CHAPTER 5
CHAPTER 5
TRI QUEUE JOB SCHEDULING

5.1 INTRODUCTION

Optimized Resource Filling (ORF) algorithm is implemented by integrating small and medium job in a single queue called the smadium queue. In ORF algorithm the priority is given to smadium queue than the long queue. The proposed Tri Queue Scheduling (TQS) algorithm provides job allocation policy for cloud environment. To avoid the starvation within the process, an efficient TQS algorithm is essential for cloud computing. The proposed strategy groups all jobs in the queue, based on burst time to complete the jobs needed for the processor and time for allocation of resources to the computer nodes. The proposed methodology gives importance to all jobs using the dynamic quantum time based round robin fashion. To make the efficient utilization of jobs, the ORF scheduling algorithm produces the optimization for cloud scheduling problems. The efficiency of the program is decided by their complexity of the problem. It satisfies two conditions named as time and space complexity that decides the best scheduling algorithm.

5.2 TRI QUEUE JOB SCHEDULING ALGORITHM

TQS algorithm helps to minimize the time and space required for executing the program in the cloud environment. Allocation and sharing of resources properly to the processor is one of the crucial jobs in cloud computing. Efficient sharing of resources is decided by the metascheduler. In the traditional job scheduling algorithms namely FCFS, SJF, EASY and CBA lead to fragmentation during the scheduling process. To overcome this, it needs an efficient TQS algorithm for cloud computing. The proposed methodology grouping small, medium and long jobs in the queue, based on the processor needed and time for allocating the resources to the computers.

TQS algorithm gives importance to all the jobs using dynamic quantum time based RR Fashion. To make the efficient use of the resources, the proposed
scheduling algorithm achieves the optimization for cloud scheduling problems. The
clients are the king of the modern business jargon. Quality is not a single step
process. It is a continuous improvement process. Normally it is difficult to measure
the client requirements in cloud environment. The service provider gathers
information and monitors the changes of user needs. TQS method is used to triumph
over the problem faced in the existing methods. TQS is an enhanced method of
previous MQS model that provides the job scheduling algorithm in an effective
manner and selects the suitable resources. The ultimate aim of the proposed concept
is to utilize the free unused space and improve the performance of scheduling
algorithm.

TQS methodology is used to improve the performance and differentiating the
jobs into three queues using dynamic quantum time. It reduces fragmentation,
because it gives equal opportunity to all jobs. Client satisfaction is one of the major
factors that affect the service provider’s reputation and indirectly provides quality
system. In the customer point of view, it gives equal importance to all the three
different jobs. TQS algorithm reduces the starvation when compared to the existing
algorithm. User submitted jobs are sorted in the descending order based on burst time
and enters into the resource manager. The scheduler processes a job from the long
queue after its completion, it goes to next job. The process is continually executed
until all jobs are scheduled.

The main purpose of metascheduler is to select the best pool among the
available resources. It is difficult to identify the job sequence and execution time of
each job, because it is internet based where client needs are dynamic. To produce a
high quality of job scheduling system, the proper allocation of job is essential. The
available tasks are queued with the help of resource filling and producing improved
results. The proposed methodology used to improve the performance, classifying the
jobs into three different queues using dynamic quantum time. It avoids fragmentation,
because equal opportunity granted to all jobs. Customer satisfaction is one of the
major factors that affect the service provider’s reputation. In the client point of view it
gives importance to all jobs equally. User submitted jobs are sorted in the descending
order and entered into the resource manager. The scheduler processes a job from the

164
long queue after completion it goes to the next job and so on. The process is repeated until all jobs are scheduled.

The following Figure 5.1 shows the architecture of TQS algorithm. The users submitted are entered into the resource manager. The resource manager identifies the resource gap in the queue that is backfilled with the help of small jobs. Based on the time of execution the jobs are stored and formed in the long, medium and small queues. The scheduler processes the queued jobs using dynamic quantum time and finally the queued jobs are ready to be executed and entered into cloud environment.

![Diagram](image.png)

**Figure 5.1 Architecture of TQS**
Queue Forming Technique

Queues is classified into three types namely small, medium and long based on sorted order of the processing time and needed processor. The total submitted jobs are considered as 100%.

(i) Long queue stores first 40% of jobs.
(ii) Medium queue stores next 40% of jobs.
(iii) Small queue stores remaining 20% jobs.

The following Table 5.1 shows the jobs with processor needed. The proposed TQS algorithm concept is implemented by selecting of list of 10 jobs J1, J2, J3, J4, J5, J6, J7, J8, J9 and J10.

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Processor Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>6</td>
</tr>
<tr>
<td>J2</td>
<td>70</td>
</tr>
<tr>
<td>J3</td>
<td>4</td>
</tr>
<tr>
<td>J4</td>
<td>28</td>
</tr>
<tr>
<td>J5</td>
<td>32</td>
</tr>
<tr>
<td>J6</td>
<td>23</td>
</tr>
<tr>
<td>J7</td>
<td>50</td>
</tr>
<tr>
<td>J8</td>
<td>15</td>
</tr>
<tr>
<td>J9</td>
<td>17</td>
</tr>
<tr>
<td>J10</td>
<td>45</td>
</tr>
</tbody>
</table>

The following Figure 5.2 shows the jobs scheduling in existing system that are placed in three different queues. X-axis denotes the execution time and Y-axis implies processor needed. Job sequence with respect to the processor required as {J2, J7, J10, J5, J4, J6, J9, J8, J1 and J3}. It takes the largest job J2 in the first position of the queue and remaining space is filled by the Job J3 in the first queue. The second is filled by job J7 in the first position followed by the J10 and J3. The third queue is filled by {J6, J9, J5, J6 and J1} in the remaining space.
The following Figure 5.3 implicitly shows the jobs scheduling in TQS algorithm using dynamic quantum time technique granting opportunity for all the jobs. Execution time is represented in the X-axis and processor needed is scaled in Y-axis. The job completion ratio of TQS using dynamic quantum time shows the improved and satisfactory result when compared to the already existed algorithm in terms of processing time and number of cloudlets processed. Based on the time complexity and space complexity aspect the TQS algorithm provides better solution for the real time problems. The proposed TQS scheduling algorithm achieves the available jobs are scheduled in an optimized for cloud computing. It attains high resource usage and provides high quality solution for cloud environment.
5.3 RESULTS AND DISCUSSION

The performance of the proposed TQS algorithm is compared with the existing algorithms like FCFS and BS. With the help of CloudSim the above said are implemented and results are shown below.

Performance Metrics:

The various performance metrics are used to compare TQS with existing algorithms that are currently used in the cloud environment. This performance metrics are essential to prove the efficiency of the proposed algorithm.

◆ **Waiting Time** – The amount of time a job has to wait for execution in the batch queue.

◆ **Execution Time** – The amount of time a job has executed for completion.
◆ **Response Time** – The sum of the waiting time spent in the batch queue and the execution time of the job.

◆ **Makespan** – The total time that it takes for all jobs in a workload to finish on the cluster.

◆ **Throughput** – The number of jobs completed on the cluster per unit time.

### 5.4 Comparison of TQS with Traditional Backfill Algorithm

The following Figure 5.4 shows the performance of TQS with Traditional Backfill. The X-axis denotes the number of cloudlet and Y-axis denotes the processing time of the job.

![Job Completion Ratio](chart.png)

**Figure 5.4 Performance of TQS with Traditional Backfill**
In traditional algorithm is backfilled with the help of FCFS. The number of cloudlet processed by traditional backfill is low when compared to TQS.

5.5 COMPARISON OF TQS WITH BALANCE SPIRAL BACKFILL ALGORITHM

The following Figure 5.5 shows the performance of MQS with BS. The job completion ratio in cloudlets is represented by X-axis and processing time is represented by Y-axis.

![Job Completion Ratio](image)

**Figure 5.5 Performance of TQS with Balance Spiral Backfill**

If the size of cloudlets ranges below 1000, the BS merely provides better. In overall the size of cloudlet is more than 1000, TQS provides better results.
5.6 COMPARISON OF TQS WITH EXISTING ALGORITHMS

The following Figure 5.6 shows the performance of TQS with FCFS and BS. The job completion ratio in cloudlets is represented by X-axis and processing time is represented by Y-axis.

![Job Completion Ratio](image)

**Figure 5.6 Performance of TQS with Existing Algorithms**

In overall the TQS algorithm provides better results when compared to FCFS and BS.

5.7 SUMMARY

This chapter describes the concept of TQS. It is implemented in CloudSim and compared with the existing algorithms like FCFS and BS. The load balance is an important issue in the previous scheduling. It also provides the graphical representation based on the results obtained from the CloudSim. TQS algorithm helps to reduce load imbalance during the distribution of jobs without blocking. The effective use of the computer nodes occurs in an optimum manner.