CHAPTER 4
OPTIMIZED RESOURCE FILLING

4.1 INTRODUCTION

Optimizing the available resources effectively is the main objective of management. The resources are valuable one in terms of manpower, computer nodes, processor, network and so on. The existing algorithm like FCFS, SJF etc., produces more starvation and meets the service provider in under usage of resources. The proposed Optimized Resource Filling helps to manage the load balance in workload and properly utilize the resources and increases the unused available working space and reduces starvation, when compared to traditional and Balance Spiral method. To make the most efficient use of the resources, it achieves the optimization for cloud scheduling problems. This research work introduced a new concept called Smadium. The main objective of this research work is to produce high usage of available resources, balance the system and reduce system unused time and to improve the throughput of the system. Resource gap filling tries to fill the unused space created by the scheduler. The important criteria that decides the best scheduling algorithm is space usage, minimum waiting time and least turnaround time.

4.2 ORF TECHNIQUE FOR JOB SCHEDULING

The projected algorithm helps to improve the resource gap, reduces the system idle time, helps to attain high resource usage and provides quality system in cloud environment. The client sends request to the service provider and the scheduler classifies the jobs based on ascending order of their burst time. The combination of small and medium jobs in the queue called as Smadium and other remaining jobs are grouped in the long queue. Cloud computing is internet based dynamic environment that newly arrived jobs are updated in the queue manger and passes to scheduler. The main function of queue manager is handling the jobs to dispatch them in ORF based policy. Each job in the queue has a job id matched by the queue manager. The scheduler invokes the jobs to priorities them. The resource manager updates the new id in the jobs queue. The scheduler sends the client request to the system. The ORF
allocates the jobs and schedule using RR fashion. If the new jobs are arrived, the tag number added to the queue. The scheduled jobs are entered and stored into the resource allocation with the help of resource agent. Based on the burst time, jobs are distributed in ordered manner. It can select a multiple jobs from both smadium and long queue in the queue with the help of resource filling technique to maximize the efficient use of system resources. It produces low average waiting time and high usage of resources than the traditional system.

![Diagram](image)

**Figure 4.1 Architecture of ORF**

The above Figure 4.1 shows the architecture of ORF. Best job scheduling is to give better system metrics like utilization of space and user metrics like less waiting time and turnaround time. The user submitted jobs are entered into the scheduler. It arranges the jobs in ascending order. Then grouping the small and medium jobs
grouped as smadium in one queue and remaining jobs in long queue. The arrived jobs are updated in the queue and scheduler. Queue manager is responsible for handling the scheduled jobs and dispatching the request. Scheduler sends the job id matched by the queue service. It invokes the job scheduler to prioritize the jobs. The jobs are updated in the resource manager.

The queue allocates the resources to the computer nodes. The scheduler dispatches the user request to the computer nodes. The resource manager allocates the job and scheduled them in round robin fashion. If the new jobs are arrived the queue manager dynamically changed them.

The queued jobs are entered into the resource allocation by the resource broker in random fashion based on burst time. The finishing time of job, it allocates them the remaining jobs in order. The CloudSim resource simulator uses internal events to simulate the execution and allocation of PEs’ share to Cloudlet jobs. CloudSim schedules a new internal event to be delivered at the forecasted time of the remaining Cloudlets. The simulation of the MQS algorithm and the Cloudlets’ execution in three different queues are queued then the following algorithm illustrates the step by step procedure of the proposed ORF.

4.3 ALGORITHM OF ORF

1. Collect all jobs and its corresponding burst_time
2. If sum ((burst_time) > 0) follow next step else step 11
3. Burst_time of all jobs are added and calculated maximum burst_time needed
4. Create queue [ Smadium_queue, Large_queue]
5. If size (Smadium_queue) < Maximum_size then store the jobs in Smadium_queue else
6. Store the job in Large_queue
7. If P1=ready allocate the processor check whether any space remaining
8. If space (Pn) = = remaining space allocated the processor else
9. Allocate new space for processor
10. Repeat the step 7 to 9 until all jobs are executed
11. Complete the process.
The proposed ORF scheduling model used by the algorithm consists of resource manager, scheduler and pool of resources. It has a queue manager to allocate the jobs. Algorithm rearranges the job queue according to the increasing order of the remaining execution time of jobs. Each job declares its type and number of resources required at the time of arrival in the queue. Status of resources is monitored by the resource manager. Its dynamically interacts with the scheduler and resource pool. When a job arrives at time ‘t’ in the cluster, it gets placed at a level according to its expected runtime.

The following Figure 4.2 shows the job scheduling in existing technique with nine different types of jobs that have various processor needed. In the first iteration P₁, P₂ and P₃ are executed with starvation, second iteration P₄ is alone executed, third iteration P₅, P₆ and P₇ with some starvation and Final iteration was P₈ and P₉.

![Job Scheduling Diagram](image)

<table>
<thead>
<tr>
<th>Job Needed Processor</th>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₄</th>
<th>P₅</th>
<th>P₆</th>
<th>P₇</th>
<th>P₈</th>
<th>P₉</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>25</td>
<td>30</td>
<td>100</td>
<td>35</td>
<td>20</td>
<td>32</td>
<td>33</td>
<td>10</td>
</tr>
</tbody>
</table>

**Figure 4.2 Job Scheduling in Existing Technique**

The following Figure 4.3 shows job scheduling in ORF Technique, when job gets its turn of execution, it tries to have access of desired resources with specifications. Scheduler interacts with the resource manager and gets the notification
of free resources. It decides whether the job can be placed into execution or not. If job cannot be scheduled, it gets into a waiting state. Scheduler tries to backfill a lower priority job. Resource manager continuously updates the pool to get the recent information about resource status.

![Diagram](image)

**Figure 4.3 Job Scheduling in ORF Technique**

The proposed research work orders the job arrival sequence in ascending order. The jobs are placed in two different queues. Job sequence with respect to the processor needed as \(\{P_1, P_2, P_3, P_4, \ldots, P_9\}\) is filled the remaining space is used in the smadium first queue. Job id \(\{P_5, P_7, P_8\}\) in the next queue and takes the largest job \(P_4\) in the fourth position of the job sequence.

### 4.4 RESULTS AND DISCUSSION

The simulation of scheduling in time-shared resources is done with the help of CloudSim [70] resource simulator uses internal events to simulate the execution and
allocation of PEs’ share to Cloudlet jobs. When jobs arrive, time-shared systems start their execution immediately and share resources among all jobs. Whenever a new Cloudlet job arrives, scheduler updates the processing time of existing Cloudlets and then adds this newly arrived job to the execution set. The schedule is an internal event to be delivered at the earliest completion time of the smallest job in the execution set. The job waits for the arrival of events. A complete algorithm for simulation is time-share scheduling and execution. If a newly arrived event happens to be an internal event whose tag number is the same as the most recently scheduled event, then it is recognized as a job completion event. Depending on the number of Cloudlets in execution and the number of PEs in a resource, CloudSim allocates the appropriate PE share to all Cloudlets for the event duration using the algorithm. It should be noted that Cloudlets sharing the same PE would get an equal amount of PE share.

The completed Cloudlet is sent back to its originator (broker or user) and removed from the execution set. CloudSim schedules a new internal event to be delivered at the forecasted earliest completion time of the remaining Cloudlets. The simulation of the time-share scheduling algorithm and its Cloudlets’ execution are implemented.

The performance of the proposed ORF compared with existing FCFS and CBA algorithm implemented with the help of CloudSim.
4.5 COMPARISON OF ORF WITH TRADITIONAL BACKFILL ALGORITHM

In the following Figure 4.4 shows the performance of ORF with Traditional Backfill. The X-axis denotes the number of cloudlet and Y-axis denotes the processing time of the job.

![Job Completion Ratio](image)

**Figure 4.4 Result of ORF with Traditional Backfill**

The number of cloudlet is traditional backfill processed in the CloudSim that is low when compared to ORF which provides better result.
4.6 COMPARISON OF ORF WITH COMBINATIONAL BACKFILL ALGORITHM

The following Figure 4.5 shows the performance of ORF with CBA. The cloudlets are represented in X-axis and processing time is represented in the Y-axis for scheduled jobs completion ratio. ORF provides efficient result when compared to CBA.

![Job Completion Ratio Graph]

Figure 4.5 Result of ORF with CBA
4.7 COMPARISON OF ORF WITH EXISTING ALGORITHMS

The following Figure 4.6 clearly shows the results of ORF with existing algorithms. In cloudlet X-axis denotes the number of cloudlet and Y-axis represents processing time.

![Job Completion Ratio](image)

**Figure 4.6 Result of ORF with existing algorithms**

Job completion ratio of different scheduling algorithms is based on average waiting time, throughput and average turnaround time. This satisfies the requirement of an efficient scheduling algorithm.
### Table 4.1 Comparison of Various Scheduling Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average Waiting Time (ms)</th>
<th>Throughput</th>
<th>Average turnaround Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCFS</td>
<td>7</td>
<td>25%</td>
<td>15</td>
</tr>
<tr>
<td>SJF</td>
<td>4</td>
<td>29%</td>
<td>12</td>
</tr>
<tr>
<td>CBA</td>
<td>3</td>
<td>30%</td>
<td>9</td>
</tr>
<tr>
<td>ORF</td>
<td>2</td>
<td>35%</td>
<td>7</td>
</tr>
</tbody>
</table>

The proposed hybrid algorithm achieves higher result and clearly shown the above comparison Table 4.1, ORF maximizes the throughput by about 15% and minimizes the average waiting time by 5% and decreases the average turnaround time by 8%.

#### 4.8 SUMMARY

This chapter summarizes the drawbacks of traditional algorithm and the concept of proposed algorithm ORF. The comparative analysis of existing algorithm and ORF are tested by using CloudSim. It explains the overall design of the newly proposed ORF approach to identify unused space in the scheduling and to achieve refilling the resource gap with the help of grouping approach.