Chapter - 5
CHAPTER 5

IMPLEMENTATION AND EXPERIMENTAL DETAILS

Throughout the implementation of the Priority Based Multiple Queue Algorithm emphasis has been on the use of Open Source Technologies and Toolkits. The Priority Based Multiple Queue Algorithm has been implemented in JAVA programming language on top of GridSim [24] Toolkit 4.0. The steps followed for the implementation are described below in detail with appropriate screenshots.

5.1 Installation of Pre-requisites and Necessary Components

5.1.1 Installation of GridSim Toolkit

In the current implementation of the Priority Based Multiple Queue algorithm, GridSim Toolkit version 4.0 has been used. This open source JAVA based toolkit can be downloaded from the homepage of Grid-Bus Society [16].

The GridSim [24]toolkit provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers. It can be used to simulate application schedulers for single or multiple administrative domains distributed computing systems such as clusters and Grids.

5.1.2 Salient Features of GridSim [24]

- It allows modeling of heterogeneous types of resources.
- Resources can be modeled operating under space-or time-shared mode.
Resource capability can be specified in the form of MIPS (Million Instructions per Second) or as per SPEC (Standard Performance Evaluation Corporation).

Resources can be located in any time zone.

Resources can be booked for advance reservation.

Applications with different parallel application models can be simulated.

Application tasks can be heterogeneous and they can be CPU or I/O intensive.

No limit on the number of application jobs that can be submitted to a resource.

Multiple user entities can submit tasks for execution simultaneously in the same resource, which may be time-shared or space-shared. Network speed between resources can be specified.

It supports simulation of both static and dynamic schedulers.

Statistics of all or selected operations can be recorded and it can be analyzed using GridSim statistics analysis methods.

5.1.3 GridSim Architecture

GridSim has employed a layered and modular architecture for Grid simulation to leverage existing technologies and manage them as separate components. A multilayer architecture and abstraction for the development of GridSim platform and its applications is shown in Figure 5.1.
Figure 5.1 Architecture for GridSim platform and components

The first layer is concerned with the scalable Java interface and the runtime machinery, called JVM (Java Virtual Machine). The second layer is concerned with a basic discrete event infrastructure built using the interfaces provided by the first layer. One of the popular discrete-event infrastructure implementations available in Java is SimJava [25]. The third layer is concerned with modeling and simulation of core Grid entities such as resources, information services, and so on; application model, uniform access interface, and primitives application modeling and framework for creating higher level entities. The GridSim toolkit focuses on this layer that
Chapter 5

The final layer is focused on application and resource modeling with different scenarios using the services provided by the two lower-level layers for evaluating scheduling and resource management policies, heuristics, and algorithms.

Figure 5.2 A Flow Diagram for GridSim based Simulations [22].
5.2 Implementation of Priority Based Multiple Queue Scheduling Algorithm in GridSim

To use the GridSim for the implementation and simulation of the Priority Based Multiple Queue algorithm, it needs to follow the steps mentioned in Appendix-A; which when together coded in JAVA programming language simulate a heterogeneous Grid Computing Environment. All the type names referenced here are part of GridSim API.

The complete Priority Based Multiple Queue algorithm is implemented in JAVA and run on top of GridSim version 4.0 Toolkit. This chapter explains and discusses the results obtained in the statistical light.

In the starting window the user is prompted to enter the values for the number of Jobs needed to be simulated.

![Figure 5.3 User Input Environment](image)

Figure 5.3 User Input Environment

Due to the GridSim, jobs will be prepared in simulated environment which shows in following figure. After that jobs will assign to the respective queue.
The GridSim toolkit supports modeling and simulation of a wide range of heterogeneous resources, such as single or multiprocessors, shared and distributed memory machines such as PCs, workstations, SMPs, and clusters with different capabilities and configurations. It can be used for modeling and simulation of application scheduling on various classes of parallel and distributed computing systems such as clusters [26], Grids, and P2P networks [27]. The resources in clusters are located in a single administrative domain and managed by a single entity, whereas in Grid and P2P systems, resources are geographically distributed across multiple administrative domains with their own management policies and goals. Another key difference between cluster and Grid/P2P systems arises from the way application scheduling is performed. The GridSim toolkit provides facilities for the modeling and simulation of resources and network connectivity with different capabilities, configurations, and domains. It supports primitives for application composition, information services for resource discovery, and interfaces for assigning application tasks to resources and managing their execution. These features can be used to simulate resource brokers or Grid schedulers for evaluating performance of scheduling algorithms or heuristics.
GridSim creates the jobs and assigns them to the respective priority queue. There are three queues with three different scheduling algorithms. There are six combinations of the different queues. Same input (job) to every combination of queues and determine which one is the best combination for execution of jobs. Figure-5.2 is self-explanatory, that what the role of the Priority based multiple queue is. It will take the job from the user in the simulated environment. Optimize the sequence of the job, and submitted to Meta Grid scheduler, which is responsible for the job execution and job completions. In this way priority based multiple queue can increase the performance of the Meta scheduler.

For example, consider the scenario of the jobs in which jobs come after regular intervals having burst time and priority. If user does not specify any priority then that job will be assign to the last queue of the multiple queue. The job scenario table below:

Table 5.1: Jobs Scenario Table for the Priority Based Multiple Queues

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Arrival Time (millisecond)</th>
<th>Burst Time (millisecond)</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job1</td>
<td>0</td>
<td>8</td>
<td>62</td>
</tr>
<tr>
<td>Job2</td>
<td>1</td>
<td>0</td>
<td>55</td>
</tr>
<tr>
<td>Job3</td>
<td>2</td>
<td>3</td>
<td>93</td>
</tr>
<tr>
<td>Job4</td>
<td>3</td>
<td>3</td>
<td>31</td>
</tr>
<tr>
<td>Job5</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Job6</td>
<td>5</td>
<td>7</td>
<td>49</td>
</tr>
<tr>
<td>Job7</td>
<td>6</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>Job8</td>
<td>7</td>
<td>8</td>
<td>56</td>
</tr>
<tr>
<td>Job9</td>
<td>8</td>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>Job10</td>
<td>9</td>
<td>1</td>
<td>54</td>
</tr>
</tbody>
</table>
In the priority based multiple queues, it has three queues, each having its own algorithm for job arrangement in their respective queue. So there are six different combinations of these three queues and try to identify which combination is best for the implementation. First combination (Figure 5.5) is First Come First Serve in first queue, Shortest Job First in second queue and Round Robin in the last queue (FCFS -> SJF -> RR). At last jobs will arrange in the sequence according to the first combination. Calculate the average waiting time, response time, and turnaround time for first combination. That jobs sequence will send to the Grid Meta scheduler which is in optimize manner, which shows in following figure.

![Figure 5.5 Final Queue Preparations in First Combination](image-url)

Figure 5.5 Final Queue Preparations in First Combination
Figure 5.6 Waiting for Combination Selection

Second combination (Figure 5.7) is First Come First Serve in first queue, Round Robin in Second queue and Shortest Job First in the last queue (FCFS -> RR -> SJF). At last jobs will arrange in the sequence according to the second combination. Calculate the average waiting time, response time, and turnaround time for second combination. That jobs sequence will send to the Grid Meta scheduler which is in optimize manner, which shows in following figure.
Figure 5.7 Final Queue Preparations in Second Combination

![Priority Based Multiple Queue](image)

Figure 5.8 Waiting for Combination Selection

Similarly in third combination (Figure 5.9) is Shortest Job First in first queue, First Come First Serve in second queue and Round Robin in the last queue (SJF -> FCFS -> RR). At last jobs will arrange in the sequence according to the third combination. Calculate the average waiting time, response time, and turnaround time for second combination. In forth combination (Figure 5.10) is Shortest Job First in first queue, Round Robin in second queue and First Come First Serve in the last queue (SJF -> RR -> FCFS). At last jobs will arrange in the sequence according to the forth combination. Calculate the average waiting time, response time, and turnaround time for second combination. In fifth combination (Figure 5.11) is Round Robin in first queue, First Come First Serve in second queue and Shortest Job First in the last queue (RR -> FCFS -> SJF).

At last jobs will arrange in the sequence according to the fifth combination. Calculate the average waiting time, response time, and turnaround time for second combination. In sixth combination (Figure 5.12) is Round Robin in first queue, Shortest Job First in second queue and First
Come First Serve in the last queue (RR -> SJF -> FCFS). At last jobs will arrange in the sequence according to the sixth combination. Calculate the average waiting time, response time, and turnaround time for second combination. That jobs sequence will send to the Grid Meta scheduler which is in optimize manner, which shows in following figures.

![Figure 5.9 Final Queue Preparations in Third Combination](image)

![Figure 5.10 Final Queue Preparations in Fourth Combination](image)

![Figure 5.11 Final Queue Preparations in Fifth Combination](image)

![Figure 5.12 Final Queue Preparations in Sixth Combination](image)
Figure 5.13 All Combination Prepared

Finally the entire job’s sequence is shown in the figure-5.13 at each combination. The best combination is in this scenario is Combination 2 (FCFS -> RR -> SJF) for waiting time and response time, and for the turnaround time Combination 1(FCFS -> SJF -> RR) is the best combination for this scenario. The summery of this case study is shown in following table.

Table 5.2: Comparison of different combination of algorithms based on performance metric

<table>
<thead>
<tr>
<th>Combination</th>
<th>Average waiting Time (millisecond)</th>
<th>Average turnaround time(millisecond)</th>
<th>Average Response time (millisecond)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFS-&gt;SJF-&gt;RR</td>
<td>15.6</td>
<td>38.2</td>
<td>14.3</td>
</tr>
<tr>
<td>FCFS-&gt;RR-&gt;SJF</td>
<td><strong>15.1</strong></td>
<td>38.4</td>
<td><strong>14.1</strong></td>
</tr>
<tr>
<td>SJF-&gt;FCFS-&gt;RR</td>
<td>37.7</td>
<td><strong>25.8</strong></td>
<td>16.3</td>
</tr>
<tr>
<td>SJF-&gt;RR-&gt;FCFS</td>
<td>37.8</td>
<td>25.9</td>
<td>16.5</td>
</tr>
<tr>
<td>RR-&gt;FCFS-&gt;SJF</td>
<td>25.4</td>
<td>29.5</td>
<td>17.9</td>
</tr>
<tr>
<td>RR-&gt;SJF-&gt;FCFS</td>
<td>26.4</td>
<td>30.1</td>
<td>37.8</td>
</tr>
</tbody>
</table>
The selection of the combination is depends upon the scenario of the jobs. It means that, anyone of the combination can give the optimize result which depends upon the scenario of the jobs. The result may vary scenario to scenario. Finally best job sequence is submitted to the middleware for the purpose of computation of jobs.