Grid Computing Scheduling based on Neural Networks

M. Faheem
Prof., Computer and Control Dept., Faculty of Engineering, Atif University, Saudi Arabia
m_t_faheem@yahoo.com

E. Sallam, T. Eltobely
Assistant Prof., Computer and Control Dept., Faculty of Engineering, Tanta University, Egypt
sallam_9@hotmail.com, tarek@ictp.edu.eg

H. El-Ghaish
Demonstrator, Computer and Control Dept., Faculty of Engineering, Tanta University, Egypt
dr_h_elghaish@hotmail.com

ABSTRACT

In Grid Computing environment, there are many of idle resources that compete for similar tasks. In order to gain maximum resource utilization while minimizing task completion time, a time optimization scheduling algorithm based on back-propagation neural network is proposed in this paper. The proposed algorithm predicts the submitted task run time by training neural network through a training set of samples. Each sample consists of several parameters. One of these parameters is an output parameter which helps in getting the predicted task run time. This predicted task run time provides reliable information for task scheduling and resource management. To show the usefulness of our work a comparison between the proposed scheduling algorithm and the Min-Min scheduling algorithm using the GridSim simulation toolkit is introduced. The proposed algorithm is applied in compute-intensive, read-intensive and write-intensive applications. Experimental results show that the proposed algorithm is better than its Min-Min counterpart in the compute-intensive, read-intensive and write-intensive applications specially when the number of jobs is increased in the experiment.

Keywords: Grid Computing, job scheduling, simulation, Back-propagation neural network

1 INTRODUCTION

Grid Computing [5] and peer-to-peer (P2P) computing [2] networks are emerging as next generation computing platforms for solving large-scale computational and data intensive problems in science, engineering, and commerce. The Grid Computing systems are different from the traditional parallel systems. In Grid Computing environments, the resources are usually geographically distributed in multiple administrative domains, and with different characteristics. They enable the sharing, selection and aggregation of a wide variety of geographically distributed resources including supercomputers, storage systems, databases, data sources, and specialized devices owned by different organizations. However, resource management and application scheduling are a complex undertaking due to large-scale heterogeneity present in resources, management policies, users, and applications requirements in these environments [9]. The resources are owned and managed by different organizations with different access policies and cost models that vary with time, users, and priorities. Different applications have different computational models that vary with the nature of the problem. The computational economy framework provides a mechanism for regulating the supply-and-demand for resources and allocating them to applications based on the deadline and budget constraints [8]. Resource management systems need to provide mechanisms and tools that realize the goals of both service providers and consumers. It offers an incentive to resource owners for sharing resources on the Grid and end-users trade-off between the timeframe for result delivery and computational expenses. The resource consumers need a utility model to represent their resource demand and preferences, and brokers that automatically generate strategies for choosing providers based on the utility model. A number of projects are investigating scheduling on distributed systems [9]. Therefore, a grid resource broker (scheduler) must be concerned with response time, system resource utilization, user satisfaction, and so on [11]. With the above applications in mind, we an effort to carry out scheduling algorithm to determine
the “queuing order” and the “processor assignment” for a given task is made so that the demand quality of service(QoS) parameters, i.e., the task deadlines, are satisfied as much as possible. Towards this goal, we have used evolutionary algorithm like a time optimization based on back-propagation neural network (BPNN). The ability of the proposed scheduling algorithm is demonstrated by implementing it within the economic grid resource broker simulator that is built using the GridSim toolkit [7]. The rest of the paper is organized as follows: Section 2 reviews related work in grid scheduling. In Section 3, we briefly explain the runtime estimation model using BPNN. Section 4 describes the complete steps of the proposed scheduling algorithm. Section 5 gives information about GridSim toolkit and describes the experimental simulation of the proposed algorithm. Section 6 presents our experimental results of the proposed algorithm and these results are compared with conventional Min-Min algorithm. Finally, Section 7 makes some concluding remarks and points to the future work.

2 RELATED WORK

The research in the area of job scheduling for parallel systems, high performance clusters and Grid Computing [5, 12, 13, 3] can be classified into two categories: developing scheduling algorithms at a particular system-level, while others interested in the application-level. System-level scheduling maximizes process throughput or the overall utilization rate of the machines [15]. While scheduling at application- level, generally, applications are described as a group of jobs and the parallel scheduling problem becomes a problem of scheduling these jobs of an application across parallel machines in order to meet various objective functions. The application-level scheduler is able to use the prediction to get better executing performance. In high performance distributed computing domains, the problem of finding an optimal solution to minimize the total execution time of a set of jobs has been shown to be NP complete[15]. Therefore, heuristics have been developed to get near optimal scheduling solutions. However, in large-scale Grid environments, besides minimizing the overall execution time, users often have some special requirements. In this work, we present job scheduling heuristic based on improved back-propagation BPNN to be used in a simulation environment. In practice, the runtime of a task is affected by many factors, so linear model cannot describe system features well. The neural networks have well non-linear mapping capability, fast parallel processing ability, powerfully self-learning and self organizing ability and so on. BPNN have simple structure, powerful simulation ability and implement conveniently etc. With a view to several factors like the resource load, the task scale, the application type and the number of committed jobs we present a dynamic predicting system to predict the runtime of task on resources. It can provide reliable information for the task scheduling and the resource management in grid computing environment. The Min-Min heuristics described in [10] is becoming the benchmark of scheduling problems. However, it does not take into account some factors such as the resource load, the type of submitted application, the task scale and the number of committed jobs which affects its effectiveness in a Grid.

3 NEURAL NETWORK BASED GRID RESOURCE BROKER MODEL

The grid computing environment has many components. In our work the resource broker component will be focused as shown in the next sections.

3.1 Economic based Grid Resource Broker Architecture

A grid scheduler (resource broker) acts as an interface between the user and distributed resources and hides the complexities of Grid Computing [4]. It performs resource discovery, negotiates for access costs using trading services, maps jobs to resources (scheduling), stages the application and data for processing (deployment), starts job execution, and finally gathers the results. It is also responsible for monitoring and tracking application execution progress along with adapting the changes in Grid runtime environment, variation in resource sharing availability, and failures. The users create an experiment which contains application specification (a set of Gridlets that represent application jobs with different processing) and quality of service requirements (deadline and budget constraints with the scheduling algorithm).

Figure1: Economic-based Grid resource broker architecture and its interaction with other entities.
We created two entities that simulate the users and the brokers. When simulated, each user entity having its own application and quality of service requirements creates its own instance of the broker for scheduling Gridlets on resources and other entities as summarized in the following:

- **User:** Each instance of the user entity represents a Grid user.
- **Broker:** Every job of a user is first submitted to its broker and the broker then schedules the parametric tasks according to the proposed time optimization scheduling algorithm.
- **Resource:** Each instance of the Resource entity represents a Grid resource the resource speed and the job execution time can be defined in terms of the ratings of standard benchmarks such as MIPS and SPEC.
- **Grid information service (GIS):** Providing resource registration services and keeping track of a list of resources available in the Grid.
- **Input and output:** The flow of information among the GridSim entities happens via their input and output entities.
- **Each independent job may require varying processing time and input files size.** Such tasks can be created and their requirements are defined through Gridlet objects. The architecture of interaction of entities is shown in Figure 1.

### 3.2 TASK RUN Time Estimation Model based on BPNN

Task run time is affected by several factors such as task and resource characteristics. These task characteristics like the task type, task scale which represents the number of Gridlets in the experiment, number of committed jobs, program length, Gridlet input file size, Gridlet output file size and so on. Some performances of a resource have a huge influence too, as the resource speed (resource capacity), number of processors in a resource, CPU availability, resource load, the storage volume and the type of operating system etc. In this paper, we choose an input vector \( x \) of seven items from which represent respectively resource speed in terms of million instruction per second (MIPS), number of processors, resource load in MIPS, number of committed jobs, the program length in terms of million instruction (MI), the input file size and the output file in terms of mega byte (MB) to the task run time estimation model because of the following reasons [6]:

- Can objectively reflect the influence on the runtime.
- Should be independent each other in measurement and control.
- Be easy to measure or estimate, and low overhead.
- Can efficiently reflect the dynamic behavior of both resources and tasks.

We denote as \( y \), the runtime parameter vector, where the element \( y \) represents the expected task run time. The Runtime Estimation model is responsible for estimating vector \( y \) from \( x \), through a non-linear model \( y=f(x) \). In order to have a generic runtime estimation model as shown in figure 2, which can be applied to any type of application, modeling of the unknown function \( f(x) \) is performed by the backpropagation neural network [14, 1, 6].

### 3.3 Learning and Testing Sets in BPNN

The Run Time Estimation model is approximated with a three-layer feed-forward neural network with back-propagation learning. A neural network is created with 7 inputs, 5 hidden units with sigmoid activation function and a linear output unit. We trained the network by 4000 samples, 3000 for training and 1000 for testing. In particular, neural networks provide an approximation of function \( f(x) \), through two phases: during the first phase the input \( x \) is presented and propagated through the network to compute the output value \( y \). This output is compared with its desired value, \( d \), resulting in an error signal \( \delta \). The second phase involves a backward pass through the network during which the error signal is passed to each unit in the network and appropriate weight changes are calculated. Training is stopped when the error does not decrease any more or we reach the maximum epoch. This error depends on the number of hidden units and the activation function.

![Figure 2: Task Run Time Estimation model and Time Optimization based on BPNN scheduling algorithm](image.png)

### 4 THE COMPLETE PROPOSED TIME OPTIMIZATION based on BPNN SCHEDULING ALGORITHM

In order to evaluate the effectiveness of the proposed algorithm, we develop a complete Grid Computing environment using the Java programming language where the proposed scheduling algorithm is...
Encapsulated within the broker scheduling heuristic scheduling algorithm steps are summarized as shown in figure 3.

1. RESOURCE DISCOVERY: Identify resources that can be used in this execution with their capability through the Grid Information Service.
2. RESOURCE TRADING: Identify cost of each of the resources in terms of CPU cost per second and capability to be delivered per cost-unit.
3. If the user supplies D and B factors, then determine the absolute deadline and budget based on the capability and cost of resources and user’s requirements according to the following formulas [11]

   \[
   \text{Deadline (DL)} = T_{\text{Min}} + D_{\text{Factor}} \times (T_{\text{MAX}} - T_{\text{Min}}) \quad (1)
   \]

   \[
   \text{Budget (BD)} = C_{\text{Min}} + B_{\text{Factor}} \times (C_{\text{MAX}} - C_{\text{Min}}) \quad (2)
   \]

   Where,
   - \( T_{\text{Min}} \) and \( C_{\text{Min}} \): the time and cost required to process all the jobs, in parallel.
   - \( T_{\text{MAX}} \) and \( C_{\text{MAX}} \): the time and cost required to process all the jobs, serially.
   - An application with \( D_{\text{Factor}} \) and \( B_{\text{Factor}} \) greater than one would always be completed as long as some resources are available with minimal user-share throughout the deadline.
4. SCHEDULING: Repeat while there exist unprocessed jobs in application job list with a delay of scheduling event period or occurrence of an event AND the time and process expenses are within deadline and budget limits:
   a. For each resource do the following:
      i. Calculate the expected processing cost for the unprocessed job and if its cost is less than or equal the remaining budget for the job go to step ii. Else do nothing and check another resource.
      ii. According to the Task Run Time Estimation Model based on BPNN we can predict the run time of any given task, and then add this resource to Broker Resources To Use list.
      iii. Add both the Broker Resources To Use list and their corresponding expected run time in another list called Broker Resource Gridlet Time.
   b. Sort in ascending order the Broker Resource Gridlet Time list according to the expected run time.
   c. For each resource in the Broker Resource Gridlet Time list compute the task completion time. If the time is less than or equal the experiment deadline, we should assign the job to the resource ready queue. Else go to another resource.
   d. For each resource in order:
      i. If the number of jobs currently assigned to a resource is less than the predicted number of jobs that a resource can consume, assign more jobs from unassigned job queue or from the most expensive machines based on job state and feasibility. Assign job to a resource only when there is enough budget available.
      ii. Alternatively, if a resource has more jobs than it can complete by the deadline, move those extra jobs to unassigned job queue.
5. DISPATCHER with Policy

   The dispatcher takes care of submission of jobs to remote machines with submission and resource policy and constraints depending on resource type (time or shared).

Figure 3: The Time Optimization based on BPNN Scheduling Algorithm

5 BUILDING SIMULATIONS WITH GRIDSIM TOOLKIT

In order to simulate resource scheduling in Grid environment, we used the GridSim toolkit (V.4.2) [11] to evaluate the proposed scheduling algorithm and compare it with conventional Min-Min algorithm.

5.1 The GridSim Toolkit

   In a Grid environment, it is hard and even impossible to perform scheduler performance evaluation in a repeatable and controllable manner as resources and users are distributed across multiple organizations with their own policies. To overcome this limitation, a Java-based discrete-event Grid simulation toolkit called GridSim is developed. The toolkit supports modeling and simulation of heterogeneous Grid resources, users and application models. GridSim supports entities for simulation of single processor and multiprocessor, heterogeneous resources. It supports entities that simulate networks used for communication among resources. GridSim based simulations contain entities for the users, brokers, resources, information service, statistics, and network based I/O. The design and implementation issues of these GridSim entities are
discussed below. All references should appear together at the end of the paper. The full references are cited in a numbered list of the following style:

5.2 Resource Modeling
A Grid resource contains one or more machines. Similarly, a Machine contains one or more PEs (Processing Elements or CPUs). In this experiment, we are simulating ten Grid resources with machines that contain PEs.

5.3 User And Job Modeling
In GridSim, jobs can be created and their requirements are defined through Gridlet objects. A Gridlet is a package that contains all the information related to the job and its execution management details such as job length expressed in MIPS, disk I/O operations, the size of input and output files, and the job originator. A Grid user has many Gridlets to be processed.

5.4 Performance Evaluation Metrics
In considering the metrics to be measured in our simulation study, we computed the make-span (MS) of every application. It is the completion time required to process all the jobs in an application’s set. The metric was calculated as follows:

\[ MS = \max\{ Completion\ Time_i, \ i \in [0, \ldots, n]\} - \text{Submit Time} \] (3)

6 EXPERIMENTAL RESULTS AND DISCUSSION
In this section, we analyze the performance of the proposed BPNN based time optimization scheduling algorithm and the conventional Min-Min scheduling algorithm, which is one of the known benchmarks in this field. Independent experiments were performed for compute-intensive, read-intensive and write-intensive applications. To ensure the stability of the achieved results against the changes in the model parameters, different experiments are implemented with different values in the following parameters: the task scale (TS) which represents the number of Gridlets (GL) in the experiment, the deadline (DL) and budget (BD) constraints, the Gridlet length (GLL), the Gridlet input file (GLIF) size, and the Gridlet output file (GLOF) size. Table 2 and Table 3 show the values of these parameters in the implemented experiments. The values of DL and BD in table 1 are computed from Eq. 1 and Eq. 2. The rest of parameters in table 2 are proposed empirically according to the type of each experiment. Where in compute intensive applications the value of GLL is extremely larger than GLIF and GLOF. Similarly in read-intensive (write-intensive) applications the value of GLIF (GLOF) is the greatest. It is worth to mention that to evaluate the performance of each experiment with different values of GLL, GLIF and GLOF with range of tolerance. For example, the value of GLL in the compute-intensive applications is 5000 with tolerance between 1% and 7%, where the value of this parameter is selected in each experiment randomly within this range.

### Table 1: Deadline (DL) and budget (BD) constraints for each experiment.

<table>
<thead>
<tr>
<th>TS (ms)</th>
<th>DL (ms)</th>
<th>BD ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>4.8E07</td>
<td>4.6E10</td>
</tr>
<tr>
<td>150</td>
<td>7.0E07</td>
<td>7.0E10</td>
</tr>
<tr>
<td>200</td>
<td>9.6E07</td>
<td>9.4E10</td>
</tr>
<tr>
<td>250</td>
<td>1.2E08</td>
<td>1.2E11</td>
</tr>
<tr>
<td>300</td>
<td>1.4E08</td>
<td>1.4E11</td>
</tr>
<tr>
<td>350</td>
<td>1.6E08</td>
<td>1.6E11</td>
</tr>
<tr>
<td>400</td>
<td>1.9E08</td>
<td>1.9E11</td>
</tr>
<tr>
<td>450</td>
<td>2.2E08</td>
<td>2.0E11</td>
</tr>
<tr>
<td>500</td>
<td>2.5E08</td>
<td>2.4E11</td>
</tr>
<tr>
<td>550</td>
<td>2.6E08</td>
<td>2.6E11</td>
</tr>
<tr>
<td>600</td>
<td>2.9E08</td>
<td>2.8E11</td>
</tr>
<tr>
<td>650</td>
<td>3.1E08</td>
<td>3.0E11</td>
</tr>
<tr>
<td>700</td>
<td>3.3E08</td>
<td>3.3E11</td>
</tr>
</tbody>
</table>

### Table 2: GLL, GLIF size and GLOF size for compute-intensive, read-intensive and write-intensive applications.

<table>
<thead>
<tr>
<th>App.</th>
<th>GLL(MI)</th>
<th>GLIF(MB)</th>
<th>GLOF(MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute-intensive</td>
<td>50000+ (1%: 7%)</td>
<td>73 + (0.5%: 1.6%)</td>
<td>73+ (0.05%: 1.6%)</td>
</tr>
<tr>
<td>Read-intensive</td>
<td>9726+ (8%: 10%)</td>
<td>5000 + (0.1%: 100%)</td>
<td>100+ (2%: 100%)</td>
</tr>
<tr>
<td>Write-intensive</td>
<td>6141 + (8%: 10%)</td>
<td>100+ (2%: 100%)</td>
<td>5000 + (0.1%:100%)</td>
</tr>
</tbody>
</table>

Figure 4: Compute-Intensive applications
The relationship between the total completion time, Make-span (MS), of the experiment and the task scale which represent the number of Gridlets in the experiment in each application is showed in Fig [4, 12, 6]. In all experiments, users submit applications randomly based on a Poisson arrival. As shown in Fig [4, 12, 6]. When the number of Gridlet exceeds 200, the completion time of the tasks in the proposed time optimization based upon BPNN algorithm is lower than the completion time in the Min-Min algorithm in the case of the compute-intensive, read-intensive and write-intensive applications. This improvement due to the fact that the proposed algorithm takes into account some factors as the resource load, the type of submitted application, the task scale and the number of committed jobs which affect its effectiveness in a Grid and help the scheduler to take the right scheduling decision than the Min-Min algorithm.

7 CONCLUSIONS AND FUTURE WORK

In this paper we present the problem of mapping tasks to grid resources through the BPNN based time optimization scheduling algorithm. We show that the grid scheduler should consider several factors like task characteristics and resource performance in a simulated Grid environment. We have compared the results obtained through the proposed BPNN based scheduling algorithm with the ones obtained from the Min-Min scheduling algorithm using GridSim toolkit. We observe that the proposed algorithm give results better than Min-Min counterpart specially when the number of Gridlets is increased. In fact, the performance of a job scheduling can be affected by various factors. Therefore, an advanced job scheduler for Grid must be concerned with network status and available bandwidth and so on. In the future work, we will focus on the research of impact of network bandwidth, memory availability and other factors upon a Grid scheduler.

8 REFERENCES


