DRR Based Job Scheduling for Computational Grid and its Variants

Shailesh Saxena  
Dept. Of CS & IT  
SRMSWCET  
Bareilly, India

Mohd Zubair Khan  
Dept. Of CS  
Invertis University  
Bareilly, India

Dr. Ravendra Singh  
Dept. Of CS&IT  
MJP Rohilkhand University  
Bareilly, India

Anuj Kumar  
Dept. Of CS & IT, SRMSWCET  
Bareilly, India

Abstract:
Scheduling of jobs and resources are two essential features of the Grid computing infrastructure. To improve the global throughput of these environments, an effective and efficient task allocation algorithm is fundamentally required. However for Grid computation we have some challenges like heterogeneity, scalability and adaptability. So to handle these challenges we used a dynamic hierarchical model to represent the architecture of the grid computing system in order to manage jobs and resources. This model was characterized as- it defines the hierarchal structure of physical and virtual computing elements and it also supports heterogeneity and scalability of computing elements. In this paper, we proposed a DRR (Dynamic Round Robin) job scheduling algorithm for grid computing system with preemptable jobs allocation (Round Robin) using dynamic time quantum according to the priority of the jobs, which is to be change with every round of execution of the jobs. The main benefit of this idea is to decrease the pre-emption and context switching thereby reducing the overhead and saving of memory space in the grid system. It gives the better performance in terms of reducing the number of context switches, average waiting time and average turnaround time and also increases the CPU utilization.

Keywords: Grid computing, hierarchical structure, Dynamic Round Robin, Dynamic time quantum, preemption, context switching

I. Introduction
Traditionally, users have developed scientific applications with a parallel computer in mind, assuming a homogeneous set of processors linked with a homogeneous and fast network. However, grids [16] of computational resources usually include heterogeneous processors, and heterogeneous network links for deploying large-scale and resource-intensive applications. Grid computing raises challenging issues in many areas of computer science, and especially in the area of distributed computing, as Computational Grids cover increasingly large networks and span many organizations. The problem of assigning each job to different processors of a distributed computing in grid system has a major impact on the resulting performance. This study is related to the pertinent previous work in the literature, since efficient resource utilization, load balancing and job scheduling are topics that attract the attention of the researchers, as computational Grids have become an emerging trend on high performance computing. Most studies in the field of resource allocation schemes aim at efficiently utilizing the otherwise unutilized computing power spread throughout a network.
Different global objectives could be considered, such as minimization of mean job completion time, maximization of resources utilization (e.g., CPU time), and minimization of mean response ratio, while in most cases load balancing among resources is considered. Job-allocation has an important role in parallel system for minimum time execution. If jobs are equal to the processors then each processor has a single job. If jobs are greater than the processors then numbers of jobs are assigned to a processor. To improve the efficiency and best utilization of processors, the main processor (normally called the master processor) knows the information of all other processors. It takes decisions to allocate the jobs to other processor using single job queue, [6] which reduces the execution time of genetic algorithm. This type of algorithm is called global parallel Genetic Algorithm which is implemented by using the concept of master-slave programs. Job scheduling algorithms are used for scheduling so that it arranges jobs in a schedule then allocate on the processors. Job allocating algorithm is used in the parallel processors and heterogeneous environment. The algorithm [7] used to divide the original job-graph on the basis of some relation. This pattern is called the clustering which uses different region of job on the basis of some pattern. After clustering, job graph uses reassignment for mapping the job group. The algorithm uses to divide the jobs into several job groups on the basis of connection component algorithm. Job groups are assigned [8] to processors of the cluster with the help of master processor. This assignment is based on the queue property. Cost is also reduced because every time master processor assigns job groups to other processor. For minimizing the job executing time. The algorithm uses the assignment of job group on processors. So Dynamic load is balanced on heterogeneous multiple processors [9]. Different jobs are assigned and scheduled in a network which is made by different processors [11]. So Allocation of jobs on multiple processors is based on parallel system [10]. But this paper mainly describes the job allocation through master-slave algorithm using DRR (Dynamic Round Robin) scheduling. DRR scheduling uses the different time-quantum for scheduling according to the priority of the jobs. The remaining of this paper is organized as follows, in Section II we briefly describe the related research literature. section III, we define the architecture model of the grid system and different parameter which is used for describing the efficient job scheduling algorithm while the proposed DRR scheduling is described in detail in Section IV. Section V describes experiment based analysis. Section VI gives the conclusions.

II. RELATED WORK

K.Somasundaram, S.Radakrishnan and M.Gowathyanayagan [1] stated highest response next (HRN) scheduling algorithm in order to correct some of the weakness in both shortest jobs first and FCFS. HRN is a non preemptive algorithm where priority of job is function of job’s service time and also amount of time job has been waiting for service. Somasundaram, S. Radakrishnan [2] proposed a Swift Scheduler and compared it with First Come First Serve (FCFS), Shortest Job First (SJF) and with Simple Fair Task Order (SFTO) based on processing time analysis, cost analysis and resource utilization. Daphne Lopez, S. V. Kasmir raja [3] has described and compared Fair Scheduling algorithm with First Come First Serve (FCFS) and Round Robin (RR) schemes. Mayank Kumar Maheshwari and Abhay Bansal [4] presented the design of new scheduling algorithm Priority Scheduler. The proposed Priority Scheduler which completes a task by using highly utilized low cost resources with minimum computational time. Gaurav Sharma and Preeti Bansal [5] enhanced the
Min-Min algorithm by classifying it according to the QOS parameters. QOS guided Min-Min scheduling algorithm outperform the traditional Min-Min heuristic on the same set of task.

III. System Model of the Grid
The basic model (Figure 1) represents a logical view of the smallest possible Grid, define as one cluster. This model, is defined by a two levels tree. The leaves of this tree, define as worker node correspond to each other in the cluster. Its root, called cluster manager, represents a virtual tree node associated with the cluster, whose role is to allocate the tasks to the worker nodes of the cluster.

The job scheduling algorithm proposed in this paper is based on the mapping model as shown in figure 2, of any Grid topology into a forest structure by aggregating several basic models.

**Grid Manager (Level 0):** In this first level (top level), we have a virtual node that corresponds to the root of the tree. It is associated to the Grid and it manages the workload on the whole Grid.

**Site Manager (Level 1):** In this second level contains G virtual nodes, each one associated to a physical cluster of the Grid. In job scheduling, this virtual node is responsible to manage its sites.

**Cluster Manager (Level 2):** In this third level, we find S nodes associated to physical sites of all clusters of the Grid. The main function of these nodes is to manage the workload of their worker nodes.

**Worker Node (Level 3):** At this last level (leaves of the tree), we find the M physical Computing Elements of a Grid linked to their respective sites and clusters.

For the job scheduling the following parameters are considered:

**Context Switch** - A context switch is computing job of storing and restoring state of a CPU so that execution can be resumed from same point at a later time. Context switch are usually computationally intensive, lead to wastage of time, memory, scheduler overhead so much of the design of operating system is to optimize these switches.

**Throughput** - Throughput is defined as number of jobs completed per unit time. Throughput will be slow in round robin scheduling implementation. Context switch and throughput are proportional to each other.

**CPU Utilization** - We want to keep the CPU as busy as possible.

**Turnaround Time** - Turnaround time is sum of periods spent waiting to get into memory, waiting in ready queue, executing on CPU and doing input output. It should be less.

**Waiting Time** - Waiting time is the amount of time a job has been waiting in ready queue. The CPU scheduling algorithm does not affect the amount of time during which a job executes or
does input-output; it affects only the amount of time that a job spends waiting in ready queue. **Response Time**—Response time is the time it takes to start responding, not the time it takes to output the response. Large response time is a drawback in round robin architecture as it leads to degradation of system performance.

For a good scheduling algorithm, CPU utilization and throughput must be maximum and waiting time, turnaround time, response time and context switches must be minimum.

### IV. Proposed Scheduling

The scheduler maintains a queue of ready Processes and a list of blocked and swapped out processes. The scheduling process based on Round Robin algorithm is described as-

**Step1:** The scheduler maintains a queue of ready Processes and a list of blocked and swapped out processes.

**Step2:** The PCB of newly created process is added to end of ready queue. The PCB of terminating process is removed from the scheduling data structures.

**Step3:** The scheduler always selects the PCB at head of the ready queue and allots the fixed TQ.

**Step4:** When a running process finishes its TQ, a) It moved to end of ready queue (if TQ < BT).

**Step5:** The event handler performs the following actions, n, a) When a process makes an input-output request or swapped out, its PCB is removed from ready queue to blocked/swapped out list. b) When input-output operation awaited by a process finishes or process is swapped in its PCB is removed from blocked/swapped list to end of ready queue.

**Dynamic TQ based DRR Scheduling**

In the proposed scheduling, initially for every job a different Time quantum (TQ) is assigned on the basis of the priority of that job and the range of the total job’s burst time as shown in Table 1. The Range for the given set of jobs is calculated as-

$$\text{Range (R)} = \frac{\{BT \text{ (Max)} + BT \text{ (Min)}\}}{2}$$

Where R = Range of burst time of total jobs.

BT = Burst Time of a job.

On the basis of the range we assign a TQ for every job which is calculated as –

$$\text{TQ (i) of process P(i)} = \left(\frac{R \times N}{\text{Pr} \times P_{\text{max}}}\right)$$

N = Total no of jobs

Pr = priority of a job

Pmax = Maximum priority

<table>
<thead>
<tr>
<th>Process No</th>
<th>BT</th>
<th>Pr</th>
<th>R</th>
<th>Pmax</th>
<th>TQ(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>25</td>
<td>2</td>
<td>15</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>P2</td>
<td>5</td>
<td>3</td>
<td>15</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>P3</td>
<td>15</td>
<td>1</td>
<td>15</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>P4</td>
<td>8</td>
<td>5</td>
<td>15</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>P5</td>
<td>10</td>
<td>4</td>
<td>15</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

**Gantt chart according DTQ**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>8</th>
<th>13</th>
<th>28</th>
<th>31</th>
<th>35</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>43</td>
<td>46</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dynamic TQS based DRR Scheduling

Now the context switching and turnaround time of the jobs will be reduced because every job have their different TQ, but some time the average waiting time will be increase because of the large TQ is allotted to some jobs according to their priority. So in this proposed scheduling a different time quantum slice TQS will be allot for every job in every round of execution on the basis of their initial time quantum TQ value to minimize the waiting time as shown in Table 2. The calculation of TQS in every round for every job will be describe as follow-

Let ‘TQSi’ is the time quantum slice in round i. The number of rounds i varies from 1 to n, where value of i will be increased by 1after every round till the job is not complete i.e burst time is not equal to null.

Step1: Calculate initial TQ for all the processes present in the ready queue.
Step2: for i = 1 to n , TQS of each jobs will be
   for 1st round (i=1) ,
   TQSi = TQ/2
   for further rounds (1 < i < n) ,
   TQSi = TQS i-1 + ½*TQS i-1
Step3: calculate remaining burst time of the job
   BT = BT – TQSi
Step4: If (BT – TQSi) > 2 , repeat step2 and step3,
Step5: Otherwise, TQSi = remaining burst time.
Step6: End, If (TQSi = 0)

Table 2. Calculation of TQS for proposed DRR scheduling

<table>
<thead>
<tr>
<th>Process No</th>
<th>BT</th>
<th>TQ</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>P2</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P4</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P5</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

Gantt chart according DTQS

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>7</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>27</td>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>37</td>
<td>40</td>
<td>49</td>
<td>52</td>
<td>57</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

V. RESULT AND ANALYSIS

Scheduling algorithms described in the previous section were implemented and tested in GridSim. In which we take 5 jobs in form of gridlets, schedule over one resource which have 3 machines and each machine have 3,5and 9 processing elements of their own capacities. The initial information of the jobs is already described in the table 1. Table 3 have the values of waiting time and turnaround time for each job using RR, DTQRR and DTQSRR. The values of average waiting time, average turnaround time and context switching of all 5 jobs are describe in table 4.

Table 3: Waiting and Turnaround Time of Jobs using RR, DTQRR & DTQSRR

<table>
<thead>
<tr>
<th>Job</th>
<th>RR</th>
<th>DTQR</th>
<th>DTQSRR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Waiting Time</td>
<td>Turnaround Time</td>
<td>Waiting Time</td>
</tr>
<tr>
<td>P1</td>
<td>4</td>
<td>7</td>
<td>15</td>
</tr>
<tr>
<td>P2</td>
<td>25</td>
<td>27</td>
<td>34</td>
</tr>
<tr>
<td>P3</td>
<td>34</td>
<td>37</td>
<td>40</td>
</tr>
<tr>
<td>P4</td>
<td>52</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4: Comparison of RR, DTQRR & DTQSRR

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average TAT</th>
<th>Average WT</th>
<th>Context Switching</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>46.6</td>
<td>31.6</td>
<td>18</td>
</tr>
<tr>
<td>DTQRR</td>
<td>42.2</td>
<td>32.8</td>
<td>12</td>
</tr>
<tr>
<td>DTQSRR</td>
<td>44.6</td>
<td>28</td>
<td>14</td>
</tr>
</tbody>
</table>

Figure 1: Waiting Time Comparison of 5 Jobs

Figure 2: Turnaround Time Comparison of 5 Jobs

Figure 3: Comparison of RR, DTQRR and DTQSRR scheduling

VI. CONCLUSION

Grid computing can solve more complex tasks in less time and utilizes the resources efficiently. To make grid work properly, best job scheduling strategies have to be employed. Scheduling helps the jobs to get resources properly. In this paper we have proposed a scheduling algorithm DRR with two types of time quantum values DTQRR and DTQSRR. Here we also done the comparison of proposed DRR with normal RR scheduling in terms of
waiting and turnaround time of jobs. Average waiting time (AvgWT), average turnaround time (AvgTAT) and context switching (CS) of jobs is also calculated. We have considered here 5 jobs, all arrive at time 0. DTQRR gives less CS and AvgTAT and DTQSRR gives less AvgWT in comparison to normal RR scheduling. Future work can be based on this algorithm modified and implemented for those jobs arrive at different time.

REFERENCES