CHAPTER 7
CHAPTER 7
IMPROVED PRIORITY BASED MULTI QUEUE JOB
SCHEDULING ALGORITHM

7.1 INTRODUCTION

Multi Queue Scheduling research work has already discussed in the Chapter 3. In MQS the client submitted jobs are processed on the basis of execution time, the first 40% of the jobs are stored in the small queue, the next 40% of the jobs are stored in the medium queue and remaining 20% jobs are stored in the long queue. It classifies the jobs based on their Burst Time (BT) of among the total submitted. In Improved Priority based Multi Queue Job Scheduling algorithm schedule the jobs based on their BT. It classifies the jobs into high, medium and small priority. The priority of jobs decides whether the created system is a quality one or not. The user submitted jobs are entered into the resource poll. The resource pool consists of collected works done by user submitted jobs that form a sequence and updates the queue manager. The queue manager is responsible for handling the queue and allocation of job to the resources. Queue manager allocates the job id for each job. The scheduler divides the jobs in Table 7.1 shows the classification of jobs in to 3 types such as follows:

<table>
<thead>
<tr>
<th>Type of Priority</th>
<th>Complexity Type</th>
<th>Queue Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>High</td>
<td>QA</td>
</tr>
<tr>
<td>Medium</td>
<td>Medium</td>
<td>QB</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
<td>QC</td>
</tr>
</tbody>
</table>

Table 7.1 Classification of Jobs

First the high BT is high priority jobs that need high processor speed and highly complex one are grouped in to QA. Second the medium BT medium priority
jobs that need medium processor speed and medium complex one are grouped into QB. Finally the low BT the low priority jobs that need low processor speed and low complex one are grouped in to QC.

Figure 7.1 Architecture of IPMQJS model

The above Figure 7.1 shows the Architecture of IPMQJS model in Cloud Environment. The job scheduler updates the total number of grouped jobs on the basis of speed of the processor needed and their execution time. The processed job is finally entered into the cloud environment.

This proposed IPMQJS algorithm gives more importance to the allocation of the high priority queue with high speed processor with high complex jobs are executed. Then it takes the medium priority queue with medium speed processor with medium complex jobs are executed and finally low priority queue with low speed processor with low complex jobs are executed among the total submitted jobs.
7.2 IPMQJS ALGORITHM

1) Start  
2) Assign value for each job as ‘j_i’ sorted in ascending order based on burst time.  
3) Resource ‘r_i’ in the queue that classified into three types namely queue QA, QB and QC.  
4) Calculate the multi queue using priority based on the burst time (BT) of the job.  
5) The sorted jobs are classified in to three types namely high priority, medium priority and low priority queue.  
6) /*scheduling for High priority */  
7) Based on the BT if job ‘j_i’ has high BT then assign the job value in to QA.  
8) Else  
9) /* scheduling for Medium Priority*/  
10) Based on the BT if job ‘j_i’ has medium BT then assign the job value in to QB.  
11) Else  
12)/* scheduling for Low Priority*/  
13) Based on the BT if job ‘j_i’ has low BT then assign the job value in to QC.  
14) Update the total number of jobs in to the job scheduler.  
15) Finally scheduled jobs are enter in to the cloud  
16) End

**Table 7.2 Input Data**

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Burst Time(MS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>50</td>
</tr>
<tr>
<td>J2</td>
<td>10</td>
</tr>
<tr>
<td>J3</td>
<td>20</td>
</tr>
<tr>
<td>J4</td>
<td>45</td>
</tr>
<tr>
<td>J5</td>
<td>17</td>
</tr>
<tr>
<td>J6</td>
<td>5</td>
</tr>
</tbody>
</table>

In the above Table 7.2 shows the input data for scheduling algorithms. It consists of six jobs with its BT (Ms).
In the above Figure 7.2 shows the job scheduling in FCFS Algorithm. The X-axis denotes the execution of the jobs and Y-axis denotes the processor needed to execute the jobs. The job id J₁ occupies full space, job id J₂ and J₃ during the time of execution it creates starvation. The job id J₄ also creates unused space and finally J₅ and J₆ are executed in FCFS policy creates starvation in large.
Figure 7.3 Job Scheduling in IPMQJS Algorithm

The above Figure 7.3 job scheduling in IPMQJS algorithm gives the job id which has higher burst time need very high processor, next the medium burst time has medium burst time and remaining small jobs are backfilled. The job id sequence J_6, J_2, J_5 are executed first, then job id J_3 execute, then J_4 complete their work. Finally job id J_1 utilise the full space with effective and improved resource utilization. It reduces the unused free space.

7.3 RESULTS AND DISCUSSION

The proposed IPMQJS algorithm is done with the help of CloudSim. It is used to simulate design and generate experimental model for create IaaS and application as a service. The important feature of CloudSim is self contained platform for modeling and simulation of large scale data centers on a unique physical computing node. The
architecture of CloudSim consists of four layers like SimJava, GridSim, CloudSim and user code. The user submitted jobs are entered in to the CloudSim process the jobs based on burst time, successful completion jobs are entered into the cloud computing. The various comparisons have been made to show their performance based on job completion time. IPMQJS system is better one when compared to the existing FCFS, SJF, Max-Min job scheduling algorithms.

7.4 COMPARISON OF IPMQJS WITH FCFS ALGORITHM

![Job Completion Ratio](image)

**Figure 7.4 Performance of IPMQJS with FCFS**

The above Figure 7.4 shows the performance of IPMQJS with FCFS. The X-axis denotes the number of cloudlet and Y-axis denotes the processing time of the job. The number of cloudlet is processed in FCFS is low when compared to IPMQJS. Hence IPMQJS provides better results.
7.5 COMPARISON OF IPMQJS WITH SJF ALGORITHM

The following Figure 7.5 shows the performance of IPMQJS with SJF. The job completion ratio of cloudlets is represented by X-axis and processing time is represented by Y-axis.

![Job Completion Ratio](image)

**Figure 7.5 Performance of IPMQJS with SJF**

If the size of cloudlets ranges increases the job completion ratio of IPMQJS provides better results when compared to BS.
7.6 COMPARISON OF IPMQJS WITH MAX-MIN ALGORITHM

In the following Figure 7.6 shows the performance of IPMQJS with Max-Min. The cloudlets are represented in X-axis and processing time is represented in Y-axis.

![Job Completion Ratio](image)

**Figure 7.6 Performance of IPMQJS with Max-Min**

The job completion ratio of Max-Min takes more for processing the input given jobs where as in IPMQJS takes less time to execute the same type of jobs. The graphical representation shows the IPMQJS provide efficient result when compared to Max-Min.
7.7 COMPARISON OF IPMQJS WITH EXISTING ALGORITHMS

The following Figure 7.7 shows the performance of IPMQJS with FCFS, SJF and Max-Min algorithms. The job completion ratio in cloudlets is represented by X-axis and processing time is represented by Y-axis.

![Job Completion Ratio Chart]

**Figure 7.7 Performance of IPMQJS with Existing Algorithms**

In terms of number of cloudlets and processing time IPMQJS provides better results when compared to other existing algorithms.
7.8 SUMMARY

This chapter summarizes the overall design of IPMQJS algorithm enhances the MQS. It explains the IPMQJS methodology to give priority for all the jobs in the queue based on their execution time, jobs are classified into three types of priority namely high, medium and low. Comparative analysis of IPMQJS with other traditional existing FCFS, SJF, Max-Min job scheduling is presented in this chapter. The research work provides QoS in terms of takes less to execute all the jobs and effective resource utilization. Hence the proposed job scheduling algorithm produces an optimum solution for job scheduling in cloud computing.