A Grid Resource Allocation Method Based on Analytic Hierarchy Process

Rouhollah Golmohammadi and Hadi Shahriar Shahhoseini
Department of Electrical Engineering
Iran University of Science and Technology
Tehran, Iran
E-mail: rgolmohamadi@ee.iust.ac.ir, hshsh@iust.ac.ir

Abstract—Resource consumers and resource providers are the main infrastructures of the computational grid environment. The part that connects these components and makes this system a usable infrastructure is grid scheduler. Its role is to select the proper resource and sending its identifier to the user. It is obvious that, this process of resource selecting is a multi-criteria decision making problem, because of different qualitative and quantitative attributes affect it. In this paper, we introduce the method of analytic hierarchy process based resource allocation (ARA) to solve this problem. Simulation results show that the proposed method reduces task failure rate and mean waiting time to less than 2/10 and 5/100 of the previous methods respectively.

Keywords—analytic hierarchy process; computational grid; grid scheduler; multi-criteria decision making; resource allocation

I. INTRODUCTION

The concept of grid computing was introduced after parallel computing, in which the grid is a more complicated environment. It contains heterogeneous resources that are globally distributed. Such a widespread environment, without proper resource management could not be appropriately useable. So, it could be said that the necessary functions at service level of grid software infrastructure are task scheduling and resource management, where resource allocation and load balancing issues represent most common problems for grid systems [1].

The grid scheduler is responsible part for resource allocation. It follows grid policies and executes defined algorithm for resource selection. A sustainable resource allocation algorithm should be able to schedule resources in a way that as much as possible tasks be completed in their time and budget constraints.

The resource management system could have three schemes as centralized, distributed and hierarchical models [2]. Although the centralized model has some problems that are solved in distributed and hierarchical schemes, the easy management and simple deployment of this model put us to use it for local resource management system, as it is used in this paper. The cost of applying a non-centralized scheme to a little grid or a local resource management system does not seem to be reasonable.

The presented techniques in this area of work have considered static and dynamic methods. In static method [3-5] the information of tasks and performing platform thought to be fixed and defined in advance. The mapping decision of each task has been taken before simulation or execution time. While, the dynamic mapping [6-8] decides on tasks immediately or in a real-time fashion and the submission time of tasks is not predefined. Also the number of resources and their characteristics are not fixed necessarily. As a real grid environment has dynamic nature and tasks are received in an unpredictable manner, most of works have concentrated on dynamic scheduling.

It is obvious that tasks and resources are basic parts of the grid, each one is owned by users and resource providers respectively, and the scheduler is mainly a mapper from tasks to resources. When a task is sent to grid, the scheduler should decide on what resource the task could be applied to. This decision progress is a multi-criteria decision making (MCDM) problem. There are some types of solution for this area of problems. The analytic hierarchy process (AHP) [9] is one of the popular and widely used techniques in MCDM problems. This technique converts the complicated problems to simple ones by evaluating their reciprocal effects. The AHP had been used in many practices till now, and its suitable applicability had been declared [10]. So in this paper, a dynamic, AHP based resource allocation method (ARA) is proposed for grid resource management. In ARA, the users submit their tasks to grid scheduler and then, the scheduler defines an MCDM problem using the task properties and dynamic resource characteristics. To select a resource, the scheduler goes through AHP and finally sends the ID of selected one to the user.

In the rest of this paper, we investigate the related works in Section 2. In Section 3, we define the problem and explain the proposed method. The simulation characteristics and its results are represented in Section 4 and Finally Section 5 concludes this work.

II. RELATED WORKS

There are many researches and works on resource allocation in grid. It can be said that the time optimized and cost optimized allocation (TOA, COA) are two basic and standard algorithms to select resources, which were used in Nimrod/G system [11]. Many of the other works have used
these methods as the base of comparison. Min-Min and Max-Min are some other heuristic algorithms for resource allocation that were used in [12]. The last two algorithms consider all unmapped tasks during each mapping decision but TOA and COA only consider one task at a time. The proposed algorithm in this paper decides on tasks, one by one, as TOA and COA.

Beside with heuristic methods like TOA and COA, there are some non-heuristic methods, e.g. intelligent algorithms. Di Martino [13] represented a suboptimal scheduling method using genetic algorithm in 2003. The slow nature of genetic algorithm makes it an inappropriate way for resource allocation in such a grid environment where tasks are received subsequently each time. Yuan et al. [14] developed a neural network based method for grid resource allocation by predicting the availability of resources. Also in [15] a combined fuzzy-neural network model for workload prediction is proposed by Doulamis et al. Game theory is one of the other techniques that is used for resource allocation. Ghosh et al. [16] used game theory scheme for resource pricing in mobile grids. Game theoretic approaches are mostly used for resource pricing and try to have a good load distribution by changing the cost of resources from time to time. This makes the users with fewer budgets to use the cheaper resources to avoid load aggregation on best resources. Furthermore, Subrata et al. [8] used this method for QoS based resource selection and load balancing in grid.

The economic model is another one that was represented to solve the resource allocation problem [17-19]. It uses market mechanisms, e.g. different kinds of auction. In this model the resources have been assumed as merchandise, in which the resource providers and users are the sellers and consumers respectively. The scheduler is also a broker in this view. Buyya et al. [17] represent a comprehensive survey of economic models for resource management. Grosu and Das [18] compared different auction protocols and showed the double auction outperforms the other auction models. But, Kant and Grosu [19] attended to double auction and compared different kinds of this protocol. One of the most appropriate methods of auction model is presented in [20], which uses continuous double auction model and performs well from both user and resource perspective.

There are some works which have used multi-criteria approaches for workload scheduling [21,22] and performance prediction of resources [23,24]. They considered different possible allocations and defined a function based on the attributes of these mappings. In this paper, an MCDM problem is defined based on resource properties, but not the attributes of each allocation. It makes the resource selection problem extremely easier.

III. PROBLEM DEFINITION

A. Grid Environment Characteristics

A grid environment contains resource consumers (users), resource providers, resource scheduler(s) and grid information server (GIS). Each user in grid submits its tasks and its constraints to a scheduler. The scheduler then, receives needed information about resources from GIS and evaluates them for a capable resource for submitted task. The scheduler follows AHP to select proper resource.

In this paper, the defined grid environment consists a set of m resources \( R = \{ R_1, R_2, \ldots, R_m \} \) and a set of n users \( U = \{ U_1, U_2, \ldots, U_n \} \). Each user has a specific number of independent tasks and the users altogether have k tasks \( J = \{ J_1, J_2, \ldots, J_k \} \). These tasks are submitted to grid in time period of T (i.e. \( 0 \leq t_i < T \)), where T is the total time of simulation. Each task \( J_i \) is characterized by a four-tuple \( J_i = (l_i, b_i, d_i, s_i) \), in which \( l_i \) is the length of the \( i \)-th task and is specified by millions of instructions (MI), \( b_i \) determines the budget allocated to this task in GS and \( d_i \) represents the task deadline by which the user desires the task to be finished. Also \( s_i \) is the preference of time constraint over budget for task \( j \). This parameter is defined by user and the scheduler uses it for selecting one of the predefined overall preference matrices for AHP approach.

Each resource has also five characteristics defined by five-tuple \( R_i = (rc_i, rp_i, pe_i, wt_i, ru_i) \). The first two parameters are static but the others are dynamic and will change during the simulation time. \( rc_i \) is processing cost of this resource in GS/s. \( rp_i \) represents computational power of each processing element of \( R_i \) in terms of million instructions per second (MIPS). \( pe_i \) refers to number of processing elements of this resource and is defined itself by two-tuple \( pe_i = (tp_e, fp_e) \). \( tpe \) defines the total processing elements of \( R_i \) and \( fpe \) represents its free processing elements at time \( t \). \( wt_i \) is mean waiting time of tasks in \( R_i \) from receiving time to starting the execution. It can be calculated by (1).

\[
wt_i = \sum_{j=1}^{k} \left( st_j - rt_j \right) \beta_{ij}, \quad \beta_{ij} = \begin{cases} 1 & J_j \in R_i \\ 0 & \text{else} \end{cases} (1)
\]

In this equation \( st_j \) is the start time of task \( J_j \) and \( rt_j \) is receiving time of it on resource side. \( \beta_{ij} \) defines if task \( J_j \) is sent to resource \( R_i \) or not. The last characteristic of resources is \( ru_i \) and determines the resource utilization of \( R_i \) from simulation start time to present time (i) and it can be expressed as:

\[
ru_i = \sum_{j=1}^{k} \left( et_j - st_j \right) \beta_{ij}, \quad \beta_{ij} = \begin{cases} 1 & J_j \in R_i \\ 0 & \text{else} \end{cases} (2)
\]

In (2) \( et_j \) is the completion time of task \( J_j \) and the other parameters are defined as before. Each resource sends its dynamic information to GIS when its state changes. Grid scheduler, in resource selecting process, gets these information from GIS and follows AHP to select a proper resource. Also task information is sent directly from users to scheduler.

In this paper, each resource consists of one or more machines and each machine is also contains a number of processing elements (PE). The PEs are homogeneous in the same resource and each one can execute one task at a time. While, the resources are heterogeneous and geographically distributed and all of them use First Come First Serve (FCFS) method for task executing.
### B. Applying AHP to Grid

As it is explained, selecting the proper grid resource from all ones is an MCDM problem, and AHP is used for solving this problem. The contribution of this paper is introducing a new function to make overall preference matrices and also defining a parameter of $s_i$ for each task to indicate that how much the time or budget constraints are important for the user.

The flowchart of the proposed algorithm is represented in Fig. 1. It shows the connections of grid scheduler and the process that is defined for resource scheduling. The algorithm is started when a user sends its task properties to grid scheduler. Then the scheduler selects the associated predefined Relative Value Vectors (RVV) based on task’s $s_i$. These vectors are defined in this paper as represented in Table I. These values are calculated by AHP from certain overall preference matrices. The matrices are formed by pairwise comparison of attributes.

After that, the scheduler recalls resource information from GIS and uses (3) to generate the overall preference matrices of resources for each resource characteristic, e.g. mean waiting time of resources.

$$c_{ij} = \begin{cases} K & : A_i > A_j \\ 1 & : A_i = A_j \\ K^{-1} & : A_i < A_j \end{cases}$$

(3)

$\eta$ is a scaled number from 1 to 9 and indicates pairwise comparison ratio of $i$-th resource by $j$-th resource. $A_i$ and $A_j$ define the value of parameter $A$ of resource $i$ and $j$, respectively. The value of $K$ is calculated as follows:

$$K = \left[ \eta \times \frac{A_j - A_i}{A_{max} - A_{min}} + 1 \right]$$

(4)

where $A_{min}$ and $A_{max}$ are minimum and maximum values of parameter $A$. While the output of equation (4) is an integer number from 1 to 9, the coefficient of $\eta$ in this equation should be a real number in the range of [8, 9). Using 8.0 makes the preference of 9 just belonging to $A_{max}$ in comparison with $A_{min}$. Further tests showed that the best value for $\eta$ is 8.2. Equation (3) is usable if preference of the resource $i$ to $j$ is defined by a larger value of $A_i$ to $A_j$, e.g. the total number of the processing elements. When the lesser value of $A_i$ stands for preference of resource $i$ (e.g. mean waiting time), equation (5) should be used as an alternative of (3).

$$c_{ij} = \begin{cases} K & : A_i < A_j \\ 1 & : A_i = A_j \\ K^{-1} & : A_i > A_j \end{cases}$$

(5)

After calculating all $c_{ij}$ values, the overall preference matrices will be formed. Table II shows the generated overall preference matrix for cost of resources. The last column is its eigenvector, which is calculated approximately by getting the geometric mean of each row. Then the Option Performance Matrix (OPM) is made by putting the eigenvectors of all characteristics together. Technically, the respective preference of resources is calculated by post-multiplying the OPM by selected RVV. The obtained vector is named Value for Money (VFM). The VFM gives a number to each resource. This number shows the preference of respected resource, i.e. the resource with a greater number is more appropriate. Then the resources that cannot maintain the task’s time or budget constraints are removed. Next, one of the two best resources should be selected by a random selection process. This is for removing the undesirable effects of negligible differences in VFM. To make it a reasonable process, we generate a random number for each of these two resources and multiply it by number of total processing elements of them. The random number is uniformly distributed within the range of $[0, 1]$. Also, to increase the probability of selecting the first resource, its random number will be multiplied by 2.
IV. SIMULATION AND RESULTS

In order to evaluate the efficiency of the proposed method, we use GridSim [25]; a computational grid simulator. The defined grid consists of 20 users 8 resources (total number of 60 PEs). The simulated resources are as WWG standard testsbed [25,26] and their characteristics are shown in Table III. Each user has 50 numbers of tasks (1000 tasks from all users) and the time interval between two task submissions has Poisson distribution. Changing the average of distribution causes to different system loads, which is used for system analysis. The lengths of the tasks are considered as an integer random number, uniformly distributed in the range of [10000, 100000]. The presented results for each value of system load are the average of 50 times of simulation.

In this experiment we compare the presented method with COA, TOA and TCOA algorithms. These are straightforward, basic and easy to implement methods were used in many works for performance evaluation. In COA algorithm the scheduler tries to decrease the cost of task execution, in which the tasks be completed in their time and budget constraints. TOA algorithm perpends to optimize the time of task completion. In TCOA first a time optimization process is done, and then a cost optimization process is implemented on the best selected resources. To have a fair comparison, we use the same attributes as in ARA for compared methods (i.e. resource load, PE numbers and mean waiting time).

TABLE II. OVERALL PREFERENCE MATRIX FOR COST OF RESOURCES

<table>
<thead>
<tr>
<th>re</th>
<th>R_1</th>
<th>R_2</th>
<th>R_3</th>
<th>R_4</th>
<th>R_5</th>
<th>R_6</th>
<th>R_7</th>
<th>eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_1</td>
<td>1/5</td>
<td>1/5</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/6 0.2043</td>
</tr>
<tr>
<td>R_2</td>
<td>1/5</td>
<td>1/5</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/6 0.3145</td>
</tr>
<tr>
<td>R_3</td>
<td>1/5</td>
<td>1/5</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/6 0.2148</td>
</tr>
<tr>
<td>R_4</td>
<td>1/5</td>
<td>1/5</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/6 0.05657</td>
</tr>
<tr>
<td>R_5</td>
<td>1/5</td>
<td>1/5</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/6 0.08585</td>
</tr>
<tr>
<td>R_6</td>
<td>1/5</td>
<td>1/5</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/6 0.31285</td>
</tr>
<tr>
<td>R_7</td>
<td>1/5</td>
<td>1/5</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/6 0.03859</td>
</tr>
<tr>
<td>R_8</td>
<td>1/5</td>
<td>1/5</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/6 0.13545</td>
</tr>
</tbody>
</table>

consistency ratio: 0.02263

TABLE III. SIMULATED CHARACTERISTICS OF RESOURCES [25, 26]

<table>
<thead>
<tr>
<th>Resource Type, Node OS, No of PEs</th>
<th>SPEC/ MIPS Rating</th>
<th>Price (G$/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compaq, AlphaServer, CPU, OSF1, 4</td>
<td>515 8</td>
<td>377 3</td>
</tr>
<tr>
<td>Sun, Ultra, Solaris, 4</td>
<td></td>
<td>377 3</td>
</tr>
<tr>
<td>Sun, Ultra, Solaris, 4</td>
<td></td>
<td>377 3</td>
</tr>
<tr>
<td>Intel, PentiumVC820, Linux, 2</td>
<td>380 2</td>
<td>410 5</td>
</tr>
<tr>
<td>SGI, Origin 3200, IRIX, 6</td>
<td></td>
<td>410 4</td>
</tr>
<tr>
<td>SGI, Origin 3200, IRIX, 16</td>
<td></td>
<td>410 4</td>
</tr>
<tr>
<td>Intel, PentiumVC820, Linux, 2</td>
<td>380 1</td>
<td>410 6</td>
</tr>
<tr>
<td>SGI, Origin 3200, IRIX, 4</td>
<td></td>
<td>410 6</td>
</tr>
<tr>
<td>Sun, Ultra, Solaris, 8</td>
<td></td>
<td>377 3</td>
</tr>
</tbody>
</table>

Fig. 2 shows the mean waiting time of tasks (from submitting to execution start time) for different values of system load. As it can be seen, TOA, TCOA and ARA algorithms have no waiting time for loads less than 0.7, and for load of 0.7 and more the application of ARA is so better than the others. Also COA has unsuitable response because of attending the cost optimization. The results show that for high system loads (0.7 to 0.9), the average of mean waiting time in ARA is decreased to about 1/150, 1/25 and 1/20 of COA, TOA and TCOA algorithms respectively. The smaller mean waiting time means tasks spend less time in execution queue, and it can be concluded that ARA makes a good distribution of tasks between resources.

The failure rate of tasks is the next metric used to evaluate the resource allocation method. In Fig. 3 the failure rate of these methods is shown. In lower loads the considerable time period between tasks submissions causes resources to have enough time to process the received tasks, so most of tasks have the chance to execute successfully. Also it can be seen that there is a significant relevance between mean waiting time and the failure rate of tasks. As it is expected the less mean waiting time is caused to more successfully executed tasks (or less number of failed tasks).

As shown in Fig. 3, ARA decreases the average of failure rate to 1/60, 1/4 and 2/7 of COA, TOA and TCOA algorithms respectively. These decrements for high loads are 1/48, 1/7 and 2/15. For all system loads the task failure rate of ARA algorithm is less than 0.5%, i.e. from each 1000 tasks more than 995 tasks succeed to perform in their deadlines.
In Fig. 4 we investigate the total revenue of resource providers (or the total expenditure of users). It shows that COA algorithm acts better from user perspective, especially in lower loads. On the other hand, TOA and TCOA algorithms increase the cost, because they try to optimize the time and so send the tasks to fastest and subsequently more expensive resources. However, our method trades off between the revenue of resource providers and the expenditure of users.

As, in higher loads, the time interval between task receiving is lower and the scheduler has to send some tasks to more expensive resources to maintain the deadlines in COA algorithm, the total revenue of resources increase. The same reason cause to reduce the total cost for TOA and TCOA algorithms. The proper resource allocation and dependency of task distribution to resource utilization in ARA algorithm caused the total cost to be almost equal for all system loads.

In order to show better the performance of ARA method, we compare it with CDA method. It is one of the newly developed methods based on continuous double auction model. Fig. 5 shows the failure rate of tasks for these methods. As it can be seen in this figure, for loads of 0.6 and less, the failure rate for CDA method is 0.0% but for our method it is about 0.1%. However for higher loads the ARA outperforms the CDA. The average failure rate of tasks for ARA is about 0.29 of CDA for high loads.

![Figure 4. The failure rate of tasks for compared methods](image)

![Figure 5. The failure rate of tasks for ARA and CDA methods](image)

V. CONCLUSION

The resource scheduler is a part that makes grid a usable environment, so the applied algorithm to this part has a vital role in resource allocation procedure. In this paper we used analytic hierarchy process to select proper resources based on user requirements and resource conditions. The scheduler in the represented method gets the resource information from GIS and uses it to properly distribute the system load between grid resources. It causes tasks to perform in their time and cost constraints.

To evaluate the performance of this method, the GridSim simulator was used and the simulation results were compared with the results of three basic and standard methods and also a newly developed method based on auction model. The results show that our method improves the value of task successful execution rate and reduces the mean waiting time of tasks.

The future works can be concentrated on applying this method to hierarchical structure of grid resource management system, and considering the effects of communication network and resource failure rate in simulations.

REFERENCES


