RESOURCE ALLOCATION ANALYSIS MODEL BASED ON GRID ENVIRONMENT

HUEY-MING LEE, JIN-SHIEH SU AND CHIA-HSIEN CHUNG

Department of Information Management
Chinese Culture University
No. 55, Hwa-Kung Road, Yang-Ming-San, Taipei 11114, Taiwan
hmlee@faculty.pccu.edu.tw; jonguser@gmail.com

Received December 2009; revised June 2010

ABSTRACT. In this paper, we propose a performance analysis of resource allocation model for grid resources under the grid computing environment. By the proposed model, we can make all grid nodes be load-balancing. The proposed model can detect the grid nodes resource status about CPU usage and memory usage, analyze the grid nodes resources load grades, and make the plans and allocations of the resources of collaborative nodes optimize. Via implementing this model, we can enhance the grid environment performance efficiently.

Keywords: Grid computing, Resource allocation

1. Introduction. The term “Grid” was coined in the mid 1990s to denote a proposed distributed computing infrastructure for advanced science and engineering [3]. In grid environment, user may access the computational resources at many sites [2,5]. The functions of information systems based on grid computing architectures are resources (e.g., CPUs, memory, storages, etc.) sharing, collaborative processing, reliable and secure connection, etc [3,7-9]. However, each resource of coordinate nodes in the grid environment, (e.g., CPU loading, memory usage, etc.) changes dynamically. Therefore, how to optimize these resources usages is an important issue.

Foster et al. [3] presented grid resource allocation and management (GRAM). GRAM simplifies the usage of remote systems by providing a single standard interface for requesting and using remote system resources for the execution of “jobs”. The most common usage of GRAM is a remote job submission and control. This is typically used to support distributed computing applications. GRAM is designed to provide a single common protocol and application programming interface (API) for requesting and using remote system resources, by providing a uniform, flexible interface to local job scheduling systems. Miller et al. [12] proposed Paradyn parallel performance measurement tools which can identify the heaviest loading process by heuristic method and find out the bottleneck point. Lee et al. [7] proposed a dynamic supervising model which can utilize the grid resources, e.g., CPU, storages, etc., more flexible and optimal. Lee et al. [8] presented an optimal analyzing resources model that can receive the information about CPU usage, number of running jobs of each grid resource node to achieve load balancing and make the plans and allocations of the resources of collaborated nodes optimize. Iosup et al. [6] proposed synthetic workloads to evaluate performance on grid. Lee et al. [9] proposed a process schedule analyzing model, when a new job is entered to be processed and the load of this node is heavy, the model can schedule some jobs to the other nodes to process by the data base of grid information. Sanjay and Vadhiyar [13] presented a performance modeling of parallel applications for grid scheduling that developed a comprehensive set
of performance modeling strategies for predicting execution times of parallel applications. The strategies adapt to changing network and CPU loads on the grid resources. Lee et al. [10] presented a resource performance analysis model under the grid computing environment.

In [16], Ye et al. proposed a new method which identifies actual effective attacks against tasks, and quantitatively evaluates the risks that every grid service will attack the tasks based on the identified attacks to protect the confidentiality and integrity of grid tasks, in open grid services architecture.

In the scheduling criteria, Silberschatz et al. [14] proposed CPU utilization, throughput, turnaround time, waiting time and response time. Li et al. [11] proposed the workload characterize include system utilization, job arrival rate and inter arrival time, job cancellation rate, job size, job run time, memory usage and user/group behavior. In [15], Xhafa et al. proposed genetic algorithms (GAs) based schedulers for efficiently allocating jobs to resources in a grid system.

In order to make the plans and allocations of the resources of collaborated nodes optimize, we have to detect the related status of collaborative nodes and analyze these status to calculate the best allocation of jobs.

In this paper, we propose a performance of resource allocation model for grid resource under the grid computing environment. By this model, we can detect the grid nodes resources status about CPU usage, memory usage, and analyze the grid nodes resources load grades, and make the plans and allocations of the resources of collaborated nodes optimize. There are three modules in the proposed model, namely, resource detecting module, resource evaluate module and resource assignment module. According to the result of experiment, the mechanism can achieve the best resources allocation, and enhance the overall grid computing performance.

2. Framework of the Proposed Model. In order to make the performance analysis model more complete and the utilizing resources more flexible and optimal, we introduced the performance grade in this model. The performance grade is a rating of collaborative node load grade. By the performance grade, we can settle the priority of collaborative node in the work schedules.

We built grid node capacity information locally. It contains node name, CPU speed, CPU usage, memory size, memory usage and disk capacity, etc., as shown in Table 1.

<table>
<thead>
<tr>
<th>Node-name</th>
<th>CPU-speed</th>
<th>CPU-usage</th>
<th>Memory-size</th>
<th>Memory-usage</th>
<th>Disk-capacity</th>
</tr>
</thead>
</table>

In Table 1, Node-name is name of this node, CPU-usage is total CPU usage in this period, Memory-usage is total memory usage in this period, Disk-capacity is available disk space.

This information can be used to select a correct grid node to process new job. We present a resources allocation analysis model (RAAM) based on grid computing architecture. The circumstance of the proposed model is as shown in Figure 1.
Based on [1], the proposed model was built on the grid node, saying $M_0$, as shown in Figure 2.

There are three modules in the proposed model, namely, resource detecting module (RDM), resource allocating module (RALM) and resource analysis module (RANM), as shown in Figure 3.

The functions of these modules are as the follows:

- **Resource detecting module (RDM):** It can detect/monitor the collaborative grid nodes resources status, and receive the node related resource information, (e.g., disk size, CPU speed, CPU usage, Memory size, Memory usage, etc.)

- **Resource analysis module (RANM):** It can analyze the collaborative grid nodes resources load grades by fuzzy inference based on the status of resource detecting module. The RANM sends the analyzed results to the resource allocating module as a basis of the plan.

- **Resource allocating module (RALM):** It can optimize the planning and allocating of the collaborative grid nodes resources. This module provides three different modes of allocation including average, optimize cost and optimize computing performance.
2.1. **Resource detecting module (RDM).** The RDM comprises two components which are job receiving component (JRC) and resource detecting component (RDC), as shown in Figure 4.

![Figure 4. Framework of the RDM](image)

The job receiving component (JRC) deals with the user’s job request, as shown in Table 2, and sends it to resource detecting component (RDC). The RDC will search and detect the collaborative nodes and then collect the related information of collaborative nodes (e.g., CPU speed, CPU usage, memory size, memory usage, disk size, etc., as shown in Table 1), send the detected messages to the resource analysis module, then, send the job requests to the resource allocating module.

**Table 2.** User’s job request message

| Job # | File name | Job amount |

2.2. **Resource analysis module (RANM).** The resource analysis module (RANM) comprises three components, which are data collection component (DCC), resource price component (RPC), loading analysis component (LAC), as shown in Figure 5.

![Figure 5. Framework of the RANM](image)

- The data collection component (DCC) integrates and collects the related information, e.g., memory utility rate, CPU loading, etc., of collaborative nodes from RDC and then
sends the integrated information to RPC and LAC, as the evaluation data. In addition, DCC also receives the information of evaluation from RPC and LAC, and then, sends it to the resource allocation module (RALM), as the basis for the allocation of planning.

- The resource price component (RPC) evaluates the price of collaborative node by its hardware information (e.g., CPU, memory, storages, etc.).

According to the memory utility rate, CPU loading, etc., of collaborative nodes, loading analysis component (LAC) applies the fuzzy inference to analyze workloads. The LAC can evaluate performance grade which is a rating of collaborative node load grade by their CPU usage and memory usage. We use assessment CPU and memory criteria which were divided into five levels by fuzzy inferences.

The criteria ratings of CPU usage are linguistic variables with linguistic values C1, C2, C3, C4, C5, where C1 = very light, C2 = light, C3 = middle, C4 = heavy, C5 = very heavy. These linguistic values are treated as fuzzy numbers with trapezoid membership functions as shown in Figure 6.

The criteria ratings of Memory usage are linguistic variables with linguistic values M1, M2, M3, M4, M5, where M1 = very light, M2 = light, M3 = middle, M4 = heavy, M5 = very heavy. These linguistic values are treated as fuzzy numbers with trapezoid membership functions as shown in Figure 7.

The criteria ratings of computing performance are linguistic variables with linguistic values P1, P2, P3, P4, P5, where P1 = very low, P2 = low, P3 = middle, P4 = high, P5 = very high. These linguistic values are treated as fuzzy numbers with triangular membership functions as shown in Figure 8.

We set zero as an initial value of computing performance of the collaborated nodes. If there has a new job, the value of computing performance will be re-evaluated. There are twenty-five rules in our fuzzy inference rule base, as shown in Table 3.

Figure 9 shows that our rule base is reasonable and robust. We may evaluate computing performance of all grid nodes by the rules.

We can plan and allocate the priority of demanders depending on computing performance. The higher performance grade means the collaborative node deals with more workload now. The lower performance grade means the collaborative node deals with lower workload. Therefore, the lower performance grade can process more workload. Those collaborative nodes of the lower performance grade will send to RAM.
2.3. **Resource allocation module (RALM).** The Resource allocation module (RALM) can optimize the planning and allocating of the collaborative grid nodes resources. It comprises two components: job arranging component (JAC) and allocation mode component (PMC), as shown in Figure 10.

The job arranging component (JAC) receives the collaborative node evaluation information from DCC, and sends it to allocation mode component (PMC) for the allocation of planning.

### Table 3. Fuzzy inference rule base

<table>
<thead>
<tr>
<th>Memory</th>
<th>CPU</th>
<th>very light</th>
<th>light</th>
<th>middle</th>
<th>heavy</th>
<th>very heavy</th>
</tr>
</thead>
<tbody>
<tr>
<td>very light</td>
<td>very high</td>
<td>very high</td>
<td>high</td>
<td>middle</td>
<td>middle</td>
<td>middle</td>
</tr>
<tr>
<td>light</td>
<td>very high</td>
<td>very high</td>
<td>high</td>
<td>middle</td>
<td>middle</td>
<td>low</td>
</tr>
<tr>
<td>middle</td>
<td>high</td>
<td>high</td>
<td>middle</td>
<td>low</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>heavy</td>
<td>middle</td>
<td>middle</td>
<td>low</td>
<td>low</td>
<td>very low</td>
<td>very low</td>
</tr>
<tr>
<td>very heavy</td>
<td>low</td>
<td>low</td>
<td>very low</td>
<td>very low</td>
<td>very low</td>
<td>very low</td>
</tr>
</tbody>
</table>
The allocation mode component (PMC) will allocate the job to proper nodes according to the user demands. There are three allocation methods which we can choose, saying, average allocation, optimization cost allocation and optimization computing performance allocation.

- The average allocation is that the jobs of workload for each grid node are almost equal.
- The optimization cost allocation is based on the grid resources cost. The one grid node of lowest cost is precedence allocation.
- The optimization computing performance allocation is based on the grid computing performance. The higher the grid computing performance is the more workload.

3. Model Implementation. In order to present the efficacy of this model, we implement this model in this section.

3.1. Implementation environment. We gave an overview of the configuration of the software and hardware used in the implementation. Referred to [4], we built a very small scenario. It is the simplest grid environment. A D-Link 10/100 fast Ethernet Switch Hub linked the three machines with the Windows XP Professional. Table 4 outlines the names of the machines to be used in the grid, their IP addresses and their functionality. We detected the status of the host M0 and nodes N1 and N2 in this environment as shown in Figure 11 and Figure 12, respectively. If we have more than three machines available,
we can build a bigger scenario for Proof-of-Concepts (PoC) proposals and/or demos. For that, we simply include more servers, such as N₃, N₄, and so on [4].

**Table 4. Collaborative grid nodes name and their IP addressing**

<table>
<thead>
<tr>
<th>Collaborative grid node name</th>
<th>IP address</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₀: CHUNG-PC</td>
<td>192.168.0.11</td>
<td>Monitoring and collaborative computing</td>
</tr>
<tr>
<td>N₁: ACERXP</td>
<td>192.168.0.22</td>
<td>Collaborative computing</td>
</tr>
<tr>
<td>N₂: PMXP</td>
<td>192.168.0.33</td>
<td>Collaborative computing</td>
</tr>
</tbody>
</table>

**Figure 11.** The status of the host node M₀

For easy to manipulate, we applied Visual Studio .NET 2005 as the development tool, and Matlab Fuzzy Toolbox as the fuzzy inferences for the CPU loading and memory utility rate of collaborative grid nodes to do the fuzzy inferences.

### 3.2. Practical implementation

There are three allocation methods which we can choose, saying, average allocation, optimization cost allocation and optimization computing performance allocation. The implementing interface is as shown in Figure 13.

By this model, after detecting resources of collaborative node on grid environment, and evaluate related resource information of collaborative node and then can choose three allocation methods which are average allocation, optimize cost allocation and optimize computing performance allocation to make the plans and allocations of the resources of collaborative nodes optimize. Under the same type of work, these three allocation methods will come up with three computed results, as shown in Table 5.

According to the results of the experiment, we have that if we spend lower cost, the implementation of the work will take longer period of time. If we want to have a good computing performance, the cost will be higher.
Figure 12. The status of the collaborative nodes, $N_1$ and $N_2$

Figure 13. The implementing interface of the proposed model

Table 5. The computed results by the three allocation methods

<table>
<thead>
<tr>
<th>Allocation</th>
<th>Executed time (sec)</th>
<th>Executed cost (G-coin)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>29.392</td>
<td>1550</td>
</tr>
<tr>
<td>Optimization cost</td>
<td>33.668</td>
<td>550</td>
</tr>
<tr>
<td>Optimization computing performance</td>
<td>21.949</td>
<td>1750</td>
</tr>
</tbody>
</table>
4. **Conclusion.** At present, in grid computing environment, job requirements are so large scale and complex that we need the allocating mechanism to manage the resources and schedule the jobs. Therefore, well-allocated mechanism is needed to enhance the grid resource be more useful and scalable.

In this paper, we propose a performance of resource allocation model for grid nodes under the grid computing environment. By this model, we can evaluate the information of collaborative nodes and make the plans and allocations of the resources of collaborative nodes optimize. According to the result of experiment, the mechanism can achieve the best resources allocation, and enhance the overall grid computing performance.

**Acknowledgment.** This work was partially supported by the National Science Council, Taiwan, under Grant NSC-94-2745-M-034-008-URD, NSC-95-2745-M-034-007-URD and NSC-96-2745-M-034-002-URD. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

**REFERENCES**


