A new Approach for Selecting Best Resources Nodes by Using Fuzzy Decision Tree in Grid Resource Broker

Asgarali Bouyer[1], Mohammadbager Karimi[1], Mansour Jalali[1], Mohd Noor MD SAP[2]

[1]Islamic Azad University- Miyandoab Branch
[2]University Technology of Malaysia-faculty of computer science and information system
basgarali2@siswa.utm.my, Mba_karimi@iaut.ac.ir Mansour_m200@iaum.ac.ir, Mohdnoor@fsksm.utm.my

Abstract. Nowadays, Grid Computing has been accepted as an infrastructure to perform parallel computing in distributed computational resources. Grid has users, resources, and an information service (IS). Resource broker service is one of the main services in grid to find resources, filter resources, allocate resources, etc. Resource selection is part of resource broker that is an important issue in a grid environment where a consumer and a service provider are distributed geographically. In this paper, we design and implement a new data mining –based Grid resource broker service for selection resources on grid environment. The role of this resource broker service is using learning method to find the best nodes according to the requirements of the job and the distributed computing resources on the Grid. The provided application can be executed on top of Globus Toolkit (GT) middleware. The results of experiments show a strong effect in improving resource finding cycle.

Keywords: Grid Resource broker, Resource Selection, Data mining, Fuzzy Decision Tree.

1 Introduction

Grid is a decentralized heterogeneous system that made up virtual organizations (VOs). Each VO is composed of several different nodes. Each node can be server computers, desktop PCs, clusters, and other kinds of hardware, which are sharing some resources with other nodes. A main goal of grid computing is enabling applications to identify resources dynamically to create distributed computing environments [1].

The Grid allows executing jobs in different nodes. In order to perform job scheduling and resource management at Grid level, usually it is used a Resource Broker or a meta-scheduler. A resource broker is fundamental in any large-scale Grid environment. The task of a Grid resource broker and scheduler is to dynamically identify and characterize the available resources, and to select and allocate the most appropriate resources for a given job. In a broker-based management system, brokers are responsible for selecting best nods, ensuring the trustworthiness of the service provider. Resource selection is an important issue in a grid environment where a consumer and a service provider are distributed geographically across multiple

administrative domains. Choosing the suitable resource for a user job to meet predefined constraints such as deadline, speedup and cost of execution is an important problem in grids. In our approach, we highly have solved some of these problems [2].

In this paper we will not do a resource discovery method, but in fact we present a novel way for selecting the best nodes in pool of discovered nodes. Resource selection involves a set of factors including application execution time, available main memory, disk (secondary memory), resource access policies, etc. resource selection must consider information about resource reliability, prediction error probability, and real time execution. However, these various performance measures can be considered under the condition that the middleware allows adaptation of its internal scheduling with desired application's services. We have considered all of these factors in our approach. Also to reach for better selection we used the Decision Tree with Fuzzy Logic theory [3]. Induced decision trees are an extensively-researched solution to classification tasks. The use of Fuzzy Logic techniques may be relevant in case representation to allow for imprecise and uncertain values in features.

The rest of this paper is organized as follows. Section 2 refers to previous research on resource brokering and scheduling. Section 3 describes Fuzzy Decision Tree Algorithm in our method. section 4 discuss the system design and implementation details of our OGSI-compliant Grid resource broker service, respectively. Section 5 describes experimental results and section 6 concludes the paper.

2. Related works

Many projects, such as DI-GRUBER [5], eNANOS [6], AppLes [7] and OGSI-based broker [4] have been performed on grid. In this section we introduce some of these brokers.

DI-GRUBER [5], an extension to the GRUBER brokering framework, was developed as a distributed grid USLA based resource broker that allows multiple decision points to coexist and cooperate in real-time. GRUBER has been implemented in both Globus Toolkit4 (GT4) and Globus Toolkit3 (GT3). The part of DI-GRUBER that dosing resource finding and selecting is called The *GRUBER engine*. *GRUBER engine* is the main component of the GRUBER architecture and that implements various algorithms to detect available resources and maintains a generic view of resource utilization in the grid [5]. GRUBER does not itself perform job submission, but it can be used in conjunction with one of various grid job submission infrastructures.

The eNANOS Resource Broker is an OGSI-Compliant resource broker developed as a Grid Service and is supported by Globus Toolkit (GT2 and GT3) middleware [6]. eNANOS architecture neither uses data mining methods to select the best nodes from the pool of discovered nodes, nor implements in Web Services (WS) bases frameworks.

AppLes (Application Level Scheduling) focuses on developing scheduling agents for individual Grid applications [7]. AppLes agents have an application oriented scheduling mechanism, and use static or dynamic application and resource

information to select a set of resources. However, they perform resource discovering and scheduling without considering resource owner policies. Also they do not support system-oriented or extensible scheduling policies.

Another resource broker service has been presented by Young-Seok Kim and et al. [4]. It is an OGSI- based broker that is supported by GT3. It is a new general purpose OGSI-compliant Grid resource broker service that performs resource discovering and scheduling with close interactions with GT3 Core and Base Services. This resource broker service considers resource owner policies as well as user requirements on the resources.

The EZ-Grid project [8] applies Globus services to create Grid resource usage easier and more transparent for the user. This is obtained by developing easy-to-use interfaces coupled with brokerage systems to assist the resource selection and job execution process.

Another works have been done in resource selection field (e.g. Condor/G [15], Nimrod/G, LSF and so forth), but we cannot introduce all of them in this paper.

Finally, we mention that none of those systems or brokers uses machine learning methods to find (select) the best nodes for purposed jobs.

3. Fuzzy decision tree

Induced decision trees are an extensively-researched solution to classification tasks. General decision tree always has a deterministic result, and therefore this feature is not good in some application. Thus, if we can use DC with fuzzy logic, we can achieve a better decision. Fuzzy decision Tree (FDT) is the generalization of decision tree in fuzzy environment. The knowledge represented by fuzzy decision tree is closer to the human classification [10]. In our approach we used a Fuzzy decision tree (FDT).

3.1. Fuzzy Logic (FL)

Essentially, Fuzzy Logic (FL) is a multi-valued logic that allows middle values to be defined between conventional evaluations like yes/no, true/false, black/white, etc. Fuzzy logic is an extension of Boolean logic that replaces binary truth values with degrees of truth. It was introduced in 1965 by Prof. L.Zadeh at the University of California, Berkeley [9]. The basic notion of fuzzy systems is a fuzzy set. for example, to classify the fuzzy set of climate, which may be consisted of members like “Very cold”, “Cold”, “Warm”, “Hot”, and “Very hot”. The theory of fuzzy sets enables us to structure and describe activities and observations, which differ from each other only vaguely, to formulate them in models and to use these models for various purposes - such as problem-solving and decision-making [9]. We will not discuss fuzzy set such natural extensions here and more about fuzzy logic can be found in [13].

3.2. Fuzzy Decision Tree Algorithm

This algorithm is a developed version of ID3 that operate on fuzzy set and it will produce a fuzzy decision tree (FDT). Before this, other researchers [3, 12] considered the FDT in their applications. Thus, their results showed that this algorithm is suitable for our approach. But there are two important points in making and applying FDT [11]:

- Select the best attribute in each node to develop the tree: there are many criteria for this aim, but we will use one of them.
- Inference procedure from FDT. In the classification step for a new sample in FDT, we may encounter many leaf nodes with deferent confidence that offer some classes for purposed sample. Thus, the fitness mechanism selection is important here.

Before we express the algorithm, we will consider some assumptions and notation:
- The training examples will be called E set with N example. Each example has N properties and every property A_j contain m_j linguistic term and so the number of output class will be as following.

$$E = \{e_1, e_2, \ldots, e_n\} \quad A = \{A_1, A_2, \ldots, A_N\}$$

$$for\ A_j, 1 \le j \le N \qquad C = \{c_1, c_2, \ldots, c_k\}$$

Fuzzy terms for A_j $(\{V_j1, V_j1, \ldots, V_j\{ [jm] _j \}\}$

- The set of exist examples in **t** nodes show by **X** .
- $\mu_{c_k}(x)$: represent the degree membership of example x belongs to the class ck.
- $\mu_{v_{jL}}(x)$: represent the degree membership of crisp value for attribute j in example **x** belongs to the fuzzy term v_{jL} in j attribute. Also consider four following formulas:

$$p^*(c_k)^{[]} = \sum_{x \in X} \mu_{c_k}(x) \qquad (1)$$

$$p^*(v_k)^{[]} = \sum_{x \in X} \mu_{v_L}(x) \qquad (2)$$

$$p^*\left(\frac{c_k}{v_L}\right) = \frac{p^*(c_k \cap v_L)^{[]}}{p^*(c_k)^{[]}} \qquad (3)$$

$$p^{\uparrow} * (c_j k \cap v_j L)^{\uparrow} = \Sigma_j (x \in X) \equiv [\mu_j (c_j k)^{\uparrow} (x) . \mu_j (v_j L)^{\uparrow} (x)] \quad \blacksquare(\&\&(4)\ )$$

3.2.1 Creating a Fuzzy Decision Tree

Step1: Start with all the training examples, having the original weights (degree membership of each sample to desired class is considered 1 value), in the root node. In other words, all training examples are used with their initial weights (This initial weight is not necessarily 1).

Step2: if in one of the node t with fuzzy set X one of the below condition is true, that node will consider as a leaf node.

Con1: for all examples of set X, the proportion for degree membership in a class to sum of degree membership of all data to different classes is equal or greater than θr .

$$\left[\frac{\sum_{x\in X}\mu_{c_k}(x)}{\sum_{k=1}^{n}\sum_{x\in X}\mu_{c_k}(x)}\right] \geq \theta_r \qquad (5)$$

Con2: sum of degree membership of all data in set X, less than Threshold θr .

$$\sum_{k=1}^{n}\sum_{x\in X}\mu_{c_k}(x) < \theta_r \qquad (6)$$

Con3: there have not been existed another attribute for selection.

Step3: if any conditions of step 2 for desired node is not true, then this node should be developed. Thus:

Step3.1: find all attributes in a path from root node to desired node, and then remove it from attribute set. So remaining attribute will be more luck for selection.

Step3.2: for every remaining attribute (A_i), we should select an attribute according to Entropy measure [10] to develop the tree (A_{MAX}).

Step3.3: split X set into subsets $X_1, X_2, \ldots, X_{mA_{MAX}}$ so that, all elements in X_i, there is a coefficient of fuzzy term v_i for A_{max}.

Step3.4: for every of these subsets, we will define nodes $t_1, t_2, \ldots, t_{mA_{max}}$ and then the edges are labeled by v_i values (i=1,2,…, mA_{max}). Then, the degree membership for each example to new node will be computed as following.

$$\mu_{c_k}(x \in X_{v_i}) = \mu_{c_k}(x \in X_T)\,\mu_{v_i}(x \in X_T) \qquad (7)$$

Step3.5: exchange each X_i with X and then repeat step 2 and 3.

4. System Architecture

We have shown a general architecture for this approach (figure 1). Our supplied application is performed on top of GT3. But it can be applied for GT4. For the nonce, we have provided an isolated application that can be worked based on GT3, for this purpose. The Result of every node is sent in an XML document and is stored in a Temporary XML Database (TXD).

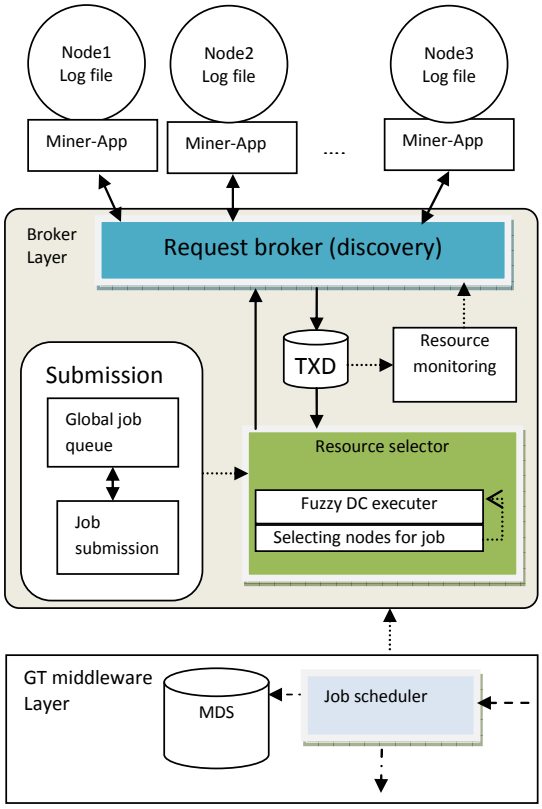

Figure 1. General architecture for our broking

4.1. Miner Application

To do this, we want to install a Miner Application (MA) for every node in a purposed grid. MA contains an internal small database (in log file role). One of the primary tasks of MA is writing log file. When desired node is connected to grid, MA must update its log file (insert a new record to database) or when a new job is submitted to this node, MA will update the related record, because we want to know the number of jobs that are executed on this node. At the moment, if the job is finished successfully or if the job is failed for any reason, thus, MA will update the log file (there is a Boolean field in table that if it is set to TRUE, this means that the related job has been finished successfully, otherwise, it means that the job is not successfully done and has failed). Also, we have considered some new tasks for Grid Resource Broker (GRB), which we have called Optimal GRB. Before selecting any nodes (for aimed job) by GRB, one of these tasks will be executed, this is responsible for sending a packet to each node on grid besides previous tasks. Needless to say, this task can be executed during recourses discovery operation by GRB. Further, as already stated, there are many different methods to find resources (nodes), but will not

concentrate on how we can discovery nodes; and we will not mention them in this paper. Suppose that, there are many different nodes in our grid that are ready for executing jobs and we want to select some nodes in the pool of these nodes. At the beginning, GRB has sent a packet to each connected nodes to our grid. This packet contains some information about a new job (e.g. IP Sender, Size of the job, Size of needed RAM and HDD, average time needed for execution, approximate execution start time, minimum power to CPU, etc.). On the other side, when MA in node gets this packet, it will open the packet for analysis. If there are sufficient resources to do the desired job, MA will perform a data processing technique on its own mini-database (or its log file) to obtain some computation for this job. Some of produced results are as follows:

- Average Hit Ratio (AHR): This attribute represents an average rate of success in all previous times.
- Number of all submitted jobs on this node (AAJ).
- Number of all jobs submitted at this time, on the previous days, on this node (AATPJ).
- Number of all jobs successfully finished at this time, on the previous days (NSTP).
- Hit Ratio for this time-period on previous days (HRTP). For example, how many jobs in 1.30 AM o'clock to 2.00 o'clock have been executed?
- Average Size of successfully finished jobs (ASF).
- Average Response Time for finished jobs (ART).
- Average Response Time for jobs that have the same size as the purposed job and have been successfully finished (ARTSS).
- Hit Ratio for the last twenty jobs (HRT).
- Date, Time and Size of the last successfully finished job (LSJ).
- Date, Time and Size of the last failed job (LFJ).
- Size of the largest successfully finished job (LSI).
- Numbers of all previous jobs that almost have the same size as the purposed job (ASS). Needless to say, the size of the previous jobs is not exactly the same as the size of the desired jobs. For example, for a job with size=340KB we must find all of the previous jobs between 1K to 500KB size.
- Number of all previous jobs that have the same size as the purposed job and are successfully finished (NSS).
- Moreover, processor speed and CPU availability (Idleness) are important for choosing a node.

In addition to the node information, these results will be sent to GRB from any node. There, GRB will analyze them to select/deselect the desired nodes. We mention that always the last collected result will be saved by GRB.

4.2 Broker Layer

In this layer we have added two new sections beside general broker's sections. The first section is related to Request Broker section. This section must broadcast packet to all of the nodes in grid, then it must receive and save the sent results from each node in temporary XML database (TXD). Next, Recourse selector section will

execute a Fuzzy decision Tree Algorithms on TXD (gathered result). We are doing this task in sub-section inside Resource Selector that we call FDT executer. Whenever this algorithm has finished its task, the next sub-section, SNJ (Selecting node for job), will use the result of the algorithm to identify suitable nodes.

4.2.1. FDT executer

This section is considered for executing FDT algorithm on TXD data. As you know, FDT is a machine learning technique for extracting knowledge that is nearer human decision. In this research, we have used FDT algorithm (FID3), because it is reliable and flexible and also has a high accuracy in selecting samples. All used samples for both training and testing are extracted from the provided database (TXD). After that FDT algorithm was performed by FDT executer, therefore we can select a desired class for purposed jobs. Also, Jobs can be divided in several groups: high reliability jobs, real-time jobs, normal jobs, testing jobs and etc.

4.2.2. SNJ sub-layer

Based on the gathered results from FDT executer, this section will select appropriate nodes based on job conditions. There are many parameters in this section, but the main parameters that must be considered, are as follows:

1. Very High Reliability jobs : if we want to execute the desired job successfully with high reliability (response time is not very important), the AHR, HRTP, ASF, HRT measures are very important. There is a priority between these measures. For example, to achieve high reliability, AHR and then HRTP have a high priority. Of course, other measures are also important. SNJ will analyze these measures form gathered results (provided by FDT executer). For example, if there are six nodes that have almost the same AHR and HRTP, or ASF and so on, then other measures (e.g. ART or HRT) will select to evaluate the performance of these nodes. It is possible that there are some states in that SNJ cannot select its own nodes without limit. For example, suppose that SNJ needs to select seven nodes for doing the desired tasks, and there exist only five nodes with high reliability (AHR and HRTP over 95%) and also, if there exist other nodes with low reliability (less than 50%), then GRB can use other parameters to decreasing risk. For example, for two remaining nodes, SNJ can consider HRT parameter, because this is better than other Random-based methods. All of this will be done by SNJ. Also it can use multi- versioning in hierarchical architecture to increase reliability [14]. In other words, it tries to start the versions through candidate nodes in parallel and distributed form by dispatching some replicas of an offered job to the best-selected nodes with a special order. For example, to perform job1, we can use three nodes in hierarchical form and send replicas of this job to the desired nodes. Thus, when one of these nodes finishes the related job and sends its results to GRB truly, then GRB will send a message to stop and abort this task on other nodes. In this way, fault tolerance will be improved and so, reliability in finishing related task will be increased.
2. Execution in Real Time: if we want to execute a job in real time status, so the CPU speed and ART have highest priority and next priority respectively belong to ARTSS, HRTP, AHR, ASF, and LSI and so on. Also processor's power and

communication line bandwidth are important. In this approach we have concentrated on two kinds of jobs that are mentioned in this section.

For a fuzzy set, the idea of vagueness is introduced by assigning an indicator function that may take on values in the range 0 to 1. The following observations are considered:

Count(S_i): *returned the number of successfully finished jobs on node$_i$*
Count(ST_i): *returned the number of successfully finished jobs in the last 20 submitted jobs on node$_i$*
Count(AAJ_i): *returned the number of all submitted jobs on node$_i$*
Min(ART): *return the minimum ART in between of all nodes*
MAX(ASF): *return the maximum ASF in between all nodes*
Min(CPU_SP_i)= *return the minimum CPU speed in between all nodes*

Suppose that $1 \le i \le n$ (n is showing the number of nodes), here we mention how to compute or convert deterministic values to fuzzy sets. Some attributes are have been computed below (member functions) and they are very important to decide on selecting nodes:

- $A1_t = M(AHR_t) = \frac{Count(S_t)}{Count(AAJ_t)}$
- $A2_t = M(ART_t) = 1 - \left(\frac{ART_t - Min(ART)}{ART_t}\right)$
- $A3_t = M(\blacksquare HRT \blacksquare_i t) = Count(\blacksquare ST \blacksquare_i t)/20$
- $A4_t = M(HRTP_t) = \frac{(Count(NSTP() * Count(AAJ(t))}{Count(AATPJ_t) * Count(AAJ_t) - Count(AATPJ_t)^2 + 1}$
- $\blacksquare A5_t = M(HRS\blacksquare_t) = \frac{Count(NSS_t)}{\square} Count(ASS_t)$
- $\blacksquare A6_t = M(ASF\blacksquare_t) = \frac{ASF_t}{MAX(ASF)}$
- $\blacksquare A7_t = M(Speed\blacksquare_t) = 1 - \frac{Min(CPU_{SP})}{CPU_{SP_t}}$
- $\blacksquare A8_t = M(IDLE\blacksquare_t) = \frac{(CPU\ IDLE_t)}{100}$
- $\blacksquare A9_t = M(RAM\blacksquare_t) = \frac{(Free\ Available\ phistcal\ memary_{\square})_t}{MAX(Free_{RAM})}$

As you see, The A5 shows the ratio of successful jobs that have similar size with the desired job to all successful finished jobs. A7 shows the CPU power and A8 shows the measure of system Idle in fuzzy range.

Table1: assign a weight for each attribute

Name of attributes	Weight for High reliability	Weight for Real-time	Weight for Normal jobs
$A_1 \rightarrow$	WH_1=1	WR_1=0.7	WN_1=0.8
$A_2 \rightarrow$	WH_2=0.4	WR_2=1	WN_2=0.8
$A_3 \rightarrow$	WH_3=0.7	WR_3=0.6	WN_3=0.6
$A_4 \rightarrow$	WH_4=0.9	WR_4=0.4	WN_4=0.6
$A_5 \rightarrow$	WH_5=0.5	WR_5=0.2	WN_5=0.4
$A_6 \rightarrow$	WH_6=0.3	WR_6=0.1	WN_6=0.2
$A_7 \rightarrow$	WH_7=0.5	WH_7=0.9	WH_7=0.5
$A_8 \rightarrow$	WH_8=0.4	WH_8=0.8	WH_8=0.5
$A_9 \rightarrow$	WH_9=0.1	WH_9=0.2	WH_9=0.1

For the nonce, these nine attributes will be evaluated in fuzzy behavior. We must mention that based on the type of jobs, they will take a weight. This weight has been allocated based on empiric and the effect of each attributed in classification by DT. These weights are representing in Table 1. Now, to find a node with very high reliability rather than other nodes, we should compute the following computation for each node and then we will select that node with maximum Value. We will have this similar method for other type of jobs.

$$Val_{node_i} = \sum_{i=1}^{9} (WH_i * A_i)$$

$$selected\ node = a\ node\ with\ Max\left(val_{node_i}\right)$$

5. Experimental Results and Discussion

We have designed two applications for our approach. The first application is executed on nodes (MA). The second application is designed to implement a new provider for GRB and will use MA's result- selects the best nodes for jobs. In our experiments, eight resource computing nodes and one server are used to evaluate performance of this approach. The hardware information has been described in Table 2. These nodes communicate with server via internet. Then, the *MA application* is installed on nodes and *broker provider application* is installed on the server computer. When a node is connected to grid (server), right away, MA will insert a new record in to node's log file. After that, we have started to obtain the samples. We divide 24 hour in to following parts:

7-9	9-12	12-15	15-17	17-20	20-22	22-24	24-2	2-7

Table 2: Hardware information

Name	*Type of hardware*
$Node_1$	Pentium4(Cache1MB),CPU2.2(INTEL), RAM 256
$Node_2$	Pentium(Cache2MB),CPU2.4(INTEL), RAM 512
$Node_3$	INTEL Pentium,CPU3.0(GLI),2, RAM 1G
$Node_4$	Intel(R) Core(TM)2 Duo CPU 2.16GHz RAM(3.49 GB)
$Node_5$	Intel(R) Core(TM)2 Duo CPU 2.16GHz RAM(3.49 GB)
$Node_6$	Intel(R) Core(TM)2 Duo CPU 2.16GHz RAM(3.49 GB)
$Node_7$	Intel(R) Core(TM)2 Duo CPU 2.16GHz RAM(2.9 GB)
$Node_8$	HP ProLiant ML370 G4 High Performance – Intel Xeon 3.4 GHz (2 processors)L2 cache(RAM 8G)

Table3: The computed result in 7.00 to 9.00 o'clock

Node's Name	A1	A2	A3	A4	A5	A6	A7	A8	A9
$Node_1$	0.89	0.9	0.9	0.9	1	0.55	0	0.54	0.02
$Node_2$	0.87	0.94	0.9	0.85	0.95	0.8	0.21	0.77	0.04
$Node_3$	0.9	0.92	0.9	0.85	1	0.67	0.39	0.97	0.05
$Node_4$	0.94	0.95	0.95	1	1	0.9	0.48	0.87	0.43
$Node_5$	0.95	0.96	1	0.9	1	0.85	0.48	0.98	0.39
$Node_6$	0.96	0.95	1	1	1	0.7	0.48	0.16	0.38
$Node_7$	0.92	0.93	0.9	0.95	1	0.85	0.48	0.8	0.28
$Node_8$	0.8	1	0.85	1	1	1	0.78	0.65	1

Table4: the computed result in 12.00 to 15.00 in 21August (job size 10.24 MB at 12:45 o'clock)

Node's Name	A1	A2	A3	A4	A5	A6	A7	A8	A9
$Node_1$	0.902	0.87	0.95	0.94	0.8	0.45	0	0.78	0.02
$Node_3$	0.908	0.93	1	0.86	0.8	0.53	0.39	0.92	0.05
$Node_5$	0.961	0.96	0.95	0.89	1	0.89	0.48	0.96	0.39
$Node_6$	0.964	0.93	0.95	0.94	0.9	0.66	0.48	0.80	0.38
$Node_7$	0.92	0.94	0.95	0.92	0.6	0.91	0.48	0.93	0.28
$Node_8$	0.81	1	0.9	0.89	0.8	1	0.78	0.75	1

In the first six days we have used MA Application but we didn't use the result of MA in our broker application. Moreover we always have sent a job for all available nodes.

After that, we have activated broker provider to select only suitable nodes. Therefore, in seventh day, we have taken the below results(Table 3) from available nodes in the morning in order to execution a high reliability job with size 4.47 MB and execution time almost 18 minutes. As you see, all eight nodes are accessible in this moment. The Table shows us, in A2 column, $Node_8$ is the best and $Node_1$ is worst (in fuzzy range). When these results have been delivered to server, broker provider on server side has selected $Node_4$ for this purpose. Then job sent to this node for execution and after a little time, job finished successfully on $Node_4$ (see figure 2).

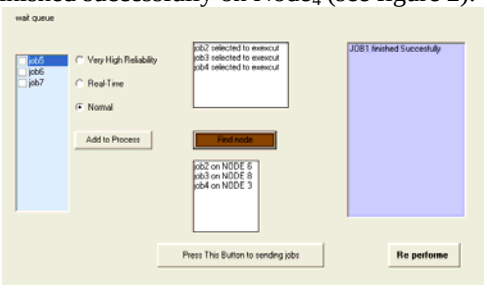

Figure 2. A part of Broker

The priority list nodes for this job were as following (high reliability priority for job):

$Node_4 > Node_5 > Node_8 > Node_7 > Node_6 > Node_3 > Node_2 > Node_1$

If this job had a real-time priority, the below order was selected by broker provider Application:

$Node_8 > Node_4 > Node_5 > Node_7 > Node_3 > Node_2 > Node_6 > Node_1$

n following days, all measures, was based-on broker provider application. After doing 120 measures, we took a below results to execute a job with 10.24 MB and 22 minutes for approximate time. The following results sent by each participated nodes at time 12-15:

As you see, there are only 6 nodes in available. This job is considered as a Real-time job, thus $Node_8$ was selected as the best node by proposed application. The selection priority of nodes is as following:

$Node_8 > Node_5 > Node_6 > Node_7 > Node_3 > Node_1$

If the job had a very high reliability priority, then the selection priority will be as below.

$Node_5 > Node_8 > Node_6 > Node_7 > Node_3 > Node_1$

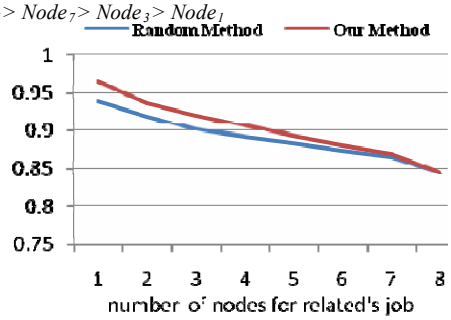

Figure 4: The ratio of successfully finished jobs in both Random method and our provider

In this method we are choosing the best conditions for job. Whereas in other methods (for example, random methods), It's possible that have a high risk to select a node. The Ratio of successful jobs in our methods is compared with another random method [16] in Figure 3. For each node, we have considered one job and the execution test has been repeated for many times. This shows that our method has a good performance in stable state.

The result shows that our approach can achieve better performance under this strategy. After each measure, it is seemed that, the ratio of successfully finished jobs, have improved. It memorable that, for all jobs smaller than 5MB and approximate time less than 2 minutes, almost all jobs finished successfully.

6. Conclusion

Selecting some nodes in the pool of discovered nodes is a challenging problem. Many methods for this purpose have been presented. Our proposed approach is learning based which can reduced extra overhead communications and faults in cycle of selection. This broker provider application along with MA offer a dynamic decision to access any of the available and appropriate nodes by using main important criteria.

The results of our experiments show that this approach has a better performance than others and it will operate according to user's requirements. Stability is a helpful characteristic for this approach, so the fault happen is nearly predictable.

Acknowledgments

The work in this paper was completely supported by Islamic Azad University-Miyandoab branch from.

References

1. K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, "Grid information services for distributed resource sharing,". In 10th IEEE Symposium on High Performance Distributed Computing, San Francisco, California,August 7-9, 2001.
2. K. Krauter, R. Buyya, and M. Maheswaran, "A Taxonomy and Survey of Grid resource Management Systems", Software Practice and Experience, 32(2): 135.164, 2002.
3. M. Umano, H. Okamoto, H. Tamura, F. Kawachi, S. Umedzu and J. Kinoshita, " Fuzzy decision trees by fuzzy id3 algorithm and its application to diagnosis systems", Department of Systems Engineering and Precision Engineering, Osaka University, Japan, IEEE, 1994
4. Y. Kim, J. Yu, J. Hahm, J. Kim, et al., "Design and Implementation of an OGSICompliant Grid Broker Service", Proc. of CCGrid, 2004.
5. Dumitrescu, C., Foster, I., "GRUBER: A Grid Resource SLA Broker", in Euro-Par, Portugal, September 2005.

6. Ivan Rodero, Julita Corbalán, Rosa M. Badia, Jesús Labarta, eNANOS Grid Resource Broker, in European Grid Conference (EGC2005), Springer Berlin / Heidelberg, 2005.
7. H. Casanova, G. Obertelli, F. Berman, R. Wolski, "The AppLeS parameter sweep template: user-level middleware for the grid", In Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, IEEE Computer Society.
8. B. Chapman, B. Sundaram, and K. Thyagaraja. EZ-Grid: Integrated Resource Brokerage Services for Computational Grids, 2001. http://www.cs.uh.edu/ ezgrid/.
9. G.J. Klir and B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice Hall, 1995.
10. C. Marsala and B. B-Meunier, “Choice of a method for the construction of fuzzy decision trees”, University P. e M. Curie, Paris, The IEEE International Conference on Fuzzy Systems, 2003
11. C. Z. Janikow, “fuzzy decision trees: Issues and methods”, IEEE Transactions on Systems, Man and Cybernetics, vol. 28, 1998.
12. C. Marsala and B. B-Meunier, “Choice of a method for the construction of fuzzy decision trees”, University P. e M. Curie, Paris, The IEEE International Conference on Fuzzy Systems, 2003
13. L.A.Zadeh,"Making computer think like people, IEEE spectrum, 8/1984, pp 26-32 [8] S.Haack, " Do we need fuzzy logic? " Int .Jr nl .of Man-Mach.stud , vol.11, 1979.
14. A. Bouyer, A. movaghar, B. arasteh. “A Multi Versioning Scheduling Algorithm for Grid System Based on Hierarchical Architecture” In Proceedings of the 7th IADIS International Conference on WWW/Internet, Vila Real, Portugal. Oct 2007.
15. J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. In Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing (HPDC10), San Francisco, CA. Aug 2001.
16. M. Meybodi, N. Ariabarzan.” A dynamic methods for searching and selecting nodes in peer to peer fashion“. Presented 10th conference computer science in Tehran, IKT2006.