Deadline based Job Scheduling for Multi-Cluster Grid

Shailesh Saxena
Research Scholar
MJPRU, Bareilly, India

Mohd Zubair Khan
Department of CS
DAS, Taibah University
Madina Al Munawwarah, Saudi Arabia

Ravendra Singh, PhD
Deptt. Of CS & IT
MJPRU, Bareilly, India

Rohit Saxena
Deptt. Of CS & IT,
SRMSWCET, Bareilly, India

ABSTRACT
Application areas of multi cluster grid are increases day by day because of the seamless and scalable access to wide-area distributed resources in grid environment. Multi cluster grid allows sharing, selection and aggregates of geographically resources over heterogeneous and distributed locations. But some issues arise in multi cluster grid due to its environment. Scheduling a job on the most suitable computational resource of the grid is one of the most important issues in grid environment. To handle this issue a new approach of scheduling is proposed in this paper which is based on priority of completion deadline of the jobs because some time the job execution have an importance only when it complete under the deadline define by the user on the basis of working circumstances as a real time scenario, which helps scheduling of jobs on computational environment of multi cluster grid in very effective manner.

Keywords
Multi Cluster Grid, Completion Deadline, Grid environment, Job Scheduling, Scheduling Algorithm

1. INTRODUCTION
In the view of multi cluster grid, the computing becomes ubiquitous, so the access of computing resources like processors, storage, and data through the users or client applications in such environment goes beyond the realm of personal computers. So grid environment can be described as a heterogeneous distributed environment in which all the spread resources are coordinated through the distributed methodology [1]. The main aspect of multi cluster grid computing that is to be considered is the dynamicity of resources. Grid systems define the sharing, selection and aggregation of distributed resources across multiple administrative domains based on their availability, performance and QoS requirements of users [2].

Multi cluster grid is a type of grid in which heterogeneous CPU's of different clusters are considered as main resources of the grid system that are handled according to the working conditions of that CPU's. So the main issues in multi cluster grid computing are how to define working complexity, access and scheduling of resources. To handle such type of issues a grid system required a middleware support to perform uniform access, resource management and job scheduling. Job Scheduling is a technique through which we use to define the most appropriate computational resource for any job to complete its execution either in terms of waiting time, turnaround time or cost. The Scheduler plays an important role in computational grids to manage the jobs and resources. Scheduler first selects the appropriate computational resource for any job and then assigns that resource to the job. Job Scheduling is done in such manner, so that all jobs will execute efficiently and the use of all resources will also be efficient. The main objective of scheduling is to reduce the completion time of an application by properly allocating the jobs to the processors.

The further paper is organizes as follows: Section 2 discusses previous works in the area of job scheduling in computational grid. Section 3 presents scheduling architecture over which the proposed scheduling algorithm will be worked. Section 4 provides a detailed simulation description of the proposed algorithm. Section 5 provides the comparison of our proposed algorithm with some existing algorithm. At the end of this paper section 6 provides the conclusion on the basis of section 5.

2. RELATED WORK
In 2007[5] K.Somasundaram, S.Radhakrishnan and M. Gowathyanayagan published a paper in which they stated a scheduling algorithm for grid system based on highest response next (HRN) in order to overcome some weakness in FCFS and SJF scheduling. It is a non preemptive algorithm where priority of any job is defined through the service time of that job and also the amount of time that job has been waiting for its service. Similarly in 2009 [6] K. Somasundaram and S. Radhakrishnan proposed a Swift Scheduler and compared it with FCFS, SJF and Simple Fair Task Order (SFTO) scheduling based on processing time analysis, cost analysis and resource utilization. In 2009[7] Daphne Lopez, S. V. Kasmirraja has described Fair Scheduling algorithm to address the fairness issues by reducing the service time error and compared it with FCFS and Round Robin scheduling in terms of Cost, completion time and the error. This scheduling is basically works according the time quantum, so if time quantum is large then it works like FCFS and if it small then overhead of preemption is increased. In 2011[8] Sunita Bansal, Bhavik Kothari and Chittaranjan Hota, describe an efficient approach of task scheduling in a grid environment, without having prior information on workload of incoming tasks. This algorithm is an enhancement to the existing round-robin heuristic approach by prioritizing tasks eligible for replication. In 2011[9] M. K. Maheshwari and Abhay Bansal presented the design a new Priority Scheduler for scheduling algorithm. The proposed Priority Scheduler which assign a resource to any task by using highly utilized low cost resources with minimum computational time. In 2012[10] Gaurav Sharma and Preeti Bansal, enhanced the Min-Min algorithm by classifying it according to the QoS parameters of the users. On the basis of user's QoS requirements the proposed scheduling algorithm outperform the traditional Min-Min heuristic on the same set of task.

3. SCHEDULING ARCHITECTURE
Job scheduling is effortless procedure in homogenous environment because every job have same effect over any computing resource. But in heterogeneous environment the job have different effect over different processor, so job Scheduling is a very tremendous procedure in multi cluster grid. In such environment a scheduler is required who perform scheduling to the jobs as their real time requirements. The grid architecture for the proposed scheduling shown in Fig1, describe as-

A. Grid Users: Initially a grid user submits the jobs to the computational grid for execution. In the grid a GRM (resource manager) handles these jobs for further allocation on processors.

B. Grid Resource Manager (GRM): Generally a grid user does not able to perform job scheduling directly in multi cluster grid environment because of the distributed nature of the grid. So there will be a resource manager named as GRM who discover suitable computing resources for any job and allocate them to that job for their computation. GRM find the appropriate computing resource for the jobs, through the communication with the GIS that maintain the status of all the resources in the grid system.

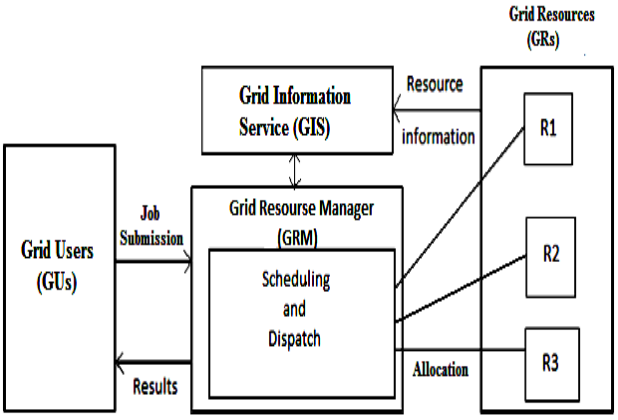

Fig1. Proposed Architecture of Grid System

C. Grid Information Service (GIS): GIS maintains the status of all resources whether they idle or busy and communicate this information to the GRM for scheduling the jobs when that is needed. After receiving the information regarding the resources, GRM take decision about the best computing resource for any job which wants their execution. When GRM allocate any resource, GIS update the status of that resource from idle to busy. After successful completion of a job, the resource becomes free, so the status of that resource is again updated from busy to idle. The procedure will follow until all the jobs are not executed completely.

4. PROPOSED WORK

Earliest Completion Deadline with Shortest Remaining Time Schedule First: ECDSRTF is an algorithm through which jobs will be schedule according to the priority of their completion deadline and remaining time. Using this algorithm the scheduler will give the preference for allocating the computing resource to that job whose deadline of completion is earlier than other jobs. This procedure will be repeated until the job queue of the grid become empty. If more than one job have same completion deadline then it is a case of tie. In such case the resource will allocated to that job which have shortest remaining time among them.

For implementation we define a multi cluster grid of 3 clusters with 2 processors each, in which 10 jobs are scheduled at a time. As shown in Fig 2, the proposed model describe clusters as CL1, CL2, CL3 and their processors as (P_{11},P_{12}), (P_{21},P_{22}) and (P_{31},P_{32}) respectively.

In proposed ECDSRTF algorithm, every job will be entered with their execution time along with the deadline of completion. Initially every processor of each cluster will be ideal, so the GRM define allocation of the processor for first 6 jobs on sequential basis that means it directly schedule the first 6 job to the processor P_{11}, P_{12}, P_{21}, P_{22}, P_{31} and P_{32} respectively. For the remaining jobs the GRM first compare their deadline with the current job's deadline on the processors. If GRM find any processor whose current job's deadline is greater than the deadline of the arriving job, then GRM allocate that processor to the next job through switching mechanism and current job will perform the wait. But if the current jobs of all processor have small completion deadline value than the next arrival job's deadline value, then all processors will continue the execution of their current jobs. This process will continue until the all jobs get the processor and complete their execution.

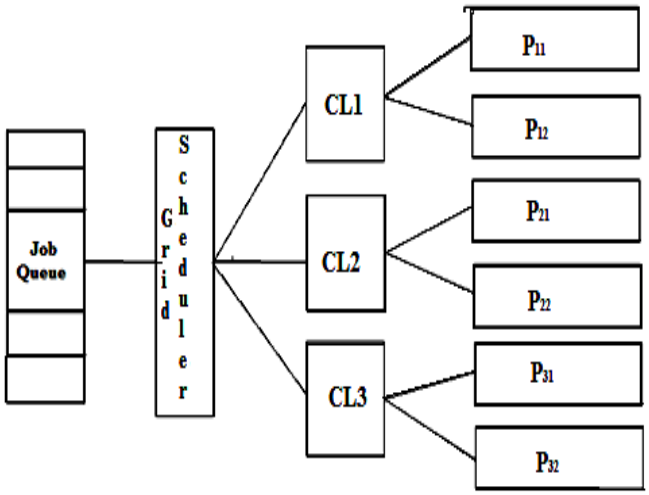

Fig2. ECDSRTF Scheduling Model

Proposed Scheduling Algorithm*:*

Step 1: Initially no. of processor (N) = 6
and no. of jobs (J) = 10

Step 2: Enter job's Execution Time and Deadline values.

Step 3: First six jobs will be assign sequentially to P_{11}, P_{12}, P_{21}, P_{22}, P_{31}, P_{32} respectively & initially SP[j] = 0

Step 4: for remaining jobs (J = 7 to 10)

```
for (i=7; i<=10: i++ )
    for (j=1 ; j<= 6 ; j++)
        If ((P[j]==0) && (d[j] > d[i]))
```

- Assign job J[i] to processor P[j]
- P[j] will switch to the new arrive job & execute it. Current job wait for the completion of new job
- Calculate WT & TAT of each job
- Set SP[j] =1 & SJ[i] =1.

Step 5: For every job, if SJ[i]=0, then allocation will define using comparing remaining time of each job and assign the processor to that job whose remaining time will be small.

Step 6: Calculate WT and TAT of job.

5. COMPARISON AND ANALYSIS

On the basis of simulation described in the previous section, the proposed ECDSRTF algorithm compare with two other scheduling algorithms i.e *First Come First Schedule* and *Shortest Job Schedule First*.

5.1 First Come-First Schedule (FCFS)

It is an effort less job scheduling algorithms in which resource manager provide the resource to any job according to their arrival. It means a job which one submitted first in the grid system through the grid user will allocate the resource first. Once a job gets resource; it execute over that resource till the full completion. So it is define as a non preemptive by nature, due to this the wait time of other jobs will be increase.

5.2 Shortest Job Schedule First (SJF)

In SJF scheduling, the resource manager allocate the resource to that job whose execution time is shortest than others. It is a preemptive scheduling algorithm so whenever the grid user submits a job which is shortest than the jobs executing in the grid than grid manager preempt any job and provide the resource of that job to the new one. If every next job will shortest than the previous job then the overhead of grid manager will increase.

In this paper the proposed ECDSRTF scheduling compared with FCFS and SJF scheduling algorithms on the two main performance measures i.e. wait time (WT) and Turnaround Time (TAT). We also calculate AWT (average waiting time) and ATRT (average turnaround time) of jobs. Table 1 has the values of waiting time and turnaround time for all 10 jobs using ECDSRTF, FCFS and SJF. The values of average waiting time, average turnaround time of all scheduling algorithms are described in table 2. Which are calculated as-

$$Avg.\ WT = \sum_{i=1\ to\ n} (WT)i / n$$

$$Avg.\ TRT = \sum_{i=1\ to\ n} (TRT)i / n$$

Table 1. WT and TAT of Jobs using EDSRTF, SJF & FCFS

Jobs	ECDSRTF		SJF		FCFS	
	Waiting Time	Turn around time	Waiting Time	Turn around time	Waiting Time	Turn around time
1	1	5	0	4	0	4
2	0	3	0	3	0	3
3	0	2	6	8	0	2
4	0	1	4	5	0	1
5	2	7	0	5	0	5
6	3	7	0	4	0	4
7	1	7	0	6	4	10
8	0	1	4	5	3	4
9	0	2	5	7	2	4
10	0	3	0	3	1	4

Table 2. Avg. WT and Avg. TRT of Jobs using ECDSRTF , SJF and FCFS

Algorithms	Average Waiting Time	Average Turnaround Time
ECDSRTF	0.7	3.8
SJF	1.9	5.0
FCFS	1.0	4.1

Fig 3 and Fig 4, shows the comparative analysis between ECDSRTF, FCFS and SJF for 10 jobs according to waiting time and turnaround time.

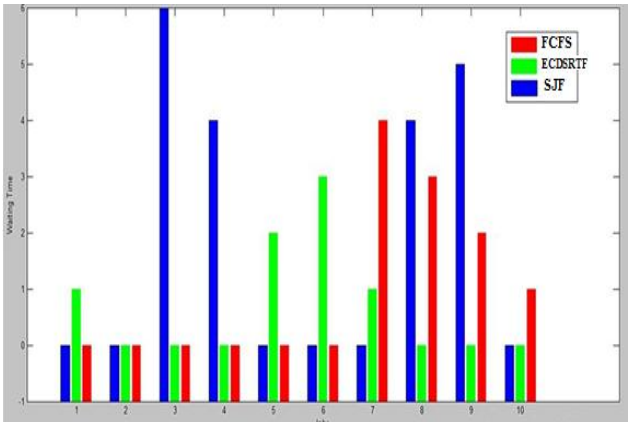

Fig 3. Waiting Time Comparison of 10 Jobs

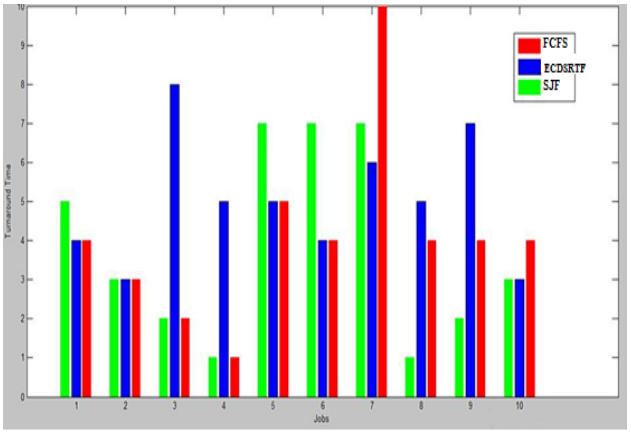

Fig 4. Turnaround Time Comparison of 10 Jobs

6. CONCLUSION and FUTURE WORK

In multi cluster grid more complex jobs can complete in less time and resources become utilizes in more effective way. An effective job scheduling algorithm is more helpful to describe such type of multi cluster grid system who properly allocates the suitable computing resource to the particular job for their execution. The proposed paper describe *ECDSRTF* scheduling algorithm, based on completion deadline of the submitted jobs. This proposed scheduling also compare with two other scheduling algorithms like FCFS and SJF in terms of wait time and turnaround time of jobs. Avg. waiting time (AWT) and avg. turnaround time (ATRT) of jobs are also calculated for comparison. The result of this comparison shows that the ECDSRTF algorithm have less avg. wait time and avg. turnaround time than FCFS and SJF. So ECDSRTF is more efficient among them.

For simulating the above comparison the proposed scheduling algorithm, schedule 10 jobs over multi cluster grid of 6 processors. In future we also simulate the results of this scheduling through either increasing the number of jobs which are to be schedule or increasing the number of processors in the grid over which the scheduling will perform.

7. REFERENCES

[1] Mrs. Radha, Dr.V.Sumathy, "A Detailed Study of Resource Scheduling and Fault Tolerance in Grid", 2011.

[2] N.A. Azeez1; A.P. idoye; A.O. Adesina; K.K. Agbele; Iyamu Tiko, and I.M. Venter, "Peer to Peer Computing and Grid Computing: Towards a Better Understanding", 2011.

[3] Manoj Kumar, Mishra Prithviraj Mohanty, G. B. Mund, "A Time-minimization Dynamic Job Grouping-based Scheduling in Grid Computing", 2012.

[4] Raksha Sharma, Vishnu Kant Soni, Manoj Kumar Mishra, Prachet Bhuyan "A Survey of Job Scheduling and Resource Management in Grid Computing", 2010.

[5] K.Somasundaram, S.Radhakrishnan and M.Gowathyanayagan "In efficient utilization of computing resources using highest response next scheduling in grid", 2007.

[6] K. Somasundaram and S. Radhakrishnan "Task Resource Allocation in Grid using Swift Scheduler", 2009.

[7] Daphne Lopez, S. V. Kasmirraja, "A Dynamic Error Based Fair Scheduling Algorithm For A Computational Grid", 2009.

[8] Sunita Bansal, Bhavik Kothari and Chittaranjan Hota "Dynamic Task-Scheduling in Grid Computing using Prioritized Round Robin Algorithm", 2011.

[9] Mayank Kumar Maheshwari and Abhay Bansal "Process Resource Allocation in Grid Computing using Priority Scheduler", 2012.

[10] Gaurav Sharma, Preeti Bansal, "Min-Min approach for scheduling in grid environment", 2012.