f_{ij} \neq \varphi \} \tag{7}$$

Where $c_{ij}^* = \{ \alpha^* \mid \alpha \in f_{ij} \}$.

The set of all prime implicants of f_S determines the set of all reducts of *IS*.

4.2. Heuristic Reduct Algorithm

The essence of the heuristic reduct algorithm lies in: when a reduct set intersect discernibility matrix, the result set cannot be Φ. Instance *i* and *j* will be indiscernible if the intersection of c_{ij} with any reduct is Φ. If so, it will be in contradiction with the definition of reduct.

We can build this heuristic algorithm based on the following plain and direct way: Let original reduct set $REDUCT=\Phi$, and scan each element of discernibility matrix. If their intersection is Φ, a haphazard attribute from the matrix will be selected and added in *REDUCT*, else pass the element. Redo this process until whole elements are scanned. Once all the elements in the matrix have been processed, we will get a coarse reduct.

Although, this algorithm is most straightforward, the below shortcoming is not overcome. For instance, there are 3 elements in the discernibility matrix: $\{s_1, s_3\}$, $\{s_2, s_3\}$, $\{s_3\}$. Based on above algorithm, the reduct set is $\{s_1, s_2, s_3\}$. In fact, the reduct is obviously only $\{s_3\}$. The reason leading to this problem is that the above algorithm for computation reduct is only necessary condition but not sufficient condition. This algorithm does not take into account that a reduct should be a minimum number of data set.

For obtaining reduct, or even the lesser reduct, we have to add in more conditions.

An effortless but efficient approach is to order the sequence of elements in the matrix. We can understand if there is just only one element in the matrix then this element should be a member of the reduct set. For a reduct set, we think that fewer elements in the set and more frequent occurrences in the matrix will have more ability to classify. Having ordered the c_{ij}, we will select the more contribution elements to prevent the shortcoming from happening again.

The final version of the heuristic algorithm has two significant ideas: one is that the more frequently the element, the more possible it is a member of reduct set; the other is to introduce a weighting mechanism, properties emerged in fewer element will obtain higher weight. The main procedure of the algorithm is presented as follows: First, all the same elements in c_{ij} are integrated and their frequency is marked. Then the matrix is arranged sequence based on the length of each element. If some elements have the same length, give priority to the more frequent element. The frequency of each property is counted while building the discernibility matrix. The frequencies are utilized in assisting selecting property when it is required to select one property from some element to join in the reduct. Algorithm 1 is the pseudo code of our reduct algorithm.

Algorithm 1:

1. *Let an IS information system IS = $(U, C \cup D, C \cap D = \phi)$.*
2. *Let REDUCT=Φ, count $(a_i) = 0$, where $A = \cup a_i$, $i = 1, \ldots, n$.*
3. *Build discernibility matrix MATRIX and calculate frequency of each property(a_i);*
4. *For discernibility matrix MATRIX, integrate and arrange sequence;*
5. *For each element e in MATRIX do*
   a. *If ($e \cap REDUCT == \Phi$)*
      Select property a with maximal count(a) in e;
   b. *REDUCT=REDUCT $\cup \{a\}$*
      End if
6. *Return REDUCT*

4.3. Service Execution Times Prediction Algorithm

In this section, we will review the prediction algorithm on the whole. The input of this algorithm is past record of services features accumulated during some period, especially covering real execution times, and a service with known parameters whose execution time we want to predict. We present the detailed description of this algorithm as follows:

- Partition the past record into condition attributes and decision attributes. The recorded execution time is the decision property and the others are the condition properties.

- Utilize the heuristic reduct algorithm to the past record and recognizing the similarity template.
- Combine the current service *CT* with the past service *PT* to build a current past *CPT*.
- Decide the equivalence classes related to the recognized similarity templates from *CPT*.
- Recognize the equivalence class *EC* to which *CT* belongs.
- Calculate the mean of the execution times of the instances: $EC \cap PT$.

Algorithm 2 shows a formal view of the prediction algorithm. As we illustrated above, the whole procedure of recognizing similarity template and matching current services to similar services is based on rough sets theory. Thus the algorithm supplies a proper method with a sound mathematical foundation.

Algorithm 2:

1. *Let past services = PT, Current Service = CT.*
2. *Partition PT such that the execution time is the decision attribute and others features are the condition attributes.*
3. *Utilize the Heuristic Reduct Algorithm to the past record and produce a similarity template ST.*
4. *Let CPT = CT + PT, where CT and PT are union compatible.*
5. *Calculate equivalence classes of CPT related to ST.*
6. *Recognize equivalence class EC to which CT belongs.*
7. *Calculate the mean of the recorded execution time PET in HT for all instances* $EC \cap PT$.

5. KDC-Knowledge Discovery Cloud

Cloud computing environment is a natural platform for deploying a high performance knowledge discovery application because this kind of application is both information and computation intensive. Once knowledge discovery transplant into cloud computing environment, it will spreads information data, software and computation on distributed cloud nodes, delivers the computation as a service, shares resources and regards the software and information as utilities. Since the resource in Cloud environment can be extended unlimitedly, so this kind of information service lets organizer make their functions increased and performance enhanced more rapidly. Correspondingly, the resources in this environment will need simpler management and less maintenance. The applications of cloud computing will also further rapidly regulate information technology to adapt to the instability and irregular business requirements.

In this section, we present the KDC environment which specially process knowledge discovery in distributed environment. The main purpose to design and develop the KDC system is to make the distributed knowledge discovery possible and improve its performance. Clients can execute the knowledge discovery application in a transparent way, that is, the concrete system structure, operation and characteristic in the Cloud environment is to be hidden. The features of KDC are list as follows:

- KDC adopts the standard, common and open Cloud service mode, and offers unified support to the knowledge discovery applications. According to the existing networks system structure, KDC use the Cloud service to realize communication, operates each other and resource management.
- KDC is open and supports various knowledge discovery tools and algorithms. Various applications can be imported through the uniform interface of Web service.
- KDC has strong scalability. Its performance can be enhanced by increasing network node, high performance computing node and cluster.
- KDC can process distributed large volumes of high dimensional dataset, support heterogeneity data source.

Figure 1 describes the distributed system framework that we designed and developed. It is mainly made up by following components:

- *KDC Client Node:* In consideration of ease of use, the system adopts Browser/Server mode. Cloud client exchanges information with cloud portal through Web browser. Users submit the requirement of knowledge discovery and receive the final result at cloud client.
- *KDC Portal Node:* It provides a single access way to distributed knowledge discovery application based cloud. Users can make use of the whole cloud resource transparently through the cloud portal. This component is responsible for translating users' demand into the language Resource Specification Language (RSL) that can be recognized by cloud. This component is also used for cloud resource discovery and cloud resource allocation management. The final result is returned to cloud portal first, and then returned to users by the portal.
- *KDC Resource Allocation Broker Node and KDC Services Scheduling Broker Node:* User's knowledge discovery requirement has driven cloud resource discovery. According to users' demand condition, KDC resource broker looks for the resources which meet the condition in a large number of cloud resources, including algorithms, computing capability and data resource. It is an important job that finds appropriate resource. As to any application based on cloud, it is first to find appropriate resource, then allocate services and management them. It is conceivable that there may be many nodes which fit a condition. Resource broker is used for finding available resource among Meta Directory Service (MDS); mapping between data resource and computing resource, i. e., the

service allocation broker is responsible for dispatching a certain service on a certain node.

- *Cloud Node:* The cloud nodes are made up of personal computer, high performance computer and cluster. They are the data carrier and the computation implementation entity.

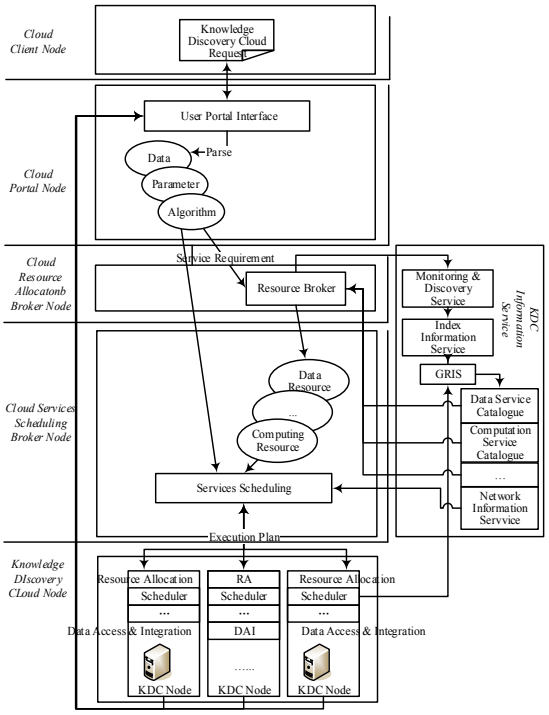

Figure 1. Knowledge discovery cloud architecture.

6. Experiments and Results

This experiment aims to validate the prediction accuracy of our reduct algorithm in KDC environment and investigate the impact on predicting performance by changing the number of condition attributes. We adjust the test case from the past records by removing the execution time attribute and changing the number of condition attributes. Therefore, an experiment case is composed of all attributes but the recorded execution time. This experimental design is to determine a predicted execution time via our prediction technology and contrast it with the real execution time.

We collect past record of knowledge discovery services by performing several knowledge discovery algorithms and recording statistics about the parameters. We conduct several experiments of knowledge discovery services by changing the parameters of the services such as the knowledge discovery algorithm, network parameters, data set, data size and so on.

The simulated environment is similar to an actual cloud environment. It is composed of fifteen machines. Those machines have different physical configurations (CPU, memory, disk, and network adaptor etc.,), operating systems (windows, Mac OS X, Linux and Unix) and bandwidth of network. We used histories with 100 and 150 records and each experimental run consisted of 25 tests.

Figure 2 shows the actual execution time versus predicted execution time of services; Figure 3 illustrates the impact on prediction performance by varying the number of condition attributes.

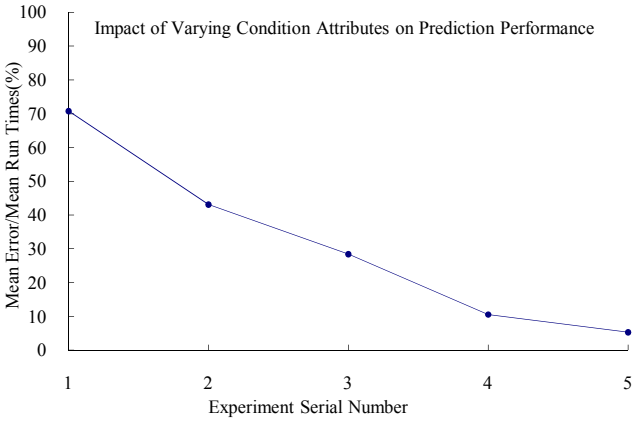

Figure 3. Attributes numbers vs. prediction performance.

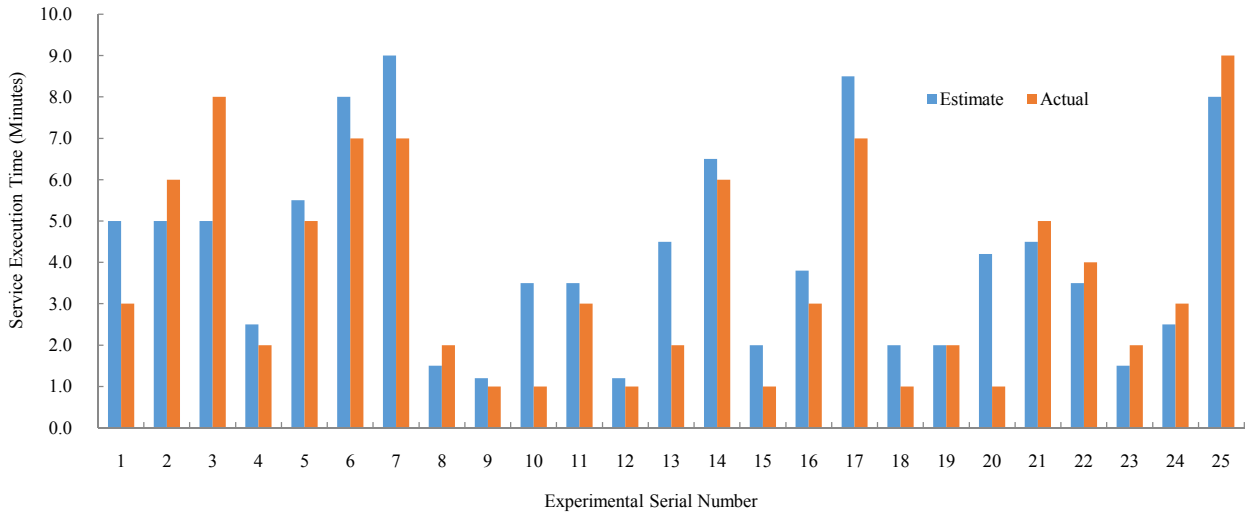

Figure 2. Actual versus predicted execution time.

Another criterion used for contrast is the percentage of the mean error to the mean execution times, that is, (Mean Error)/ (Mean Execution Time). The lower this value is, the better prediction accuracy it has. For the San Diego Supercomputer Center data set SDSC95 and SDSC96, we do 10 times the experiment. For each test case, we recorded the reduct set, the predicted execution time and the real execution time. In [9], authors have published their experimental results for their greedy search and genetic algorithm techniques. We present comparison results of three technologies, which is greedy search, genetic algorithm and Reduct Algorithm, in Figure 4. The experimental result indicates that we obtained more accuracy prediction than other technologies in most cases.

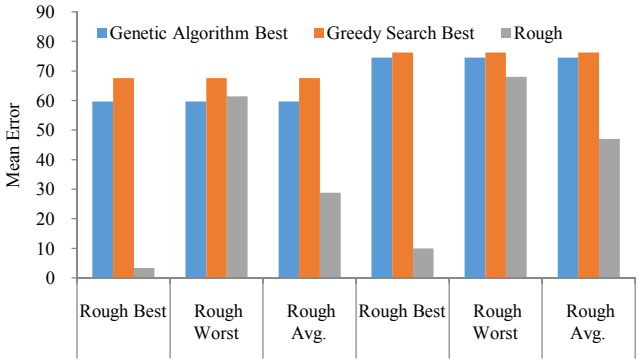

Figure 4. Accuracy contrast.

Since reduct algorithm fully operate on the foundation of history records and need no other information, thus the more abundant the information related to performance, the more accurate the prediction is.

7. Conclusions

We have proposed a KDC system for supporting knowledge discovery application running in cloud environment as well as a holistic approach to predict the service execution times. In order to meet the needs of the system to improve performance, we propose a reduct algorithm for predicting the services execution time in cloud computing environment. Our hypothesis that rough sets are suitable for predicting Web Service execution time in cloud environment is validated by the experimental results, which demonstrate the good prediction accuracy of our approach. The prediction technique presented in this paper is generic and can be applied to other optimization problems.

Acknowledgement

This work is supported by Zhejiang Province Public Technology Research and Industrial Project (No. 2011C31005, and Ningbo Science and Technology Bureau Project (No. 2010C80064).

References

[1] Allen B., "Predicting Queue Times on Space-Sharing Parallel Computers," *in Proceedings of the 11th International Symposium on Parallel Processing*, Geneva, Switzerland, pp. 209-218, 1997.

[2] Daniel N., Rich W., Chris G., Obertelli G., Soman S., Youseff L., and Zagorodnov D., "The Eucalyptus Open-Source Cloud-Computing System," *in Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, Shanghai, China, pp. 124-131, 2009.

[3] Gibbons R., "A Historical Application Profiler for Use by Parallel Schedulers," *in Proceedings of Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*, Berlin, Germany, vol. 1291, pp. 58-77, 1997.

[4] Hassan M. and Abdullah A., "A New Grid Resource Discovery Framework," *the International Arab Journal of Information Technology*, vol. 8, no. 1, pp. 99-107, 2011.

[5] Herawan T., Deris M., and Abawajy J., "A Rough Set Approach for Selecting Clustering Attribute," *Knowledge-Based Systems*, vol. 23, no. 3, pp. 220-231, 2010.

[6] Komorowski J., Pawlak Z., Polkowski L., and Skowron A., "Rough Sets: A Tutorial," *in Proceedings of Rough-Fuzzy Hybridization: A New Trend in Decision Making*, Berlin, Germany, pp. 3-98, 1998.

[7] Luis M., Luis R., Juan C., and Maik L., "A Break in the Clouds: Towards a Cloud Definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50-55, 2008.

[8] Michael A., Armando F., Rean G., Joseph A., Katz R., Konwinski A., Lee G., Patterson D., Rabkin A., Stoica I., and Zaharia M., "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50-58, 2010.

[9] Smith W., Foster I., and Taylor V., "Predicting Application Runtimes Using Historical Information," *in Proceedings of Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*, Berlin, Germany, vol. 1459, pp. 122-142, 1998.

[10] Smith W., Taylor V., and Foster I., "Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance," *in Proceedings of Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science*, Berlin, Germany, vol. 1659, pp. 202-229, 1999.

[11] Vecchiola C., Pandey S., and Buyya R., "High-Performance Cloud Computing: A View of Scientific Applications," *in Proceedings of the 10th IEEE Computer Society, the International*

Symposium on Pervasive Systems, Algorithms and Networks, Kaohsiung, Taiwan, pp.4-16, 2009.

[12] Wang C., Wang Q., Ren K., and Lou W., "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," *in Proceedings of IEEE INFOCOM*, San Diego, USA, pp. 1-9, 2010.

[13] Wang X., Yang J., Teng X., Xia W., and Jensen R., "Feature Selection Based on Rough Sets and Particle Swarm Optimization," *Pattern Recognition Letters*, vol. 28, no. 4, pp. 459-471, 2007.

[14] Yu S., Wang C., Ren K., and Lou W., "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," *in Proceedings of IEEE INFOCOM*, San Diego, USA, pp. 1-9, 2010.

[15] Zhang Q., Cheng L., and Boutaba R., "Cloud Computing: State-of-the-Art and Research Challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7-18, 2010.

Kun Gao received his Master degree and PhD from Jilin University and Donghua University, respectively. He is a researcher at the School of Computer Science and Information Technology in Zhejiang Wanli University, China. His research interests include distributed and parallel computing, cloud computing, knowledge discovery, rough set and so on.

Qin Wang is working as an associate professor in the School of Computer Science and Information Technology in Zhejiang Wanli University, China. Her research interests include computer networks, algorithms and soft computing.

Lifeng Xi is working as a professor in the School of Computer Science and Information Technology in Zhejiang Wanli University, China. His research interests include algorithms analysis, complexity and image processing.