CHAPTER 6

DESIGN OF HIERARCHICAL LOAD BALANCING MODEL WITH USER DEMAND AWARE SCHEDULING

This chapter describes a hierarchical load balancing approach which considers load of each resource and performs load balancing. It considers dynamicity of the resources i.e. the availability time of the resources. Since load is shared effectively among all the resources in the grid, it minimizes the response time of jobs and improves the utilization of resources in grid environment.

6.1 INTRODUCTION

In grid environment, workload management and resource management are two essential functions provided at the service level of grid software infrastructure. To improve the global throughput, workloads have to be evenly distributed among the available resources. Load balancing attempts to evenly distribute the computation load across multiple computational resources. An effective load balancing mechanism should improve the performance at application level and system level.

To minimize the total time needed to complete all the tasks, the workload has to be evenly distributed over all the resources based on their computational power. The load balancing problem is closely related to scheduling and resource allocation. It is mainly concerned with the techniques which perform even distribution of workload among available resources in a
system. The main objective of a load balancing is to optimize the average response time of tasks.

Load balancing scheme maintains the workload proportionally equivalent on the whole system resources. Load balancing involves four basic steps. They are monitoring resource state and load, exchanging load and state information between resources, calculating the new load information and updating data movement.

Generally grid computing consists of two major parties namely resource consumers (users) who submit the applications and the resources providers who share their resources. While grid users mainly concentrate on the performance of their tasks, such as total cost and time to execute a particular task, resource providers mainly concentrate on the performance of their resources such as the resource utilization in a particular period. Thus, the objective functions of scheduling can be classified into two categories such as application-centric and resource-centric. Load balancing algorithms which adopt an application centric objective function aim to optimize the performance of each application. Most of the grid applications are concerned about time which is the makespan. Load balancing algorithms that adopt resource centric objective functions aim to improve the performance of the resources. Resource centric objectives are mainly related to resource utilization.

An efficient task scheduling approach is required for allocating tasks to resources efficiently. Since grid resources are dynamic, an effective load balancing algorithm is required to maintain load at resources. To improve user satisfaction, the deadline of the jobs has to be considered. The HLBA algorithm considers both user satisfaction and load balancing.
6.2 SCHEDULING MODEL

Figure 6.1 depicts the proposed HLBA scheduling architecture.

![HLBA Scheduling Architecture Diagram]

Figure 6.1 HLBA scheduling architecture

Users submit jobs in the form of gridlets to the machines. A gridlet is an entity which contains information such as length, data to be transferred for execution and the information of the user who submits the job. Based on these information, expected execution time and data transfer time between the user and the resource are calculated which are used to select appropriate resources for the jobs.

Grid resource broker collects information about the resources such as capacity (MIPS), current load of the resource and baud rate of the resource from grid information service. Grid resource is an entity which is next to the grid resource broker in the hierarchy. It is responsible for maintaining load at the machines which are next to the resource in the hierarchy. Machines are
responsible for scheduling and maintaining balanced load at the processing elements.

6.3 HLBA ALGORITHM

The proposed scheduling algorithm mainly aims at effective resource utilization and minimized makespan. Hence it considers both application and system aspects. It considers user deadline, expected completion time, data transfer time and load of each resource. It calculates the load at different levels. Since machine and Processing Elements (PEs) are arranged very close to the resource, data transfer time to machine/PE is negligible and the resources are geographically distributed, data transfer time from resource broker to the resource is considered.

Steps:

1. Users submit the jobs to the machines and the jobs are added to the unassigned list of the machine.

2. Assign the jobs among the PEs of the machine with satisfied user demand using Algorithm 6.1 and remove the job from unassigned list. If it is not possible, then forward the set of unassigned jobs to the resource.

3. Assign the jobs among the machines of the resource with satisfied user demand using Algorithm 6.1 and remove the job from unassigned list. If it is not possible, then forward the set of unassigned jobs to the resource broker.

4. Assign the jobs among the resources with satisfied user demand using Algorithm 6.1 and remove the job from unassigned list.
User submits jobs at the machines and then the machines schedule jobs to their PEs which are capable of completing the jobs within user deadline. If there is no suitable PE, then the jobs will be forwarded to resources. The resources will then schedule the jobs to the machines. If there is no suitable PEs found at all machines under that resource, then the resources will forward the jobs to the resource broker which will schedule the jobs to other resources. The symbols that are used in this algorithm are listed in Table 6.1.

**Table 6.1 List of Symbols used in HLBA Algorithm**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PELoad_j</td>
<td>Load of the PE j</td>
</tr>
<tr>
<td>MLoad_j</td>
<td>Load of the machine j</td>
</tr>
<tr>
<td>RLoad_j</td>
<td>Load of the resource j</td>
</tr>
<tr>
<td>r.state</td>
<td>State of the resource</td>
</tr>
<tr>
<td>m.state</td>
<td>State of the machine</td>
</tr>
<tr>
<td>pe.state</td>
<td>State of the PE</td>
</tr>
</tbody>
</table>

**Load Balancing:** To ensure effective utilization of the resources/ machines/ PEs, load is calculated at different levels. Load of a PE is calculated based on the jobs that are already assigned to it, MIPS rate of the PE and availability time of that particular processing element.

\[
\text{PELoad}_j = \frac{\sum_{i=1}^{n} \text{Length}_i}{\text{MIPS}_j \times \text{AT}_j}
\]  

(6.1)

where \( n \) is the number of jobs allocated to PE\(_j\). Load of a machine is the average load of the PEs under the machine which is calculated as follows.

\[
\text{MLoad}_j = \frac{\sum_{j=1}^{n} \text{PELoad}_j}{n}
\]  

(6.2)
where \( n \) is the number of PE’s under machine \( j \). Resource is a collection of machines and the average load of each resource is calculated as follows.

\[
R_{\text{Load}}_j = \frac{\sum_{i=1}^{n} M_{\text{Load}}_i}{n} \quad (6.3)
\]

Since grid is a collection of dynamic resources from different administrative domains, the resources/machines/PEs can be online and offline from the grid. So the status (either online or offline) of the resource/machine/PE must be updated periodically. To perform load balancing, the state of the resource/machine/PE must be checked when a job arrives or status of the resource/machine/PE is changed. The state of a resource is classified into three types such as heavily loaded, normally loaded and under loaded. To assign the state of resource/machine/PE, the threshold value is calculated at different levels. Based on the threshold value at each level, a resource or a machine or a processing element is classified as under loaded, heavily loaded and normally loaded.

Threshold value for resource (\( T_{\text{Res}} \)) is calculated as follows

\[
T_{\text{Res}} = \frac{\sum_{i=1}^{n} R_{\text{Load}}_i}{n} \quad (6.4)
\]

where \( n \) - Number of resources under resource broker

Threshold value for machine (\( T_{\text{Mac}} \)) is calculated as follows

\[
T_{\text{Mac}} = \frac{\sum_{j=1}^{m} M_{\text{Load}}_j}{m} \quad (6.5)
\]

where \( m \) - Total number of machines under a resource

Threshold value for PE (\( T_{\text{PE}} \)) is calculated using the formula given below
where \( l \) - Number of PEs under a machine.

Algorithm 6.1 HLBA scheduling algorithm

Algorithm 6.1 depicts the proposed HLBA scheduling algorithm for assigning jobs to suitable resources in the grid system. Initially, jobs are collected from the user and added in the unassigned list. Then a job which has minimum user deadline is selected from the unassigned list. Then for each resource in the grid system, expected execution time of the selected job, data transfer time of the job, ready time of the resource and expected completion time of the job are calculated. Then the difference factor is calculated by considering the user deadline and the resources/machines/processing elements.
are classified as over loaded, under loaded and normally loaded. Then the resource with minimum difference factor is selected from the under loaded list and the selected job is assigned to it. Finally job from the unassigned list is removed and the load of the resources is updated.

In this methodology, threshold value of the load factor is calculated for the resource level based on load factor at each resource in the grid system. Then the resources are categorized based on the threshold value and actual load level at each resource by using Algorithm 6.2. For example, if the load factor of a resource is less than the threshold value, then the resource state is assigned as under loaded and added to the under loaded list of resources. If the load factor of a resource is greater than the threshold value, then the resource state is assigned as heavily loaded and added to the heavily loaded list of resources. Otherwise the resource state is assigned as normally loaded and added to the normally loaded list of resources.

**Algorithm 6.2 State allocation algorithm for resource**

```
Begin
    For all resources
        if (RLoad < T_{Res})
            r.state = “underloaded”
            Add the resource to r.underloaded list
        else if (RLoad > T_{Res})
            r.state = “heavilyloaded”
            Add the resource to r.heavilyloaded list
        else
            r.state = “normallyloaded”
            Add the resource to r.normallyloaded list
    End for
End
```
Similar to resource level, threshold value of load factor is also calculated at machine level based on load factor at each machine under a resource. Then the machines are categorized based on the threshold value and actual load level at each machine by using Algorithm 6.3. For example, if load factor of a machine is less than the threshold value, then the machine state is assigned as under loaded and added to the under loaded list of machines. If load factor of a machine is greater than the threshold value, then the machine state is assigned as heavily loaded and added to the heavily loaded list of machines. Otherwise the machine state is assigned as normally loaded and added to the normally loaded list of machines.

```
Begin
  For all resources
    For all machines
      if (MLoad < TMac)
        m.state = “underloaded”
        Add the machine to m.underloaded list
      else if MLoad > TMac
        m.state = “heavilyloaded”
        Add the machine to m.heavilyloaded list
      else
        m.state = “normallyloaded”
        Add the machine to m.normallyloaded list
    End for
  End for
End
```

Algorithm 6.3 State allocation algorithm for machine
Finally, the threshold value of load factor is calculated at PE level based on load factor at each PE under a machine. Then the PEs are categorized based on the threshold value and actual load level at each PE by using Algorithm 6.4. For example, if load factor of a PE is less than the threshold value, then the PE state is assigned as under loaded and added to the under loaded list of PEs. If load factor of a PE is greater than the threshold value, then the PE state is assigned as heavily loaded and added to the heavily loaded list of PEs. Otherwise the PE state is assigned as normally loaded and added to the normally loaded list of PEs.

```
Begin
    For all resources
        For all machines
            For all PEs
                if (PELoad < T_{PE})
                    pe.state = “underloaded”
                    Add it to underloaded list
                else if (PELoad > T_{PE})
                    pe.state = “heavilyloaded”
                    Add it heavilyloaded list.
                else
                    pe.state = “normallyloaded”
                    Add it normallyloaded list.
                End if
            End for
        End for
    End for
End
```

**Algorithm 6.4 State allocation algorithm for processing element**
Algorithm 6.5 Status update algorithm

When a resource broker/resource/machine gets information about dynamicity (status) of the resource/machine/PE i.e. when a resource is newly added, then the status is updated using Algorithm 6.5.

6.4 SIMULATION RESULTS AND PERFORMANCE ANALYSIS

The proposed HLBA algorithm is simulated by using GridSim toolkit for varying number of jobs from 100 to 600 with 16 resources. The parameters of jobs and resources in grid hierarchy are given in Table 6.2.

<table>
<thead>
<tr>
<th>Characteristics of grid resources and jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of resources</td>
</tr>
<tr>
<td>Number of machines</td>
</tr>
<tr>
<td>Number of PEs per machine</td>
</tr>
<tr>
<td>Number of jobs</td>
</tr>
<tr>
<td>MIPS Rate of PEs</td>
</tr>
<tr>
<td>Number of instructions of a job (MI)</td>
</tr>
</tbody>
</table>

The simulation is carried out in two ways. First, the number of resources is kept at 16, the number of jobs is varied from 100 to 600 and the performance measures such as makespan, hit rate and resource utilization are measured. Second, the number of jobs is considered as 512 and the number of

```plaintext
Begin
    if (Resource/Machine/PE is newly arrived in the grid system)
        Calculate new threshold value at different levels
        Classify the resource/machine/PE based on Algorithms 6.2, 6.3 and 6.4
    End if
End
```
resources is considered as 16 which is the benchmark for scheduling algorithms to evaluate their efficiency.

Bardsiri & Rafsanjani (2012) proposed a New Heuristic approach based on Load Balancing (NHLB) mechanism which has two phases. In the first phase, heuristic begins with the set of all unmapped tasks and minimum completion time is calculated for all the jobs in all the machines like Min-min heuristic algorithm. In the second phase, jobs are allocated to the machines by considering workload of the machines with two choices based on threshold value. In the first choice, for each task the minimum, second minimum completion time and minimum execution time are found. Then criteria value (k) is calculated based on minimum completion time values and minimum execution time. In the second choice, that the number of the remaining tasks is less than threshold, heuristic algorithm criteria for allocation is applied and the machine is selected which has workload of it is less than average. Finally, the task which has maximum criteria value (k) is selected and removed from set of unmapped tasks.

The result of the proposed load balancing algorithm is analyzed and compared with NHLB algorithm. Factors such as resource/machine heterogeneity and job heterogeneity are considered. The improvement of the proposed algorithm is proved by comparing makespan, hit rate and resource utilization. Utilization of the resources is calculated using the following formula.

$$RU_j = \frac{R_{\text{load}}_j}{100}$$  \hspace{1cm} (6.7)

where \( RU_j \) is utilization of the resource \( j \)

Average Resource Utilization (ARU) is calculated as follows

$$\text{ARU} = \frac{\text{mean}(RU_j)}{\forall \ j = 1 \ to \ m}$$  \hspace{1cm} (6.8)

where \( m \) is the total number of resources.
Table 6.3 shows the makespan values of NHLB and HLBA algorithms. The proposed HLBA algorithm has less makespan values than the other two algorithms for varied number of jobs from 100 to 600. By considering load and dynamicity of the resources, the makespan values are reduced.

**Table 6.3 Performance of HLBA algorithm based on makespan for varied number of jobs**

<table>
<thead>
<tr>
<th>Number of Jobs</th>
<th>Makespan (Seconds)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NHLB</td>
<td>HLBA</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>321</td>
<td>265</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>946</td>
<td>732</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>1054</td>
<td>967</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>1456</td>
<td>1324</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>1854</td>
<td>1672</td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>2487</td>
<td>2259</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6.2 Makespan of HLBA algorithm for varied number of jobs**
In Figure 6.2, HLBA algorithm is compared with NHLB algorithm in terms of its efficiency of scheduling which can be measured by makespan for varying number of jobs from 100 to 600. The performance of the proposed algorithm is better when compared to NHLB algorithm.

Table 6.4  Performance of HLBA algorithm based on hit rate for varied number of jobs

<table>
<thead>
<tr>
<th>Number of Jobs</th>
<th>Hit Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NLB</td>
</tr>
<tr>
<td>100</td>
<td>79</td>
</tr>
<tr>
<td>200</td>
<td>73</td>
</tr>
<tr>
<td>300</td>
<td>67</td>
</tr>
<tr>
<td>400</td>
<td>63</td>
</tr>
<tr>
<td>500</td>
<td>57</td>
</tr>
<tr>
<td>600</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 6.4 shows the percentage of hit rate for different number of jobs for various algorithms such as NHLB and HLBA algorithms.

Figure 6.3  Hit rate of HLBA algorithm for varied number of jobs
In Figure 6.3, the efficiency of the proposed HLBA algorithm based on hit rate is analyzed and the analysis shows that the HLBA algorithm has highest hit rate compared to the other two algorithms.

**Table 6.5** Performance of HLBA algorithm based on average resource utilization for varied number of jobs

<table>
<thead>
<tr>
<th>Number of Jobs</th>
<th>Average Resource Utilization (%)</th>
<th>NHLB</th>
<th>HLBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>89</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>92</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>90</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>91</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>88</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>91</td>
<td>95</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.5 shows the average resource utilization for a number of jobs between 100 and 600. The average resource utilization values are calculated for various algorithms such as NHLB and HLBA algorithms. The proposed HLBA algorithm has better average resource utilization compared to the other algorithms.

**Figure 6.4** Average resource utilization of HLBA algorithm for varied number of jobs
Figure 6.4 shows the performance analysis based on average resource utilization. The proposed HLBA algorithm has better resource utilization when compared with other algorithms.

Table 6.6 Performance of HLBA algorithm based on makespan for different cases

<table>
<thead>
<tr>
<th>Cases</th>
<th>Makespan (Seconds)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NHLB</td>
<td>HLBA</td>
</tr>
<tr>
<td>Case 1</td>
<td>1734</td>
<td>1720</td>
</tr>
<tr>
<td>Case 2</td>
<td>1428</td>
<td>1397</td>
</tr>
<tr>
<td>Case 3</td>
<td>1690</td>
<td>1675</td>
</tr>
<tr>
<td>Case 4</td>
<td>1703</td>
<td>1687</td>
</tr>
</tbody>
</table>

Table 6.6 shows the makespan values for different cases with 512 jobs and 16 resources. The jobs are heterogeneous because they vary in terms of length of the job, data to be transferred for execution of the job and user deadline. The resources that are considered for the simulation are also heterogeneous since these resources are varied based on capacity and baud rate. The proposed HLBA algorithm has the minimal makespan when compared with other algorithms.

Figure 6.5 Makespan of HLBA algorithm for different cases
Figure 6.5 shows the performance analysis based on makespan for different cases with 512 jobs and 16 resources. The proposed HLBA algorithm has minimum makespan when compared with NHLB algorithm.

Table 6.7 Performance of HLBA algorithm based on hit rate for different cases

<table>
<thead>
<tr>
<th>Cases</th>
<th>Hit Rate (%)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NHLB</td>
<td></td>
<td>HLBA</td>
</tr>
<tr>
<td>Case 1</td>
<td>68</td>
<td></td>
<td>70</td>
</tr>
<tr>
<td>Case 2</td>
<td>77</td>
<td></td>
<td>80</td>
</tr>
<tr>
<td>Case 3</td>
<td>67</td>
<td></td>
<td>69</td>
</tr>
<tr>
<td>Case 4</td>
<td>68</td>
<td></td>
<td>73</td>
</tr>
</tbody>
</table>

Table 6.7 shows the percentage of hit rate for different sets of 512 jobs and 16 resources for various algorithms such as NHLB and HLBA algorithms. The proposed HLBA algorithm produces more hit rate than NHLB algorithm.

Figure 6.6 Hit rate of HLBA algorithm for different cases
Figure 6.6 shows the performance analysis based on hit rate for different cases with 512 jobs and 16 resources. The proposed HLBA algorithm has more hit rate when compared with the other algorithms.

Table 6.8  Performance of HLBA algorithm based on average resource utilization for different cases

<table>
<thead>
<tr>
<th>Cases</th>
<th>Average Resource Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NHLB</td>
</tr>
<tr>
<td>Case 1</td>
<td>90</td>
</tr>
<tr>
<td>Case 2</td>
<td>94</td>
</tr>
<tr>
<td>Case 3</td>
<td>90</td>
</tr>
<tr>
<td>Case 4</td>
<td>95</td>
</tr>
</tbody>
</table>

Table 6.8 shows the percentage of average resource utilization for different sets of 512 jobs and 16 resources for various algorithms such as NHLB and HLBA algorithms. The proposed HLBA algorithm produces more average resource utilization than the other algorithms.

Figure 6.7  Average resource utilization of HLBA algorithm for different cases
Figure 6.7 shows the performance analysis based on average resource utilization for different cases with 512 jobs and 16 resources. The proposed HLBA algorithm has higher average resource utilization when compared with the other algorithms.

6.5 CONCLUSION

In this work, hierarchical load balancing approach with user demand aware scheduling algorithm is proposed. It considers the dynamicity of the resources/machines/PEs as well as machine heterogeneity and task heterogeneity. Since this algorithm considers the load of resource/machine/PE, their utilization is improved. By considering the user deadline of the job, the number of jobs completed within user deadline (hit rate) is increased and hence user satisfaction is improved. When compared with other scheduling algorithms, the overall system performance is improved in the proposed scheduling algorithm.

The next chapter describes a Fault Tolerant Load Balancing scheduling algorithm which considers failure rate of the resources to predict job failures before scheduling.