Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Job Scheduling Algorithm for Computational Grid in Grid Computing Environment

Dipti Sharma
Kurukshetra University, Kurukshetra
Haryana, India

Mr. Pradeep Mittal
Kurukshetra University, Kurukshetra
Haryana India

Abstract: Grid Computing allow sharing of resources from heterogeneous and distributed locations. Grid Computing has wide variety of application areas including science, medical and research areas. But there are also some challenges which arise in the environment of grid computing. One of the main challenges is Resource or Job scheduling in the grid. This paper presents such an algorithm which helps in scheduling computational resource to the jobs in efficient way.

Keywords: Computational grid, Deadline, Grid computing, Job Scheduling, Scheduling Algorithm.

I. INTRODUCTION

Grid computing has emerged as a distributed methodology that coordinates the resources that are spread in the heterogeneous distributed environment [1]. Grid is a type of parallel and distributed system that enables the sharing, selection and aggregation of resources distributed across multiple administrative domains based on their availability, capability, performance, cost and user's quality-of-service requirements [2]. A Grid is loosely coupled, geographically distributed and heterogeneous computers in the grid donate printers, application software, disk storage, CPU power etc. Grid uses a middleware layer to communicate with heterogeneous hardware and datasets. Grid can be of many types like data grid, computational grid utility grid, network grid. **Computational** grid has processing power as main computing resource that is shared among the nodes. Grid computing helps in reducing computing cost, turnaround time of each job and it also helps in increasing computing resources and productivity. But there are also some problems related to it like security, complexity and resource scheduling. To realize the full potential of grid computing, grid middleware needs to support various services such as security, uniform access, resource management, job scheduling, application composition, economic computation, and accounting [3]. **Job Scheduling** is choosing the most suitable resource for a job to complete its execution either in terms of waiting time, turnaround time or cost. Scheduler is used to manage the jobs and resources. Scheduler performs two main functions; First scheduler selects the appropriate computational resource for the job and then assigns the resource to the jobs. Job Scheduling is done in order to make the efficient use of resources and for seamless execution of jobs. The main objective of scheduling is to reduce the completion time of an application by properly allocating the jobs to the processors.

II. RELATED WORK

K.Somasundaram, S.Radhakrishnan and M.Gowathyanayagan [5] stated highest response next scheduling algorithm in order to correct some of the weakness in both shortest job 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. Radhakrishnan [6] 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 [7] has described and compared Fair Scheduling algorithm with First Come First Serve (FCFS) and Round Robin (RR) schemes. Sunita Bansal, Bhavik Kothari and Chittaranjan Hota [8] describe an approach to schedule tasks efficiently in a grid environment, without having prior information on workload of incoming tasks. An enhancement to the existing round-robin heuristic by prioritizing tasks eligible for replication is proposed. Mayank Kumar Maheshwari and Abhay Bansal [9] 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 [10] 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. SCHEDULING MODEL

A. *Users:* The user enters the jobs to be executed on processor in computational grid.

B. *Resource Broker:* Users typically do not interact with Grid services directly. Resource Broker is used to discover computing resources with the help of Information System and provide the jobs suitable resource for their computation. Resource broker is used to find the appropriate resource for the jobs, to do so it contacts the grid information server that keeps the status of all the currently available resources in the grid system.

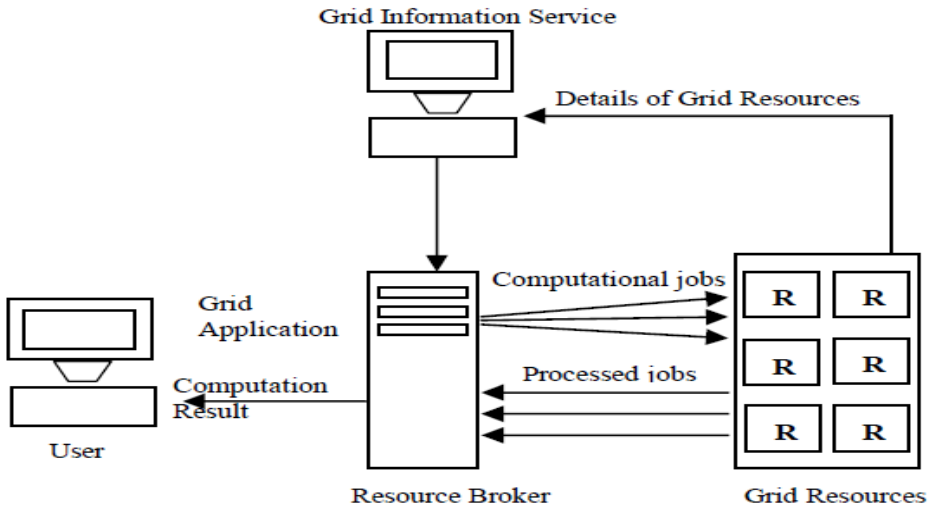

Fig1. Basic Grid Model [4]

C. Grid Information Service:

GIS maintains the information of all resources that is needed by the broker for scheduling the jobs in efficient way. Then resource broker selects the best resource for the job to be executed. After allocation of resource, updating is performed. After successful completion of a job, the information about the status of resource is again updated. This procedure is repeated until all the jobs are assigned to the resources.

IV. PROPOSED WORK

We have implemented a job scheduling algorithm; EDSRTF in C++. In implementation we assume there are six processors and maximum of 10 jobs can be entered at a time. We have considered three nodes Node 1, Node 2, Node 3 and each node consists of 2 processors P1, P2, P3, P4, P5 and P6 respectively.

Earliest Deadline First With Shortest Remaining Time:

Earliest deadline first With Shortest Remaining Time (EDSRTF) is a scheduling algorithm in which scheduling is done according to the deadline of jobs and remaining time. The job with the earliest deadline will get resource first while the job with large value of deadline will have to wait irrespective of their execution time. In EDSRTF scheduling algorithm first execution time along with the deadline of the job is entered. Then according to this scheduling algorithm, first six jobs are directly assigned to the processor P1, P2, P3, P4, P5 and P6 respectively. For the next jobs; processors are assigned according to the deadline of the jobs. If the current job on the processor has large value of deadline than the arriving job, then processor will stop executing the job and will execute the new coming job as its value of deadline is small. But if the current job has small deadline value then processor will keep executing the current job. Then in this case the next processor will be checked for the job and this process will continue until the job gets the processor for its execution. This process is repeated for each job. In the case if no processor satisfied this deadline condition then job will get the processor with shortest remaining time.

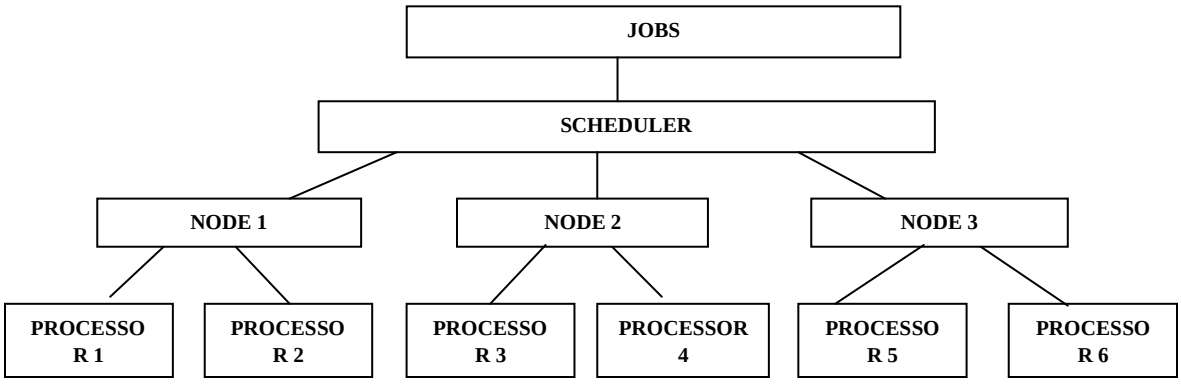

Fig2. Diagram for EDSRTF

Algorithm:

Step1: Enter number of Jobs, their Execution Time and their Deadline.

Step 2: Assign first six jobs directly to processor P1, P2, P3, P4, P5, P6 and set status of processors; SP1=0, SP2=0, SP3=0, SP4=0, SP5=0, SP6=0.

Step3: Repeat following for each job from i=7; to i<=11. (Here we have taken 10 jobs)

i. If((P1==0)&&(d[1]>d[i]))
Assign it to P1 and P1 will first execute this job then after its execution it will execute the previous job and calculate waiting and turnaround time of job and set SP1==1, status of job i =1.

ii. Else If((P2==0) &&(d[2]>d[i]))
Assign it to P2 and P2 will first execute this job then after its execution it will execute the previous job and calculate waiting and turnaround time of job and set SP2==1, status of job i =1.

iii. Else If((P3==0) &&(d[3]>d[i]))
Assign it to P3 and P3 will first execute this job then after its execution it will execute the previous job and calculate waiting and turnaround time of job and set SP3==1, status of job i =1.

iv. Else If((P4==0) &&(d[4]>d[i]))
Assign it to P4 and P4 will first execute this job then after its execution it will execute the previous job and calculate waiting and turnaround time of job and set SP4==1, status of job i =1.

v. Else If((P5==0) &&(d[5]>d[i]))
Assign it to P5 and P5 will first execute this job then after its execution it will execute the previous job and calculate waiting and turnaround time of job and set SP5==1, status of job i =1.

vi. Else If((P6==0) &&(d[1]>d[i]))
Assign it to P6 and P6 will first execute this job then after its execution it will execute the previous job and calculate waiting and turnaround time of job and set SP6==1, status of job i =1.

Step 4: Now check status of every job and if job's status is still = 0, then compare remaining time of each processor and assign the job to the processor with small remaining time. Calculate waiting and turnaround time of job.

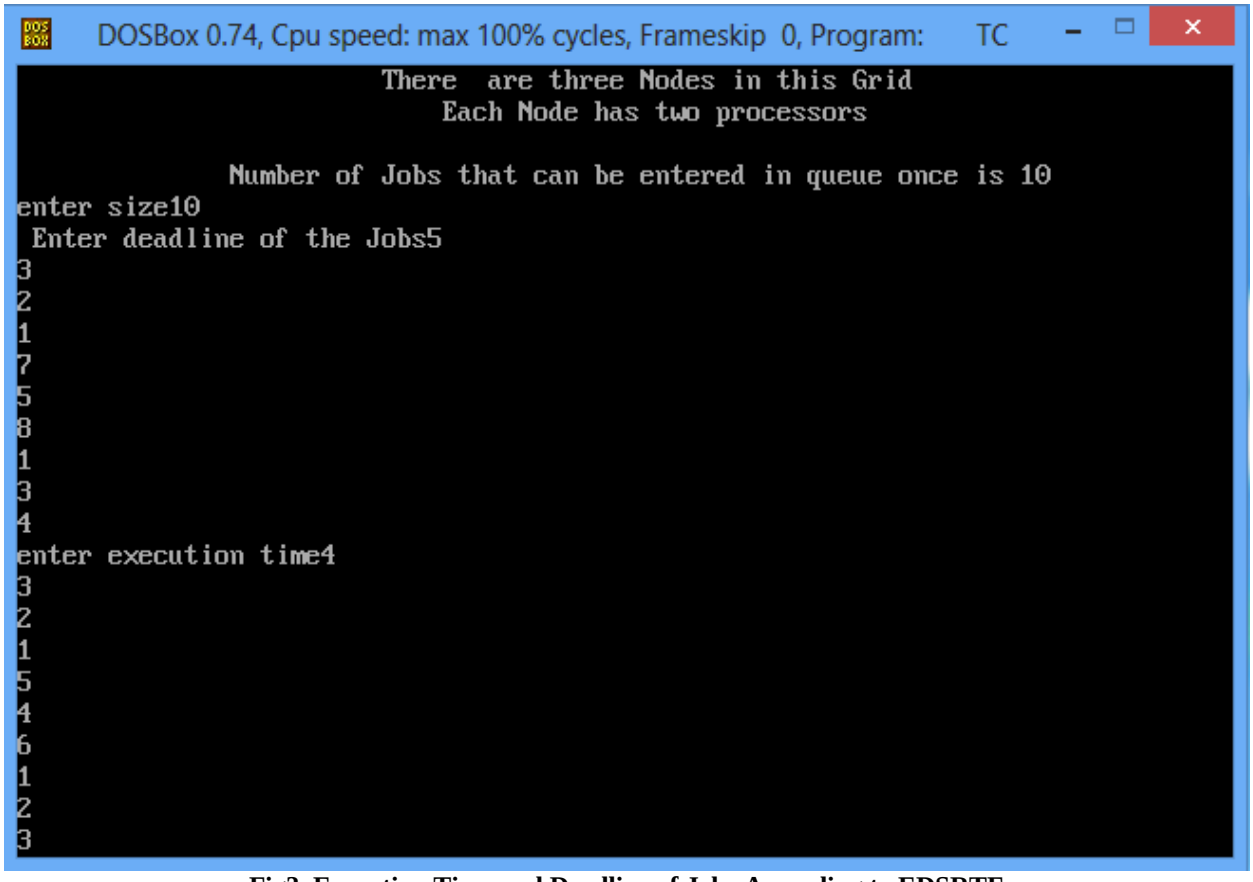

Fig3. Execution Time and Deadline of Jobs According to EDSRTF

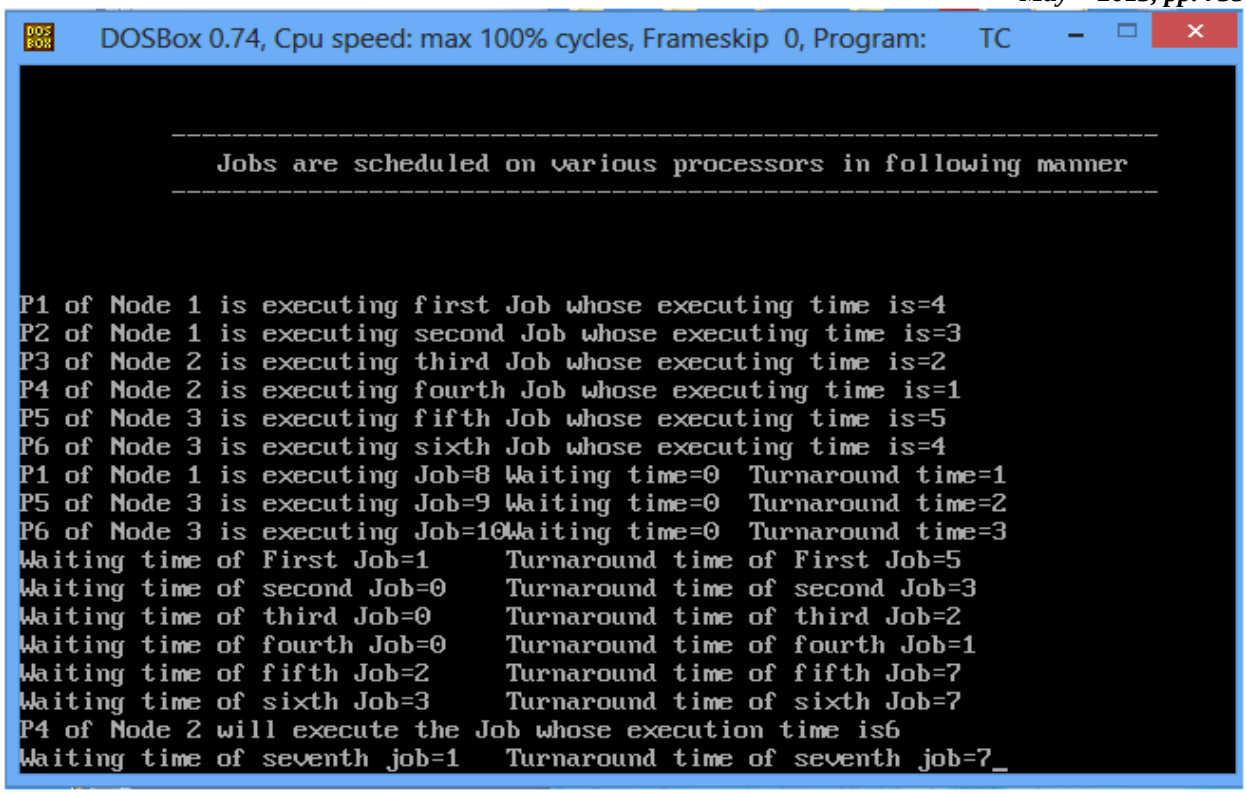

Fig4. Scheduling according to EDSRTF

V. COMPARISON with OTHER SCHEDULING ALGORITHMS

Comparison of the proposed algorithm EDSRTF with two other algorithms FCFS and LJF is also done in this paper.

A. Longest Job First: In longest job first scheduling, the job with large execution time is executed first in comparison to job with shortest execution time. It means a job whose execution time is small will have to wait because of a job which has large execution time than that job.

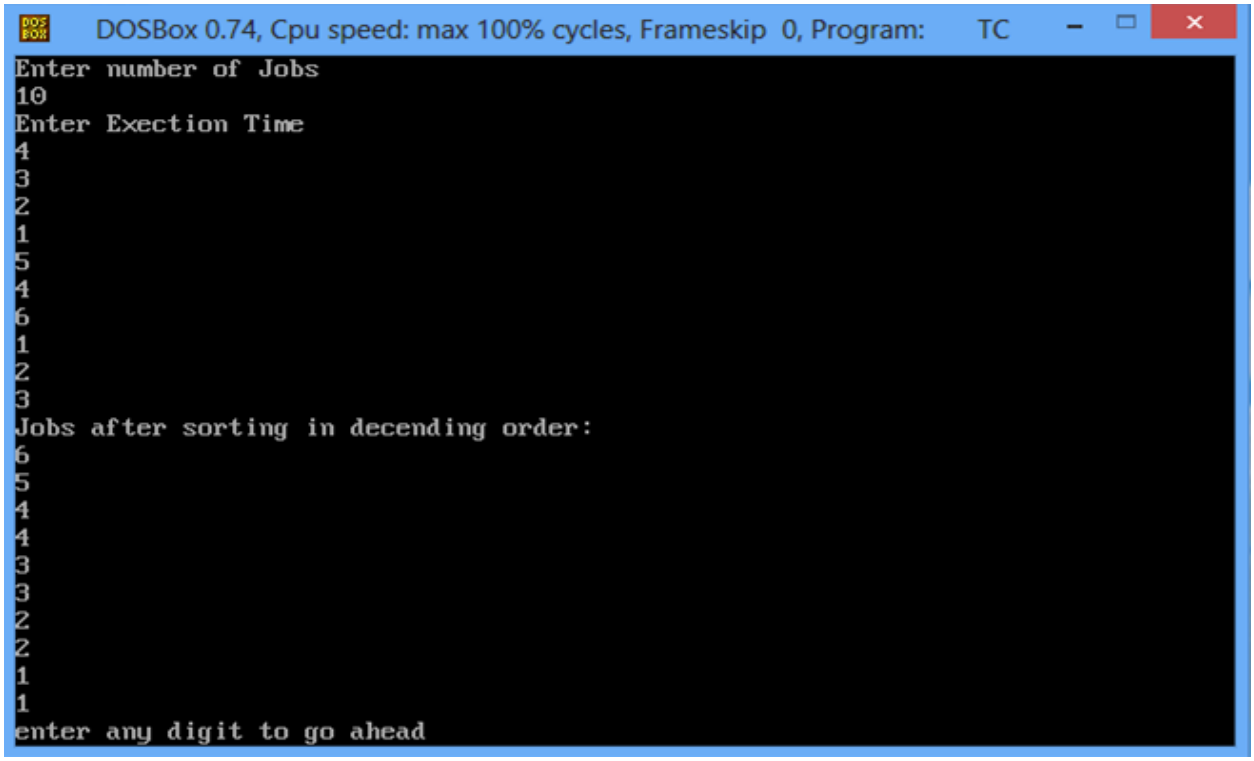

Fig5. Execution Time of Jobs

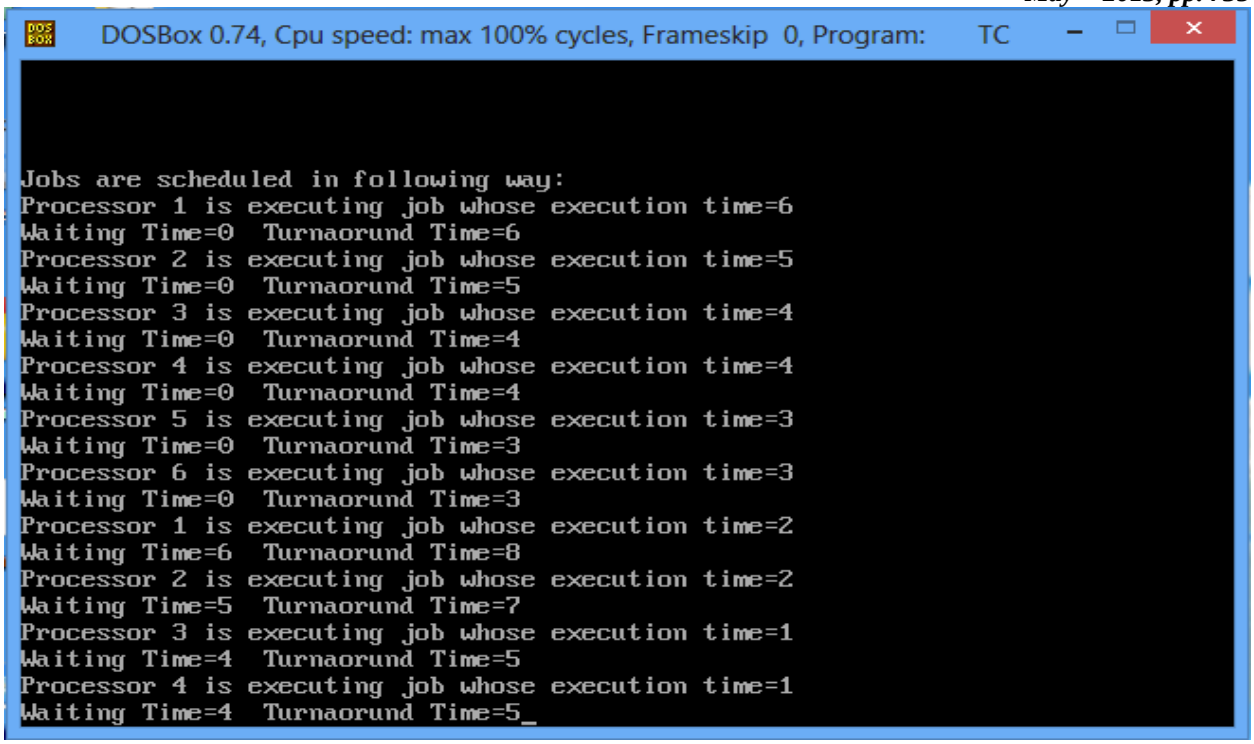

Fig6. **Scheduling according to LJF**

B. First Come-First Serve: First-Come-First-Serve is one of the simplest scheduling algorithms. Jobs get the processors according their arrival. Job which will enter first will get the processor first. It is a non preemptive scheduling algorithm, once a process gets processor, it runs to completion.

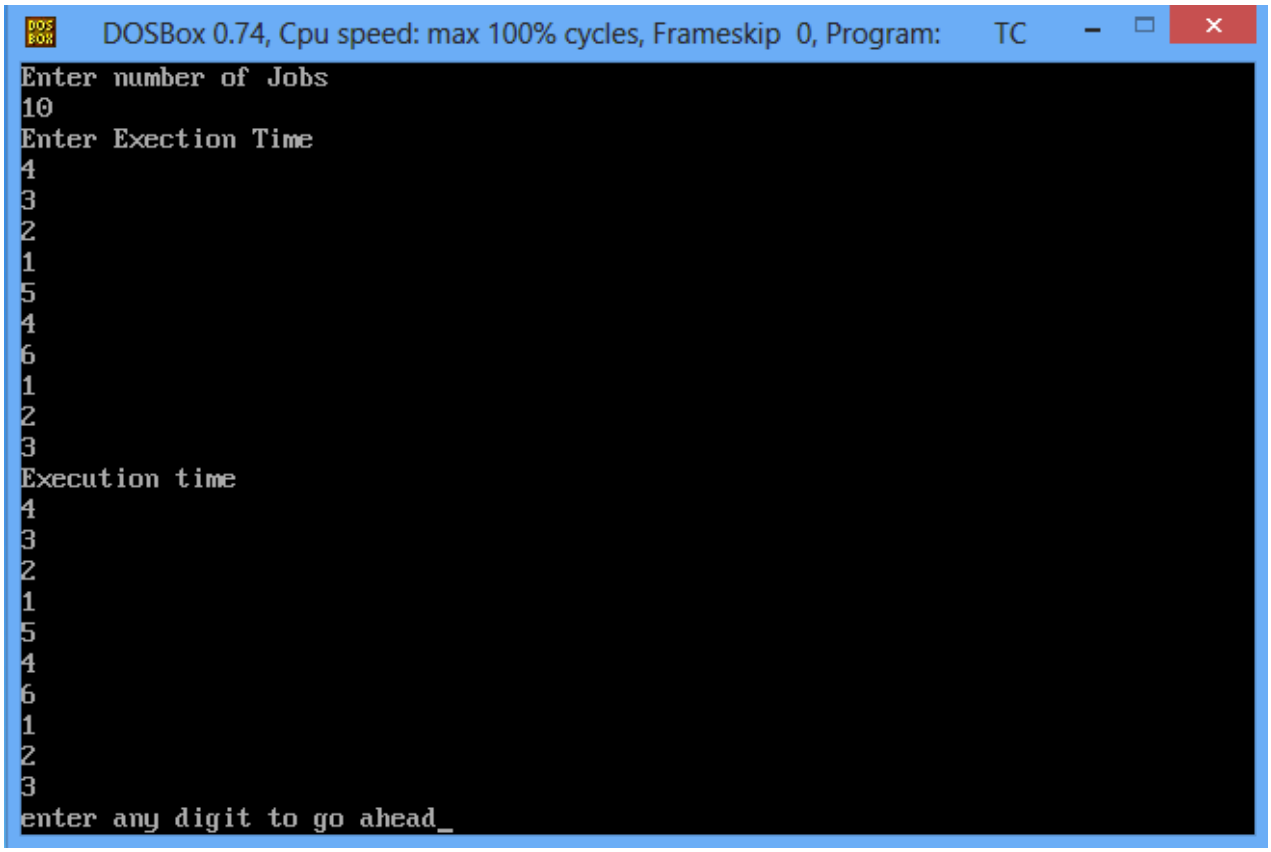

Fig7. Execution Time of Jobs

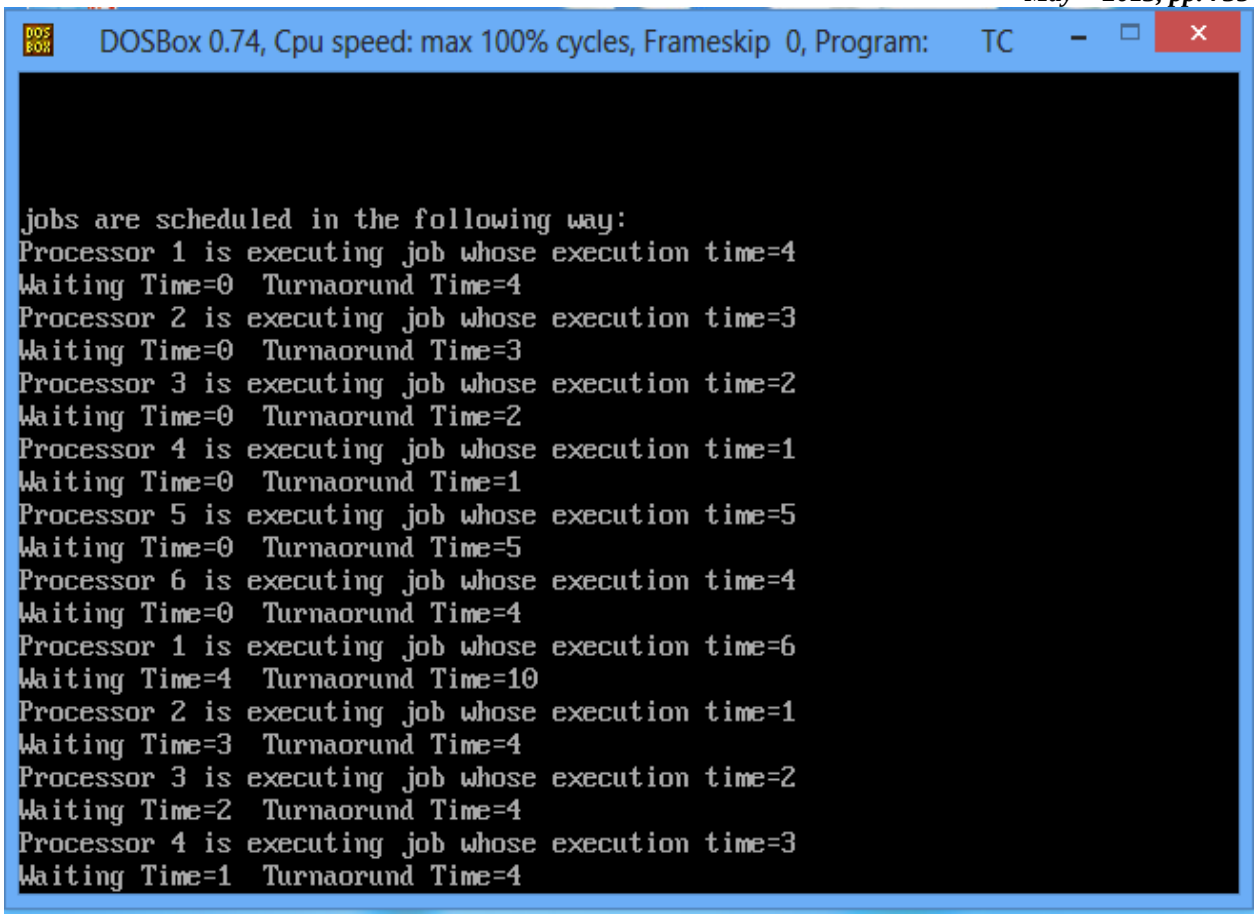

Fig8. Scheduling according to FCFS

Table I. Waiting and Turnaround Time of Jobs using LJF, EDSRTF and FCFS

Jobs	LJF		EDSRTF		FCFS	
	Waiting Time	**Turnaround time**	**Waiting Time**	**Turnaround time**	**Waiting Time**	**Turnaround Time**
1	0	4	1	5	0	4
2	0	3	0	3	0	3
3	6	8	0	2	0	2
4	4	5	0	1	0	1
5	0	5	2	7	0	5
6	0	4	3	7	0	4
7	0	6	1	7	4	10
8	4	5	0	1	3	4
9	5	7	0	2	2	4
10	0	3	0	3	1	4

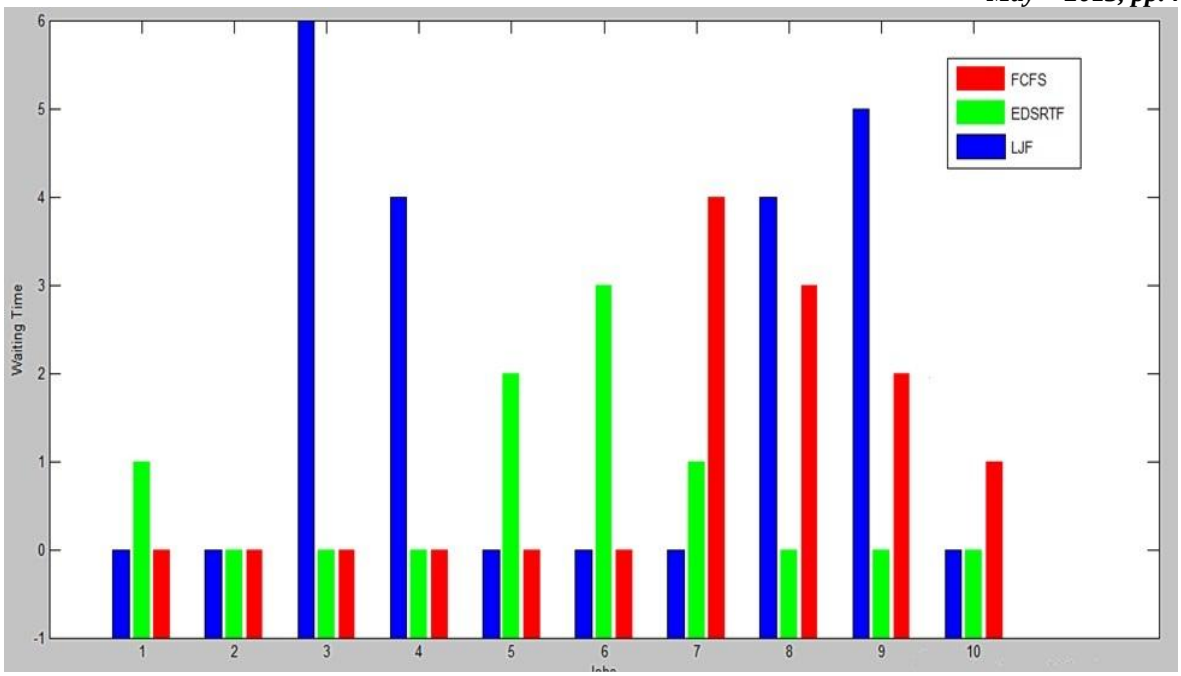

Fig9. Waiting Time Comparison of 10 Jobs using EDSRTF, LJF and FCFS

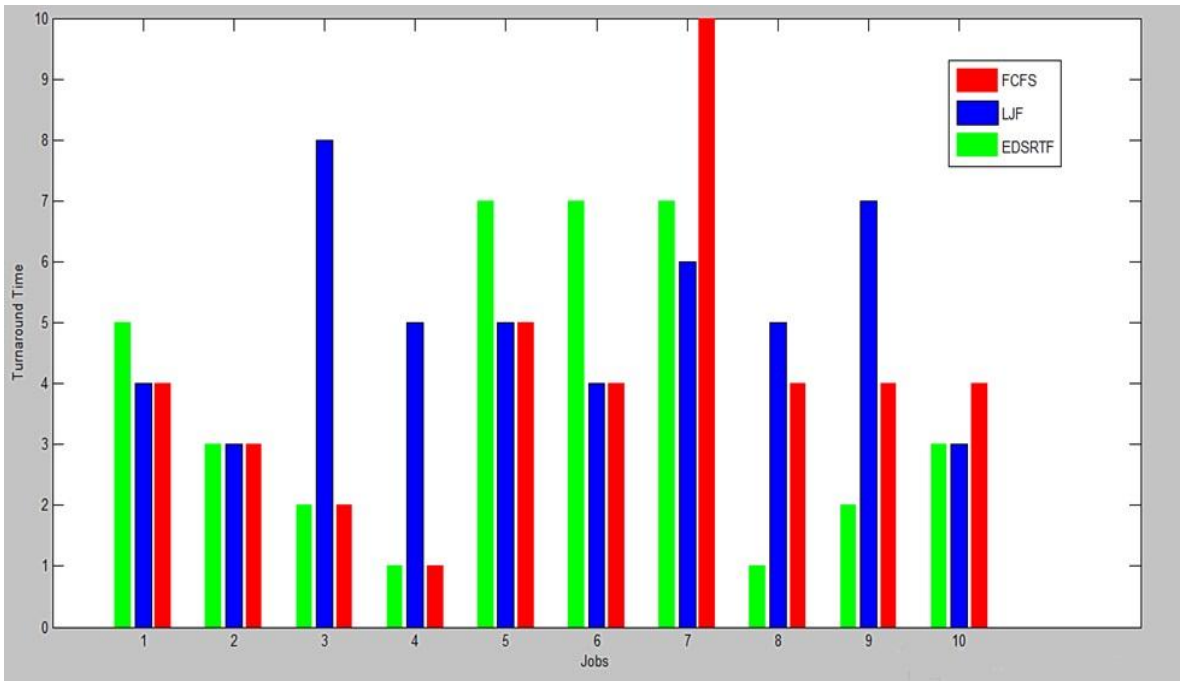

Fig10. Turnaround Time Comparison of 10 Jobs using EDSRTF, LJF and FCFS

Average Waiting Time = $_{i=1}^{n}\sum W_{i/n}$

Average Turnaround Time = $_{i=1}^{n}\sum T_{i/n}$

Table II. Average Waiting and Average Turnaround Time of Jobs using LJF, FCFS and EDSRTF

	LJF	FCFS	EDSRTF
Average Waiting Time	1.9	1.0	0.7
Average Turnaround Time	5.0	4.1	3.8

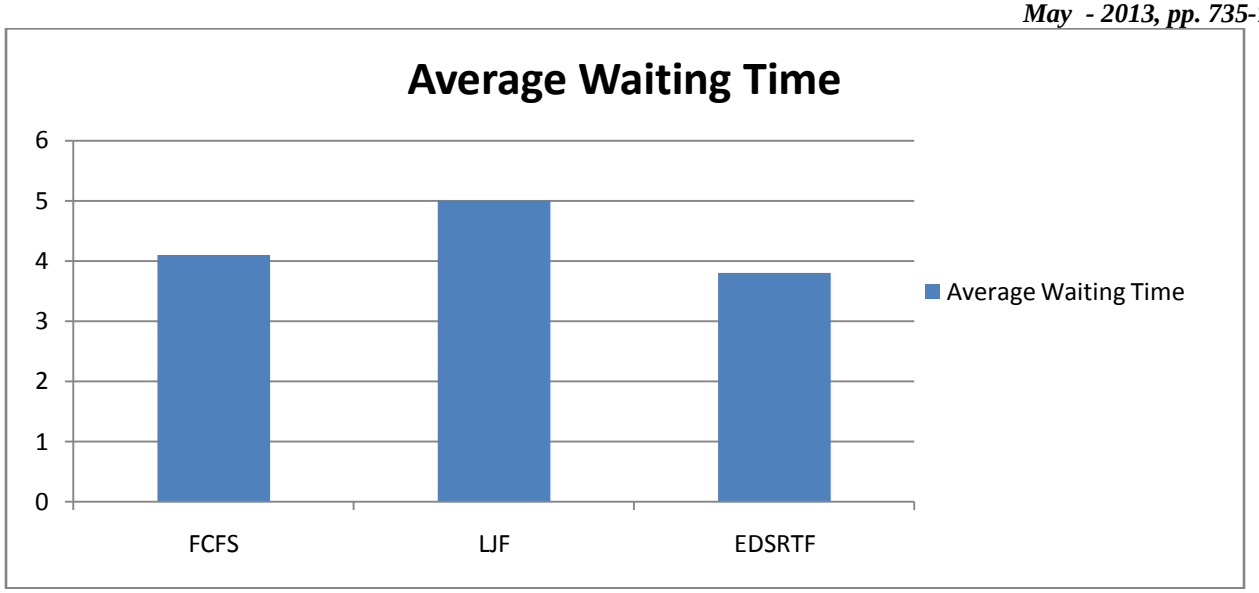

Fig11. Average Waiting Time Comparison of different Scheduling Algorithms

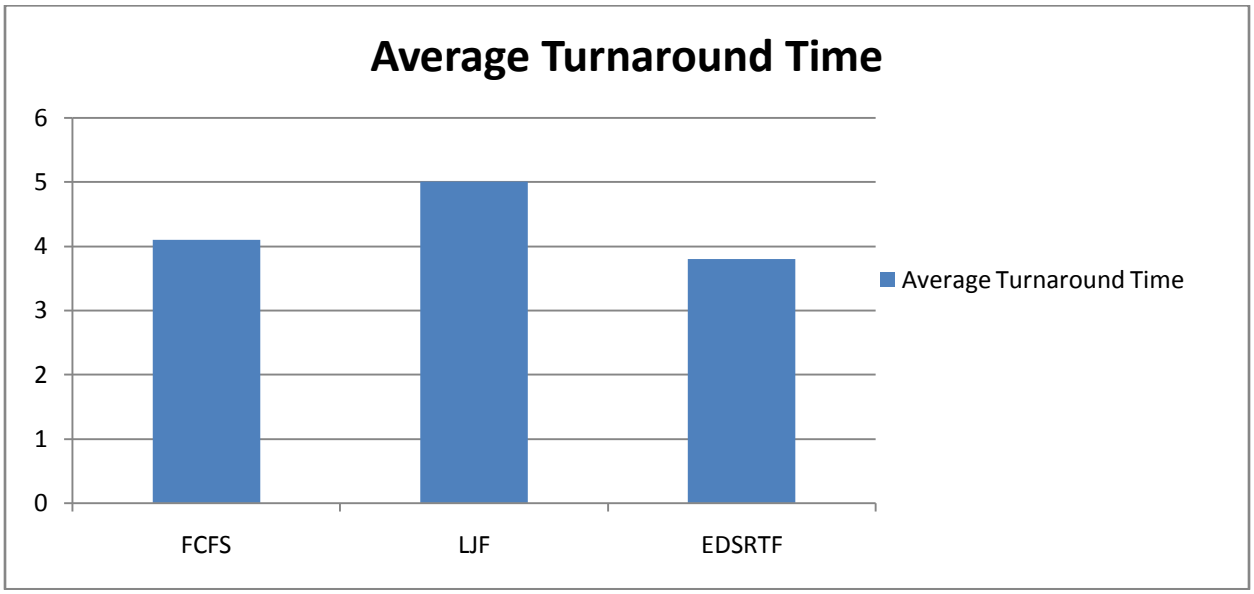

Fig12. Average Turnaround Time Comparison of different Scheduling Algorithms

VI. CONCLUSION And FUTURE WORK

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 EDSRTF and also done its comparison in terms of waiting and turnaround time of jobs with two other algorithms FCFS, LJF. Average waiting time and average turnaround time of jobs is also calculated. Average waiting time and average turnaround time of proposed algorithm is less in comparison to other scheduling algorithms so EDSRTF is best among them. We have considered here number of jobs as fixed 10; in future this number can be made dynamic.

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. Kasmir raja, "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.

