Committee-based Evaluation and Selection of Grid Resources for QoS Improvement

Zhen Wang[1] and Junwei Cao[2]

[1,2]*Research Institute of Information Technology*
Tsinghua National Laboratory for Information Science and Technolog
Tsinghua University, Beijing 100084, P. R. China
[2]jcao@tsinghua.edu.cn

***Abstract*—The grid enables the facility of distributed resource sharing but can provide limited QoS guarantee to grid applications, e.g. in terms of execution time. In this paper, we propose a Committee-based Resource Evaluation and Selection Method (CRESM) to evaluate and select reliable resources for Users, which have higher possibility of finishing tasks in time and provide better QoS. CRESM is composed with a representative layer and a committee layer. The representative layer consists of informed and experienced Users who contribute their individual experiences to judge a particular grid resource. The committee layer aggregates these individual judgments and makes a comprehensive decision based on these aggregated information. All the judgments and the final decision are made fuzzily based on historical statistics of resource evaluation. Experimental results show that CRESM is stable and accurate in different grid environments. CRESM can also defeat collusive deceitful behaviors. Meanwhile, CRESM based resource scheduling can improve QoS support for grid applications.**

I. Introduction

A. Background

The Grid enables the facility of distributed resource sharing in large scale environments [1], where tasks can be executed on remote computers or other resources. But the dependency on remote resources and network transmission makes the Quality of Service (QoS) [2, 3, 4] is easily influenced and hard to control. Among various resource quality requirements, execution duration is the most important and concerned performance metric. In this paper, we mainly use task execution duration or the makespan of multiple tasks to evaluate the quality of a resource. The quality of a resource is defined as the possibility of finishing a task in the predicted execution duration on this resource. Reliable resources are resources with higher possibility to complete a task within the expected execution duration.

How to guarantee the QoS of a resource is a significant problem, especially for applications with execution duration requirements. GARA [3], proposed in 1999, enables the resource reservation and co-allocation for tasks. Market scheduling algorithms including commodity models and auction models bring the QoS as an attribute of a resource in the commerce-based resource selection and allocation. In all these methods, developers assume a task will be successfully finished in time as long as a corresponding resource is reserved. To guarantee this, Users usually reserve a much longer duration than the actual execution duration on a resource, leading to lower resource utilization.

In this paper, we propose a two-layer method, namely Committee-based Resource Evaluation and Selection Method (CRESM), to evaluate and select QoS guaranteed resources from a number of potential Grid resource candidates, based on statistics and pattern recognition theories. The bottom layer, namely the representative layer, consists of informed Users. They contribute their experiences and make fuzzy-based judgments about a resource individually. The top layer is the committee layer which selects judgments from all representatives and makes a comprehensive decision according to the representatives' judgments and their importance through a fuzzy decision making method. Our method is evaluated and tested by experiments in Section 3, which evaluate reliability of a resource accurately and efficiently, defeat collusive malicious behaviors in the Grid and select reliable Grid resources for tasks with QoS improvement.

B. Related Work

QoS is a fundamental issue in the Grid. Current work can be classified into three types.

The first type of solution is to guarantee the QoS by resource reservation and scheduling, based on the Service Level Agreements (SLAs) [2, 5]. The basic idea is to ensure that a task has a corresponding resource usage privilege during a particular duration. As long as a resource is reserved for a task, researchers assume the task is guaranteed to be finished. The QoS for a job then can be regarded as reserving a serial of time slots on corresponding resources for tasks in the job, obeying the workflow requirement. Many papers have been published to address to the QoS based resource reservation scheduling and negotiation [6, 7] issue. Market based scheduling and negotiation algorithms [8] including the commodity model [9, 10] and the auction model [11] appraise the quality of a resource as its attribution in the commerce based negotiation and mapping process. All these work focus on guaranteeing the QoS by ensuring every task is allocated with appropriate execution duration on a corresponding resource and little attention has been put on whether or not this resource can finish the task in time. Currently Users usually simply reserve a resource with overfull duration to guarantee the task can be finished in this reserved duration.

Another method to address uncertain running errors on a resource or time delay on the Internet is to design a fault-tolerance service, which can migrate and restart a task when errors occur. Fault-tolerance services [12, 13] have to detect the status of resources and maintain intermediate statuses frequently, which consume lots of memories, CPU capability or other resources. Besides that, design a general fault-tolerance is also a very difficult work. Existing fault-tolerance services are mainly application specific.

In paper [14], authors try to rank resources according to their usage frequencies and believe the higher the usage frequency of a resource is, the higher quality this resource is. But it may blind users to find a new resource. Also the collusive malicious behavior between a resource and a user has obvious impact on the ranking result.

The rests of this paper are organized as follows. CRESM is described theoretically in details in Section 2. We test and evaluate the performance and the effectiveness of CRESM through simulation experiments included in Section 3. Section 4 concludes the paper with a brief introduction to our future work.

II. CRESM

CRESM is a two-layer method as shown in Figure 1. RP means Resource Provider, which will be judged by CRESM. FKNN is fuzzy k-nearest neighbor algorithm, which is described in the sub-section *C*. We use User to represent a resource consumer or task submitter in the Grid.

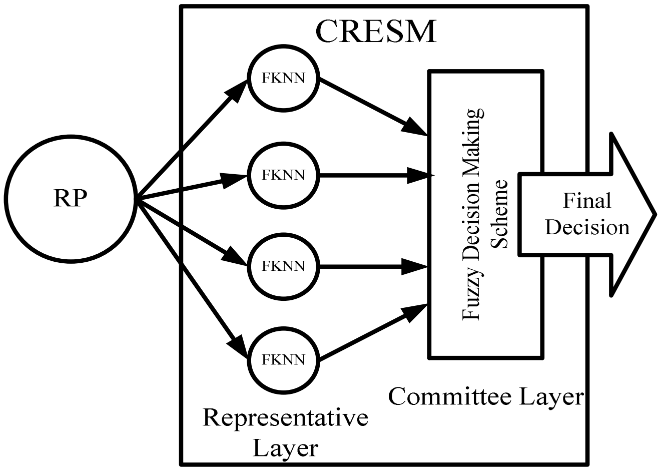

Figure. 1. The architecture of CRESM

The representative layer consists of informed Users in Grid, who are regarded as representatives of other Users. The representative makes an individual judgment about a specific resource according to its experience and contributes it to the committee layer. The committee layer collects judgments from representatives and makes a comprehensive and compromising decision through a fuzzy decision making method. The main purpose of CRESM is to evaluate resources, classify them into two classes: reliable ones which provide high QoS and unreliable ones which provide low QoS, and then selects reliable resources for User applications, which will provide QoS improvement in the end. The way of establishing the representative layer and methods of making judgments and decision are explained in details in the following sections.

A. Evaluation Statistics

We select two types of evaluation statistics to characterize a resource. Before the explanation, we firstly give some essential definitions.

Predicted execution duration: The time a task is expected to spend, based on experiences or calculated according to the resource status and the task complexity. This time can be calculated from theoretical models or is just given by the User based on his experiences.

Execution duration: The time that a task actually takes on a Grid resource. That is the duration from submitting the task to receiving the result.

Interaction: We define the whole activity of submitting a task to a certain resource and receiving the result from the resource an interaction between a User and a resource.

When an interaction between a User and a resource is finished, an evaluation is given according to the task completion situation.

If Execution Duration <= Predicted Execution Duration and the task is finished successfully, this is a high quality interaction and one positive evaluation is given to the resource; If Execution Duration > Predicted Execution Duration or the task is failed, this is a low quality interaction and one negative evaluation is given to the resource.

We simply use a positive evaluation statistics (accumulate all positive evaluations) *pes* and a negative evaluation statistics (accumulated all negative evaluations) *nes* as statistical records to character a resource. In the following section, the two parameters constitute a target vector of a resource in the recognition method. The target vector of the resource *j* is:

$$x_j=(pes_j,\ nes_j).$$

B. Choosing Representatives from Ordinary Users

In the Grid, it is infeasible and inefficient to select judgments of all Users for a particular resource. However, in decentralize virtual community User activity distribution obeys the Power-law [15, 16] distribution, so it is reasonable to select part of informed Users as representatives to opinions of all other Users. We select the User who has interacted with Grid resources many times as a representative, namely the informed and active User.

C. FKNN Classifier at Representative Layer

There are many kinds of recognition methods to classify unknown entities including neural networks, the fuzzy k-nearest neighbor (FKNN) classifier, the Bayesian classifier, the support vector machine (SVM) and so on. The Bayesian classifier needs to know the possibility distribution, SVM can't provide reliability theoretical analysis and the neural network is an unstable method without mathematical theory support. Compared with them, FKNN is a reliable, stable and extendable method with little overhead and strict mathematical support in theory and is good at classifying an unknown sample with sparsity learning samples. In CRESM, we just use FKNN to classify resources on the representative layer.

We divide an unclassified resource into three types: the reliable one which can provide high QoS, the uncertain one which needs to be further or manually determined and the unreliable one which provide low level of QoS and denote these three classes as *r*, *d,* and *u* respectively.

Each representative has a learning table recording resources which have been already evaluated and classified artificially based on the quality of interactions with these resources, as shown in Figure 2. When a low-quality interaction happens between a representative and a resource, namely the task execution duration on the resource is longer than the predict execution duration, the representative records this resource in his learning table as classified sample. The same thing happens when a successful interaction happens. We just select the most informed Users, namely the most active Users in VO as representatives to found a committee.

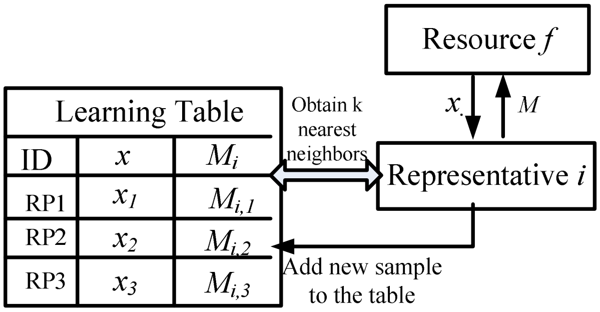

Figure. 2. Learning table of a representative

Assume the representative is *i*. The resource *j* in the table of representative *i* is described with corresponding target vector x_j and evaluated with membership vector $M_{i,j}=(r_{i,j},\ d_{i,j},\ u_{i,j})$, respectively representing the membership degree of belonging to the class *r*, *d,* or *u*. $M_{i,j}$ represents the individual judgment of the representative *i* to the resource *j*.

To evaluate the unclassified resource *f*, the representative firstly selects the *k* nearest neighbors of resource *f* in its learning table based on the distance of target vector. The distance between x_1 and x_2 is calculated as:

$$\|x_1 - x_2\| = x_1 \bullet x_2 = \sqrt{(pes_1 - pes_2)^2 + (nes_1 - nes_2)^2}$$

Assume the unclassified resource *f* has the target vector x_f. Assume the *k* nearest neighbors to *f* in the table of *i* is x_j, $j=1,2\ldots k$ with corresponding target vector x_j, $j=1,2\ldots k$, and membership $M_{i,j}=(r_{i,j},\ d_{i,j},\ u_{i,j})$, $j=1,2\ldots k$. The judgment about the resource *f* the representative *i* made is described by a membership vector $M_{i,f}=(r_{i,f},\ d_{i,f},\ u_{i,f})$, which elements respectively indicate the membership degree to the class *r*, *d,* and *u*. The element of $M_{i,f}$ can be calculated by FKNN using the following formula:

$$r_{i,f} = \frac{\sum_{j=1}^{k} r_{i,j}(1/\|x_f - x_j\|)}{\sum_{j=1}^{k}(1/\|x_f - x_j\|)}$$

$d_{i,f}$, $u_{i,f}$ can be calculated by a similar way, that is replace parameter *r* with *d* or *u* in the formula.

This process happens for every representative and generates a serial of judgments about the unclassified resource, $M_{i,f}$, $i=1,2\ldots n$, *n* is the number of representatives. Then all these judgments will be contributed to the committee to generate a comprehensive decision.

D. Committee Based Decision Making Scheme at Committee Layer

Fuzzy decision making method (FDMM) has two advantages, capability to handle uncertainty, fuzziness, and incomplete information adaptively and capability to make comprehensive and fault-tolerance decision by integrating multiple factors. Because of the consideration of the weightiness discrepancy among different representatives and robustness of decision, we use FDMM to comprehensively evaluate and aggregate various judgments from representatives [17].

Generally speaking, FDMM consists of four elements: 1. a factor set, namely the list of representatives in CRESM; 2. a corresponding weight vector *A*, indicating the importance of a representative in the committee; 3. a factor membership matrix *R*, consisting of all judgments representatives made about a resource; 4. a decision set *B*, the final decision to a resource which is made by the committee according to judgments from representatives, the factor membership matrix and the weight vector.

Assume there are *n* representatives in the committee, the i^{th} representative submits a judgment vector $M_{i,f}$ about the resource *f* to the committee and these judgments are aggregated and considered with the weight vector $A=(a_1,a_2,\ldots\ldots,a_n)$, respectively represents the importance of the corresponding representative. a_i is a parameter from 0 to 1, high value of a_i means more important the judgment of representative *i* is. 0 means the opinion from the representative *i* is totally unimportant and incredible and 1 means the committee will totally adopt the judgment of the representative *i*.

The factor membership matrix *R* is:

$$R = \begin{bmatrix} M_{1,f} \\ M_{2,f} \\ \vdots \\ M_{n,f} \end{bmatrix} = \begin{bmatrix} r_{1,f} & d_{1,f} & \mu_{1,f} \\ r_{2,f} & d_{2,f} & \mu_{2,f} \\ \vdots & \vdots & \vdots \\ r_{n,f} & d_{n,f} & \mu_{n,f} \end{bmatrix}$$

The i^{th} row of the *R* is the judgment contributed by the representative *i*. These elements $r_{i,f}$, $d_{i,f}$ and $u_{i,f}$ respectively represent belonging membership degrees of the resource *f* to the class *r, d* and *u*, judged by the i^{th} representative.

Then use a $M(\wedge \vee)$ model to calculate the final decision set *B'* and normalize it as *B*. The decision set can be described as

$$B' = A \circ R = \{r', d', u'\}$$

The detailed formulas are:

$$r' = \max_i(\min(a_i, r_{i,j})) \qquad i = 1, 2, ..., n$$

$$d' = \max_i(\min(a_i, d_{i,j})) \qquad i = 1, 2, ..., n$$

$$u' = \max_i(\min(a_i, u_{i,j})) \qquad i = 1, 2, ..., n$$

Normalize *B'* as *B*:

$$r = \frac{r'}{r' + d' + u'} \quad d = \frac{d'}{r' + d' + u'} \quad u = \frac{u'}{r' + d' + u'}$$

The element of *B* represents a percentage degree the committee thinks the resource should belong to the corresponding class. The committee can configure a threshold for the membership. If an element of *B* is higher than the threshold, the committee will classify the resource as the corresponding class, since this decision can represent the majority view of the committee or Grid Users. For example, *u* > threshold, the committee believes the resource belong to the *u* class, namely is the unreliable resource. The similar thing happens in the case of *r* or *d*.

If there is no element higher than the threshold, the committee can choose the highest one as the final decision or submit the resource to the User for personally evaluation and determination.

III. Performance Evaluation

We evaluate CRESM in the following two aspects.

Firstly we test the stability and accuracy of CRESM with different parameters and Grid environments. Experiments in the sub-section *B* show how CRESM performs with various configurations and in different Grid environments. As expected, CRESM is tested to be stable, effective and accurate.

Secondly we check the effect of CRESM on the QoS improvement, which Users are directly concerned with. Experiments described in the sub-section *C* show how the makespan is improved if tasks are submitted to a group of resources which are selected by CRESM, compared with other contrastive groups. And the result shows that CRESM can greatly improve the QoS for Users.

A. Simulation Configuration

We build a simulated Grid platform and a random number of interactions happen between Users and resources (Users submit tasks to resources). When an interaction is finished, an evaluation is produced to the resource. The behavior pattern of a resource is described with a two-element vector PT=(pt_1, pt_2), respectively represent the possibilities of finishing the task in time and out time, which will lead to one positive evaluation or one negative evaluation to that resource.

The collusive deceitful behavior should also be considered, for a User may provide a pseudo complex task with a long predict execution duration to a resource to cheat a positive evaluation for the resource. CRESM should defeat this collusive deceitful behavior and select real reliable resources from them. In the experiment, we test the performance of CRESM in an extremely worse case, in which all unreliable resources will try to obtain positive evaluations through collusive deceitful interactions with other Users and to cheat CRESM. Of course, the frequency and the scope of collusive deceitful behaviors are also limited in usual conditions, or Grid is not available (Most of resources are deceitful).

Table 1
Parameters in experiments

Number of resources:	500
Deceitful scores frequency:	1 *pf* per 10 interactions
Threshold:	0.5
Reliable resource: pt_1	0.97-0.99
Reliable resource: pt_2:	1- pt_1
Unreliable resource: pt_1	0.9-0.92
Unreliable resource: pt_2:	1- pt_1
Number of resources in the group for tasks:	100
Representative Number:	5
k:	3
Expected number of interactions before checked by CRESM	100
Percentage of unreliable resources:	30% (150 unreliable resources)

Because the distribution of interactions between resources and Users obey the power-law [15, 16], we simulate a number of random interactions happened on a resource based on the exponential distribution.

The detailed configurations of experiments are shown in Table 1. In order to obtain a stable and meaningful result, we repeat every experiment for 10 times. The experiment results below are mean values of these 10 repeated experiments.

B. Performance of CRESM over Different Environments and Parameters

1) Performance Metrics: The main purpose of CRESM is to classify real reliable resources from unreliable ones or deceitful ones (who may have high positive evaluation statistics because of collusive deceitful behaviors or other uncertain deceitful activities). So the main metrics to evaluate the performance are the error of classifying reliable resources as unreliable resources and the error of classifying unreliable resources as reliable ones, respectively, denoted as the negative error (*NE*) and the positive error (*PE*). For CRESM, *NE* and *PE* is calculated as their definitions.

PE = (Number of unreliable resources classified as reliable ones) / (Number of unreliable resources);

NE= (Number of reliable resources classified as unreliable ones) / (Number of reliable resources).

Obviously, the smaller *PE* and *NE* are, the better the evaluation and selection performance of CRESM is. However, *PE* and *NE* change as the Grid environment or parameters of CRESM changes. In different Grid environments, the resource expected numbers of interactions are different. And the representative number and *k* value of CRESM also affect the final performance.

We analyze the performance of CRESM with different resource expected numbers of interactions (Grid environment), representative numbers and *k* values and respectively show the results in Figures 3-5.

2) Performance of CRESM: Figure 3 shows how CRESM performance when the resource expected number of interaction in the Grid changes. The points in figure 3 present PE and NE when resources are evaluated by CRESM. The percentage of unreliable resources in Grid is 30% before CRESM. After CRESM, more than 85% percentage unreliable resources have been recognized and classified. In selected resources which are regarded as reliable, the percentage of unreliable resources is only 6.42% (200 expected interactions). As the cost, a negative error is generated, about one tenth of the reliable resources are classified as unreliable, as show in figure 3 (200 expected interactions). But in many applications, time is a critical attribution for the selection of resources. A User just wants to aggregate the most reliable resources for his application in unacquainted resource environment.

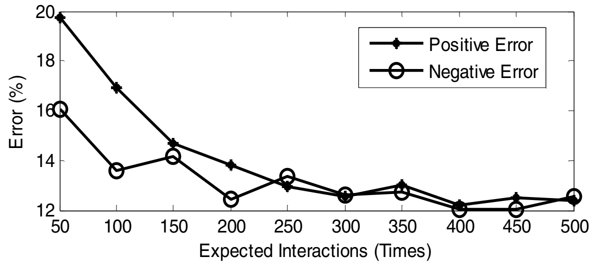

Figure. 3. Performance of CRESM over different interaction numbers

From the figure we can conclude that both *NE* and *PE* decrease as the expected number of interactions increases, especially when the resource expected number of interactions is lower than 300. So we can expect a quick improvement of classification as Grid resources are gradually used. The accuracy nearly keeps stable if the expected number of interactions of resources is more than 300.

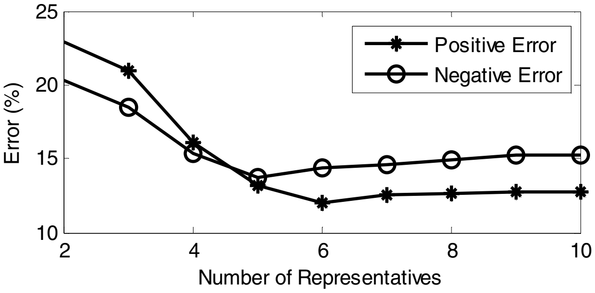

Figure. 4. Performance of CRESM over different numbers of representatives

Figure 4 shows how the scale of the committee affects the performance of CRESM. The number of representatives has a significant effectiveness to *PE* and *NE*, which greatly decrease as the number of representative increases from 2 to 6. It is perceptible and acceptable because a large committee contains more learning samples and can cover large scale of situations, especially when there are few representatives in the committee. But the effect of representatives decreases if the committee is big enough. As demonstrated in Figure 4, *PE* and *NE* nearly keep stable or even slightly increase as the number of representative increases. That is because the influence of every representative on the final decision decreases if the number of representatives in the committee increases. In this case, even if a new representative is introduced into the committee, it can hardly affect the final decision of the committee.

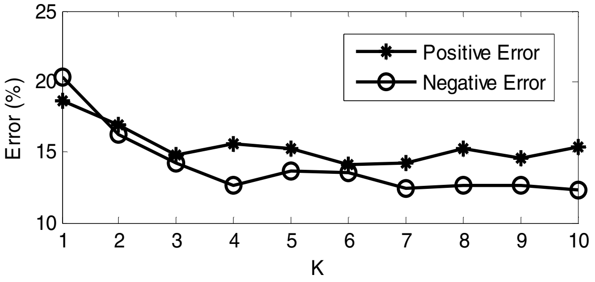

Figure. 5. Performance of CRESM over different *k*

Figure 5 shows the performance of CRESM in different *k* values. Obviously, *PE* and *NE* sharply decrease as *k* increases from 1 to 4. After that, *k* has little effect on the final performance. Select more neighbors will add more samples in the classification process and improves the performance while consume more computational capability. However, the judgment of FKNN on a resource is mainly determined by a limited number of nearest neighbors. If the *k* is more than that number, the new joined neighbor has little effect on the final judgment of the representative (for the new neighbor is quit far away from the resource) or even brings negative effect. In our experiment, *k* with value of 3 or 4 brings a best performance cost ratio to CRESM.

C. Makespan Improvement of CRESM

The purpose of CRESM is to evaluate large number of unacquainted resources in the Grid and provide a selection of reliable resources for Users and tasks automatically.

In this paper, we define the QoS of a resource as the possibility of finishing the task on this resource in the predicted or reserved execution duration. In the following experiments, we calculate makespans of a number of tasks on different groups of resources, which are scheduled by two widely used scheduler algorithms: Min-Min scheduler and Max-Min scheduler [18, 19]. These different groups of resources are the resource group selected by CRESM, the resource group selected randomly from the Grid, and resource group consists of totally ideal resources (provide 100% QoS). The results of the experiments indicate that CRESM can provide improved QoS for Users compared with random resource selection.

1) Experiments and Performance Metrics: There are three kinds of resource groups. The first resource group is CRESM based Group, selected by CRESM. The second resource group is Random Group, selected randomly from Grid. The third resource group is Ideal Group, consisting with ideal resources which provide totally quality guaranteed services to Users (finish tasks in time). The last two kinds of resource groups are the contrastive groups.

We take makespan as the primary metrics to evaluate the improvement of QoS, which is the duration that a serial of tasks are all finished on a resource group. In order to get a general conclusion over different scheduling algorithms, we test makespans of two widely used scheduling algorithms: Min-Min scheduler and Max-Min scheduler. The Min-Min scheduler is good at scheduling tasks with little predicted execution duration difference while Max-Min scheduler can provide better makespan performance than Min-Min when predicted execution durations of tasks on different resources are disparity. We assume the task predicted execution durations for Min-Min scheduler obeys the Poisson Distribution with expected value of 10 and the same distribution with expected value of 50 for Max-Min scheduler. If a resource provides a failed or unreliable service, we simply assume the task the actual execution duration is twice as its predicted execution duration (assume this task will be restart when it is failed).

In order to uniformly compare the scheduler performance in different scenarios, we propose a makespan ratio to evaluate the QoS improvement: *Makespan Ratio = Makespan of other group / Makespan of Ideal Group.*

In this experiment, we also adopt the Grid configuration in Table 1. The results of experiments are shown in Figures 6 and 7.

2) Makespan Improvement:

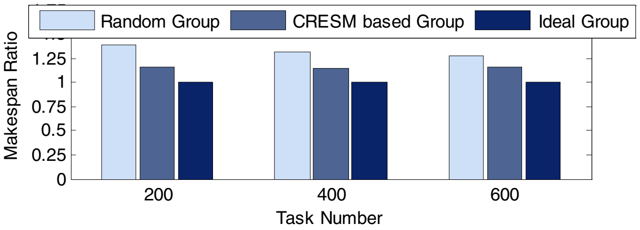

Figure 6. Makespan ratio of Min-Min scheduler

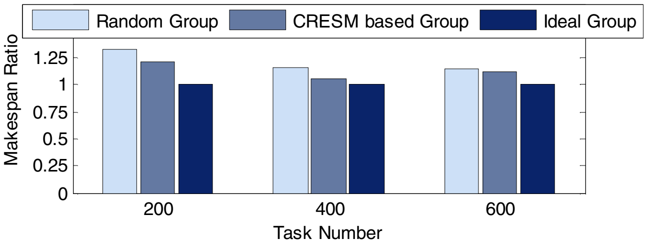

Figure 7. Makespan ratio of Max-Min scheduler

Figures 6 and 7 are the makespan ratios of two schedulers on the three resource groups which are collected by different rules. The Random Group has the same percentage of unreliable resources with that in Grid, namely 30% as the configuration. CRESM based Group consists of resources which have been evaluated and selected by CRESM. The percentage of its unreliable resources keeps stable at 8% in different scenarios. The Ideal Group provides 100% quality guaranteed services for Users. The makespan ratio of Ideal Group is 1, just for contrast.

As demonstrated above, makespan ratios of both schedule algorithms in CRESM based Group are lower than those of Random Group, no matter how many tasks are allocated. The average makespans of CRESM based Group are 1.158 in Min-Min scheduler and 1.129 in Max-Min scheduler. The corresponding makespans of Random Group are 1.324 and 1.212, much higher than those of CRESM based Group. That is to say, CRESM can evaluate and select resources for tasks with QoS improvement to Users.

IV. Conclusion and Future Work

In this paper, we propose a Committee-based Resource Evaluation and Selection Method, consisting of two layers. We describe and explain these two layers theoretically, namely the representative layer with the FKNN classifier and the committee layer with a fuzzy decision making method. Then we test the stability, accuracy and QoS improvement of CRESM through a serial of experiments and data analysis. The results show that CRESM can stably, accurately evaluate the reliability of resources in Grid and select reliable resources for tasks with QoS improvement. And its evaluation performance can be continuously improved as Grid resources gradually used.

CRESM tries to help Users evaluate and select reliable resources in an unfamiliar and large scale Grid environment and guarantee the selected resources can finish a task in the predicted execution duration or the reserved duration. Resource reservation scheme and scheduling algorithms ensure a task is allocated with an appropriate duration on a particular resource and our CRESM tries to guarantee the selected resource can finish the task in this reserved duration. It can not only bring QoS improvement, but also advance the resource utilization ratio in the Grid. Users don't need to reserve overfull execution duration to handle uncertain errors for resource selected by CRESM can provide reliable services with higher possibility.

Currently, evaluation results of CRESM about a certain resource are fuzzy-based membership degrees to the three types of resources. Establishing a theoretical association between membership degrees and the possibility of providing reliable services will be a significant work, for it enables a more accurate quality control over a resource. Further more, CRESM can be combined with resource reservation scheduling algorithms to provide a high resource utilization ratio, for a User or a scheduler can reserve different overfull execution durations on a resource according to its reliability level.

ACKNOWLEDGMENT

This work is supported by National Science Foundation of China (grant No. 60803017), Ministry of Science and Technology of China under the national 863 high-tech R&D program (grants No. 2006AA10Z237, No. 2007AA01Z179 and No. 2008AA01Z118), and Ministry of Education of China under the program for New Century Excellent Talents in University and the Scientific Research Foundation for the Returned Overseas Chinese Scholars.

REFERENCES

[1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, 1998.

[2] I. Foster, A. Roy, V. Sander, *A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation*, 8th International Workshop on Quality of Services, 2000.

[3] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy, *A Distributed Resource Management Architecture that Supports Advance Reservation and Co-Allocation*, Proc. IWQOS, 1999.

[4] D. A. Menasce, E. Casalicchio, *QoS in Grid Computing*, IEEE Internet Computing, Vol. 8, pp.85-87, 2004.

[5] R. Al-Ali, O. Rana, D. Walker, S. Jha, and S. Sohail. G-qosm: *Grid Service Discovery Using QoS Properties*, Computing and Informatics Journal, Special Issue on Grid Computing, 21(4):363–382, 2002.

[6] M. Wu, X. Sun, Y. Chen, *QoS Oriented Resource Reservation in Shared Environments*, Sixth IEEE International Symposium on Cluster Computing and the Grid, Vol. 1, pp. 601-608, 2006.

[7] B. Li, J. Chen, D. Zhao, *Looking-Ahead Algorithms for Single Machine Schedulers to Support Advance Reservation of Grid Jobs*, 10th IEEE International Conference on High Performance Computing and Communications, pp. 335 – 341, 2008.

[8] L. Tang, Z. Yang, Z. Yu, Y. Wang, *A Quality-Driven Algorithm for Resource Scheduling Based on Market Model on Grid*, International Conference on Parallel Processing Workshops, 2007.

[9] D. Allenotor, R. K. Thulasiram, *Grid Resources Pricing: A Novel Financial Option Based Quality of Service-profit Quasi-static Equilibrium Model*, 9th IEEE/ACM International Conference on Grid Computing, pp.75 – 84, 2008.

[10] D. Allenotor, R. K. Thulasiram, *G-FRoM: Grid Resources Pricing A Fuzzy Real Option Model*, IEEE International Conference on e-Science and Grid Computing, pp. 388 – 395, 2007.

[11] M. Schwind, O. Hinz, R. Beck, *A Cost-based Multi-unit Resource Auction for Service-oriented Grid Computing*, 8th IEEE/ACM International Conference on Grid Computing, pp. 137-144, 2007.

[12] L. Valcarenghi, P. Castoldi, *QoS-Aware Fault Tolerance in Grid Computing through Topology-Aware Replica Placement*, International Conference on Transparent Optical Networks, Vol. 5, 2006.

[13] H. Lee, S. Chin, J. H. Lee, D. Lee, K. Chung, S. Jung, H. C. YuA, *Resource Manager for Optimal Resource Selection and Fault Tolerance Service in Grids*, IEEE International Symposium on Cluster Computing and the Grid, 2004.

[14] N. Hamid, F. Haron, C. Yong, *Resource Discovery Using PageRank Technique in Grid Environment*, Sixth IEEE International Symposium on Cluster Computing and the Grid, 2006.

[15] M. Faloutsos, P. Faloutsos, C. Faloutsos, *On Power-Law Relationship of the Internet Technology*, Proc. ACM SIGCOMM' 99, pp. 251-262, 1999.

[16] G. Siganos, M. Faloutsos, P. Faloutsos, C. Faloutsos, *Power laws and the AS-level Internet Topology*, IEEE/ACM Transactions on Networking (TON), 2003.

[17] N. Baldo, M. ZorzH, *Cognitive Network Access Using Fuzzy Decision Making*, IEEE International Conference on Communications, pp.6504-6510, 2007.

[18] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, R. F. Freund, *A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems*, Journal of Parallel and Distributed Computing, Vol. 61(6), pp. 810-837, 2001.

[19] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. Freund, *Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems*, 8th Heterogeneous Computing Workshop (HCW '99), pp. 30-44, 1999.