Using Self-announcer Approach for Resource Availability Detection in Grid Environment

Asgarali Bouyer
Islamic Azad university-Miyandoab branch
Miyandoad, west Azarbayjan, Iran
Basgarali2@siswa.utm.my

Ehsan Mohebi, Abdul Hanan Abdullah
Faculty of Computer Science and Information System
University Technology of Malaysia
Skudai, Johor, Malaysia
Mehsan3@siswa.utm.my, hanan@utm.my

Abstract—Since the Grid is a dynamic environment, the detection of available resources and prediction of resource availability in near future is important for resource scheduling. In a Grid environment, prediction and evaluation of resource availability is the prerequisite for a reasonable resource selection and good scheduling guarantee. There are many approaches for discovery and prediction of the resource availability that have some weaknesses (e.g., complexity time, predicting, using out of date data, etc). In this paper, we present a new method for detection of available resource based on online-announcer with no inquiry from Grid scheduler. We use a rough set analysis in each grid node to get some useful rules for predicting about node’s behavior in the grid. The experiment results show that our proposed approach is fast algorithm and it applies effectively a reliable method to predict of proper resources for scheduling.

Keywords-component; availability; resource scheduling; scheduler; Rough set

I. INTRODUCTION

Nowadays many of researchers have believed that Grid computing is best infrastructure for implementing of very large process instead of supercomputers. Due to grid’s ability, many research Laboratories and research groups e.g. CERN[18], TeraGrid [20], AustrianGrid [19] are established to study on e-Science by Grid infrastructure. The grid is a distributed infrastructure for sharing large number of heterogeneous nodes for cooperating problem solving. Node in grid can be a PC-desktop, Server, Cluster or even Super computer [4]. Grid’s users able to spread out their computation over thousands of machines on grid systems to get the increasing of performance, accuracy and speedup for own computations. In this paper, we propose solutions for the problem of discovering resources and matching proper task to compatible resources in the grid.

Recently, Machine learning has become very popular as an aid to researchers interested in resource management in grid. Several thousands of computing resources are participating in grid computing system to executing job (tasks) and almost the situation, conditions and behavior of these resources saved by GIS (Grid Information Service) a part of middleware or related applications. Information Service in the Grid provides the ability to discover and monitor resources which is fundamental for the Grid infrastructure [3]. Resources schedulers need information about the efficiency, reliability, availability load and etc to determine which node should be used to run a desired task according to GIS information. But this information usually is not complete and up to date; and it gathered arbitrary. So that, to record more characters about nodes and their treat, we have provided a database on each node to record of the node’s performance.

The situation is complicated by the fact that the resources in the grid change dynamically over time, due to nodes joining or leaving the grid. Note that a solution to the resource discovery problem necessary but not sufficient for example, suppose that only two nodes in the grid are compatible with a task. Therefore, the task must be matched in one of the two nodes, but then it should (ideally) go to the one that would allow it to finish earlier or reliable or available until finishing task. For this purpose, we used Rough Set (RS) theory [7] to generate rules in order to analyze by scheduler to match suitable task to proper node. RS as a mathematical tool to deal with uncertainty and vagueness in data provides us with a sound theoretical basis to determine the properties that define similarity [6]. The goal of using rough set in this paper is generate useful rules for applying by grid scheduler to find proper nodes and match tasks to available nodes based on job’s status.

This paper mainly focuses on a new grid resource availability algorithm in cooperation with rough set tool. The rest of the paper is organized as follow: Section II gives a detailed related works in the past. Section III introduces the rough set concepts, and in Section IV a new approach has been presented to do an online resource announcer based on up-to-date information. The simulation experiments results and evaluation of obtained result with other approach is brought in Section V. The last section concludes the whole paper and alludes to the future work.

II. RELATED WORK

One of the predictor methods for availability is based on probability theory that is presented in [14]. Probability theory is used to get the resource availability prediction and evaluation. They have considered four important criteria for availability metrics such as resource local task execution
time, resource off-line time, waiting queue length and waiting time for design a predictable method. Each measure has particular computing for each node. Based on obtained value for these measures, this method predicts the availability of nodes. It seems that, this method due to use probability theory is one of the fast methods. However, the prediction results in the probability theory are not so reliable and number of considered parameters is low. In general, this method uses a fast prediction algorithm.

Grid Resource Prophet (GriS-Prophet) [21] is a system for resource availability characterization and prediction. The availability predictor on this framework use Bayesian Inference and the Nearest Neighbor Predictor. The Availability Pattern Recognition and Classification module is a general toolbox of data mining tools that uses expert knowledge about grid systems to provide fast and accurate predictions for grid resource availability. It has been tested on Australian Grid (consisting of 28 grid sites geographically distributed in Austria with 1500 processors) and according to authors’ claims this approach normally in 90% was successful (accuracy measure).

Another prediction based approach has been presented in [9][10]. These mentioned models use empirical host CPU utilization and resource contention traces, prediction technique, and multi-state prediction based scheduler for resource availability. These models compute the transitions from available to each state during the previous N days to produce a Markov chain for state transition predictions. After that, described models determine the probability of transitioning to each state from the available state. author counts the number of transitions to each state from available (i.e. three counters), uses these counts to determine the probability of exiting available for each state, then computes the probability of staying available as one minus the sum of the three exit state probabilities.

Brent Rood and et al. in [11] have presented a new model based on Condor [22]. Their model, as a transitional predictor, considers each day’s transitions separately by first computing the probabilities for each of the N days and then combining them. They also develop several transition weighting techniques and apply them to these algorithms to further increase prediction accuracy. By the way, this model uses multi-state availability prediction to forecast the occurrence, and improving scheduling results. Their results show that this model can be effective to get the better accuracy in prediction. In contrast, due to analysis and computation of all transaction in separately manner for each day and then combination of these results for predicting, significantly computation time for this model is considerable.

Another method is Integrating Genetic and Ant Algorithm into P2P Grid Resource Discovery [13]. In this algorithm at the beginning, genetic algorithm is applied to find the coarsest Grid Peer resources quickly. After that, the ant algorithm is used to find accurate resources. They have argued that their algorithm in peer-to-peer grid can improve the performance of resource discovery.

Finally, several machine learning techniques use categorical time-series data to predict rare target events rule-based classification [12, 26], signal processing [17], and hybrid models [5]. Moreover, two useful researches have been introduced in [15][16] that we devolve them to readers.

III. ROUGH SET ANALYSIS

Rough Sets Theory [7][8] has often proved to be an excellent tool for the analysis of vagueness and uncertainty inherent in making decisions. It also proved to be very useful for analysis of decision problems concerning objects described in a data table by a set of condition attributes and by a set of decision attributes. Rough sets can be applied on information represented in the form of a table called information system, Decision Tables or information tables.

We present some concepts of the rough set theory used in the process of generating rules in order to select nodes by grid scheduler.

Our purpose is to carry out a reasonable approach for resource availability in order to resource selection by using rough set rules for desired tasks in the grid scheduling cycle. To do this issue we will use classification algorithm based on rough set theory to generate rules. It takes the nodes information data table (NIDT) as input. In the context of rough set theory, Data are often presented as a table, columns of which are labeled by attributes, rows by objects of interest and entries of the table are attribute values. The output is three Matrixes (generated rules are showed in matrix form).

The incoming data table as a whole can be treated as a information system of the form L =< U, A, V, f >. Here, U = \{x_1, x_2, ..., x_n\} is a nonempty set of objects (NIDT data) and n is the number of objects in NIDT data; A = C \cup D, in which C =\{c_1, c_2, ..., c_m\} (m is the number of conditional attributes) is a nonempty set of conditional attributes, and D is a finite set of decision attributes; and V = \bigcup_{a \in A} V_a, where V_a is a domain (value) of the attribute a, and there is a map f: U \rightarrow A \rightarrow V is called the information function such that f(x, a) \in V_a for every a \in A, x \in U.

![Figure 1. This figure represent an overview of the upper approximation and the lower approximation and boundary region.](image)

Let X be the target class; the lower approximation \( B^-(X) \) contains all the patterns or equivalence classes of B-indiscernibility relation that definitely belong to the class X and the upper approximation \( B^+(X) \) permits overlap. Since the upper approximation permits overlaps, each set of data
points that are shared by a group of classes define indiscernible set (Figure 1). Thus, the ambiguity in assigning a pattern to a class is captured using the upper approximation. Employing rough set theory, the proposed classification scheme generates soft classes (classes with permitted overlap in upper approximation).

Positive Regions and Dependency: Suppose that IND(B) denote the set of equivalence classes of U with respect to B (and B C A and A = C u D). Also let IND(D) denote the set of equivalence classes of U with respect to D. The positive region of B in IND(D) is defined as POSB(D) = ∪ { B(X) : X E IND(D) }.

We use three attributes (start time, final status of task, and completion time) as decision attributes. These attributes can be acted upon as condition attributes and decision attribute of a decision system. Desired application only uses one of this attributes at a moment as decision attribute and at the same time, other two attributes will be considered as conditional attributes. For example, if dependability and speed factors be more important, the second and third attribute respectively is considered as Decision attribute. Moreover, there are other conditional attributes that we have mentioned in next section.

In addition, many different algorithms have been suggested for data discretization [22], but here due to get a fast discretization, our method uses a manual discretization as following:

- Step1: A = the average value for attribute a. ((Σ Va)/n and n is the number of objects).
- Step2: Min = minimum (Va), Max = Maximum (Va).
- Step3: If Va ≥ [(A+Max)/2] then Va = 1.
- If Va < [(A-Min)/2] then Va = 0.

This discretization has been considered for all mentioned attributes except that two attributes (Number of Faulted Grid Tasks, Number of Successfully Finished Grid Tasks). Because, at the first time, we should define a new attribute as following, and then do this discretization for new attribute. In other word, these mentioned attributes are equivalent with one new computed attribute.

\[ H = \text{Number of Successfully Finished Grid Tasks} / (\text{Number of Faulted Grid tasks} + \text{Number of Successfully Finished Grid Tasks}) \]

After doing all above mentioned steps, the generated output will be packed and sent to Scheduler’s Application.

IV. OPTIMAL AVAILABILITY DETECTION BY ROUGH SET ANALYSIS

In this method, each resource is a voluntary announcer to inform its status to scheduler without any inquiry from scheduler. In other words, each node evaluates to accept or deny new tasks. To do this, all nodes implement rough set analysis in its own data. Each node must records all information about submitted or executed grid’s tasks due to get truly decision based on previous recorded data. An overview of proposed approach is represented in Figure 2.

In this approach, we provide two applications with their own local databases (DBs) for grid nodes and scheduler. Each node will record its information about submitted or executed grid’s tasks in its own local DB. Moreover, the scheduler DB is considered to save the received current status of grid nodes. Every time a node is connected to grid, it must update its new information on scheduler’s DB. This information is very important, because scheduler based on this information will select nodes for scheduling. In this approach two applications are provided to analyze and inform available resources to scheduler. These applications are called Node Application (NA) and Scheduler Application (SA).

![Figure 2. An overview of our offered approach](image)

A. Node Application (NA)

This application is considered for grid nodes (user nodes) and is responsible to following tasks:

- Record full information about grid’s task: When scheduler wants to submit a new task on a desired node, delivers it to the First layer (Sender and Receiver layer) in NA application. This layer is responsible for getting the scheduler’s requests and also sends the result files to scheduler. When a new task is received by this layer, Task-recorder section is called. Task-recorder in NA should save all required data about submitted grid tasks. Therefore, for every new grid task NA record several important properties at this time as CPU Load, Free memory (RAM), Task-ID, size of the new task, priority of the new task (we consider only 3 priority Low, Normal and High), number of all local tasks (system’s and user’s tasks), number of all local tasks with high priority (above normal, high or real-time), number of all grid tasks (in...
running, ready, waiting status), amount of Data Transmission Rate (DTR) related to this node in the grid (DTR probably has upheaval in some times), start time of task execution, spent time for this task, completion time, and final status of task (running, success, Abort or fail). Some of this information (e.g. spent and completion time, final status of task and so on) is saved after finishing the task.

- Deciding on whether the node is ready to accept or deny new grid tasks, Pre-processing section in NA analyzes its own previous and current status (mentioned in above) to specify coming status. So if there are required conditions to get new tasks, the desired node would accept new tasks. Otherwise, it evaluates execution time to determine how many processes in near future will finish. With the assumption of finishing these processes, is the desired node enable to accept new tasks coming or not? Also it is possible that some high priorities processes will join to current processes in near future (e.g. automatically start a Virus Scan program, Auto saves or backup by some application, and so forth). Thus, NA has to consider all possible status to get the best decision.

- Doing Rough Set Analysis (RSA) to find useful rules: we have considered rough set analysis to produce some useful rules about availability of desired node in NA application. Previous information (Above mentioned) is necessary for doing rough set analysis. NA will do RSA in order to determine these: when node was successfully available to terminate a task? Why tasks are failed? And which tasks has maximum success on this node? To do this, we consider three attributes (start time, final status, and completion time) as decision attributes. It mentions that, NA only uses one of these attributes in a moment; and other two attributes will be conditional attributes at the same time. By the way, rough set analyzer section (in NA application) uses discretization method on real data. For example, consider the spent time that expresses by seconds (e.g. 640sec); we have used a simple formula to discretize this attribute based on our experience as following:

\[ D_{\text{spent-time}} = \lceil \log_2 \left( \frac{\text{Spent-time}}{60} \right) \rceil \]  

All of attributes have been discretized by rough set analyzer according to our considered formulas; thus, it is not necessary to mention them here.

Since rough set analysis takes time, though not considerable, but it is possible that we have encounter with this question: When will NA execute rough set analysis? To answer this question, we supply two conditions for doing a rough set analysis:

a. The first condition is that any Rough set analysis in last 24 hours has not been done.

b. Number of currently added tasks to this node is more than 1% of previous submitted tasks in the past days.

- Computing some related information about this node and sending it along with other obtained results (mentioned in 2) to scheduler: there is a section in NA that is called Final-processing and sending. This section must compute Success Ratio, Average of Completion Time (ACT), Average of CPU-Idle (how much percent is CPU free or idle) and Average of free memory (RAM). Finally sending this computed data and all information about accepting or denying new tasks and also the result of rough set analysis to scheduler.

\[ \text{Success Ratio} = \frac{\text{Number of success tasks}}{\text{Number of success tasks} + \text{Number of Failed or Aborted tasks}} \]

\[ \text{ACT} = \frac{\sum \text{GT}_{pi}}{n} \]

\[ \text{GT}_{pi} \text{ is completion time for Grid task } i; \text{ and } n \text{ is the number of success tasks.} \]

- Immediately changing status: suppose that we have a ready node. If node’s resources fully had been occupied, NA has to change its status to non-acceptance and then inform this change to scheduler. This task is done by Urgent Change section in NA.

**B. Scheduler Application (SA)**

This application is responsible to do the following tasks:

- At first, SA is responsible to receive data from grid nodes and save them in SA database. To do this, we have provided a layer in SA application for communication with grid nodes. All input and output data is done by this layer.

- Classifying grid nodes: there is a section in SA called “Nodes’ Classifier” to categorize existing nodes. Nodes’ Classifier divides grid nodes in four classes based on sent information by NA.

  - Class A: these nodes are ready to accept new tasks and also will be available in future. The nodes in this class are better to select by scheduler in order to task scheduling.

  - Class B: in this class the nodes are available to submit new task. However, it is possible that in near future they will not available. If there is a task with least deadline time to start execution and also there are not enough resources from class A, scheduler is forced to schedule nodes from class B.

  - Class C: the nodes belong to this class are not available at this time, but may be in near future they will be available. These nodes are useful for long time deadline tasks or also can be replaced with coming failed nodes.

  - Class D: this class’s nodes are worst nodes for scheduling. Because these nodes are unavailable and in near future they almost are not ready to accept new tasks.

- Another task for SA is computing of the priority for nodes in each class based on gathered results from each node. This is done by computing priority Section in SA application and it is vital for grid scheduler to selecting proper nodes. The main goal of computing priority is obtaining the priority for nodes in Class A. other three Classes are used at the least. By the way, CPU Speed and data transmission rate (DTR) in grid for each node must be considered. Priority for each \( i \)th node will be computed as below (formula 2):

\[ P_i = \text{(SuccessRatio)} + \left( \frac{\text{CPU IDLE}_{i}}{100} \right) + \left( 1 - \frac{\text{ACT}_{i}}{\text{MAX(ACT)}} \right) + \left( \frac{\text{freeRAM}_{i}}{\text{MAX(freeRAM)}} \right) \]
The MAX(ACT) in formula 2 return biggest average of completion time in between of all nodes; and MAX(freeRAM) return largest free RAM in between of all nodes.

For example, if \( P_i \) is larger than \( P_j \) it means that \( P_i \) is better than \( P_j \) or \( i^{th} \) node has better priority than \( j^{th} \) node for selecting by scheduler.

- This method also utilize the nodes with Low Priority (LP) based on its ideal condition as far as possible (for normal tasks) and holding High Priority (HP) nodes for important tasks or maybe normal tasks. HP node almost is better for all situations, whereas LP node rarely is consistent with a few situations. Therefore, at the first, scheduler will try to match a suitable LP node for desired task based on rough set rules. If it is not founded, it will select a HP node for desired task.

V. EXPERIMENTAL RESULTS AND DISCUSSION

Our approach simulated by GridsimV.4.2. In our approach we provided Node Application (NA) for grid nodes (user nodes or provider nodes) and Scheduler Application (SA) for Scheduler’s machine.

At the first time, we matched a provided database with real data for each node based on our previous work [2]. We used this data in order to get an estimation of node’s behavior in future and also to obtain accurate and reliable results and finally time complexity.

We also considered five types of tasks (940KB, 2.5MB, 9.3MB, 27.5MB, 124 MB) to submit on desired nodes. Based on the behavior of a computer in real world, these nodes are simulated with different characteristics by Java programming in Gridsim Simulator. Sometimes simulated nodes are enabled to execute processes (some special processes provided as internal process that belong to its own nodes and not grid) from internal queue to change its own status. This is necessary because we want always encounter with dynamic nodes. When a node executes an internal process, it maybe needs to update its information on scheduler DB.

At the first testing, we experimented the availability of nodes for an instance task (940KB) over 250 nodes. We consider random value for some parameters e.g., number of local process at this moment, CPU, RAM, and so on; and due to this parameters changing, the number of ready nodes was variable after each NA execution. After executing NA by Nodes, amount number of nodes had been readied to accept new tasks (normally more than 50%). However, all results from all nodes saved in scheduler DB. Then, SA program on desired node immediately was executed and nodes grouped in desired classes. At continue, an instance task (940KB) was executed by 90% of available nodes. This test repeated for twenty time that obtained result have been showed in Figure 3.

The analyze of the Figure 3 show that in this approach, tasks averagely finished (successfully) in 98.3% of available nodes that previously had been readied to accept new tasks. The evaluation of obtained results show that if availability detection is done by our criteria, our approach incredibly detects available nodes and it improves more than 3% accuracy of availability in comparison with [21]. Also in our proposed approach seldom in some nodes the task is failed due to increase of local high priority processes (staying in waiting status until expiration).

On the other hand, this method spends very little time to resource discovery in comparison with MARS [1] and [14]. For instance, we compared our approach with MARS method and [14] in Figure 4. We only have a little time consuming in SA application to analyze Rough set Rules for matching task to node. Needless to say, if the number of task is less than available resources, our approach will take a very little time. When the number of tasks is more than available nodes (the nodes in Class A), SA must analyze the nodes in Class B to find proper nodes for desired tasks. In this case, elapsed time will little much time rather than previous state.

At the next time, we evaluated completion time between three methods in [1][14][21] and our approach. The purpose of completion time in this section is the spent times to execute all tasks on desired nodes until successfully are finished. To do this, we selected 10 type tasks that we bring in Table 1. Each job has a several copy that all copies must successfully is finished. We have showed this comparison between these four methods in Figure 5. As mentioned, we defined 250 nodes. At the beginning, we executed 100 tasks (10 copy of each task). Then we executed 200, 250, 500, and 1000 task (100 copy of each task) in each experiment. It needs to say, all task are considered as independent task and necessary condition was finishing all task in each experiment. According to Figure 5, it is clear that due to better reliability in our approach and GridProphet [21], these approaches have best completion time. By the way, due to use a smart method for applying nodes by Scheduler, for a large number of tasks, our method is better than others. Because according to our considered measures, heavy task (with much completion time and may be memory) always is matched to HP nodes. Probability theory based prediction method [14] in this comparison has worst completion time because selection measures for this method are imperfect and weak. At the end, we say that, if nodes are selected from Class A and at least 50% of selected nodes have been had High Priority (HP) then our methods will be best in this evaluation. This case is happen if number of available nodes be more than tasks.

### TABLE I. SOME CONSIDERED SAMPLE JOBS.

<table>
<thead>
<tr>
<th>Task-name</th>
<th>Size</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task1</td>
<td>40 KB</td>
<td>10 sec</td>
</tr>
<tr>
<td>Task2</td>
<td>110 KB</td>
<td>25 sec</td>
</tr>
<tr>
<td>Task3</td>
<td>230 KB</td>
<td>68 sec</td>
</tr>
<tr>
<td>Task4</td>
<td>364 KB</td>
<td>90 sec</td>
</tr>
<tr>
<td>Task5</td>
<td>640 KB</td>
<td>310 sec</td>
</tr>
<tr>
<td>Task6</td>
<td>1.23MB</td>
<td>330 sec</td>
</tr>
<tr>
<td>Task7</td>
<td>7.11MB</td>
<td>240 sec</td>
</tr>
<tr>
<td>Task8</td>
<td>8.16MB</td>
<td>290 sec</td>
</tr>
<tr>
<td>Task9</td>
<td>9.41MB</td>
<td>250 sec</td>
</tr>
<tr>
<td>Task10</td>
<td>10.33MB</td>
<td>760 sec</td>
</tr>
</tbody>
</table>
VI. CONCLUSION AND FUTURE WORKS

Computational Grid, as a dynamic environment, deals with large amount of resources and jobs. This approach uses the Rough Set Analysis to find useful rules between data and then using these rules by SA to detect the available resources at the moment and coming. By using rough set, it can to learn even with small number of samples. The experiment results show that prediction is effective, and the amount of existing efficient resources determined by online resource availability can significantly improve time complexity of resource finding and reliability, and so completion time. As a future work, we will design a fault-tolerant scheduling algorithm based on dynamic resource availability evaluation by using rough set and Neural Network (NN).

REFERENCES