Minimum CDT based Scheduling Algorithm versus FCFS in Grid Environment

Deepti Malhotra, PhD.
Assistant Professor
Department of Computer Science
Central University of Jammu, Jammu

ABSTRACT

To achieve the promising potentials of tremendous distributed resources, effective and efficient scheduling algorithms are fundamentally important. Unfortunately, scheduling algorithms in traditional parallel and distributed systems, which usually run on homogeneous and dedicated resources, e.g. computer clusters, cannot work well in the new circumstances. In this research paper, MJ_CDT_{min} [1] (multiple jobs based on the minimum cumulative departure time) algorithm is compared with the already existing FCFS (First Come First Serve) algorithm in terms of the execution time (in secs).Since there were no results for minimizing the execution time in for existing algorithms. Hence the comparison is done only for the proposed algorithms. This is achieved with the experimental test bed by specifying deadline, while submitting the jobs. Simulation was carried out by different number of jobs varying from 1000 to 10,000. In the experimental testing heterogeneous machines were used and tested for different number of tasks/jobs. During the experiment, the comparison was carried out by considering the six different values for the Service time (ST_k) of the jobs. The main aim of proposed scheduling algorithm is to increase the system efficiency and to satisfy the job requirements from the available resources. The experimental results showed a significant improvement in terms of a smaller makespan time as compared to the already existing FCFS scheduling algorithm.

Keyword

Grid Computing, Job Scheduling, Scheduler, makespan, MCDT, FCFS.

1. INTRODUCTION

A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [2]. It is a shared environment implemented via the deployment of a persistent, standards-based service infrastructure that supports the creation of, and resource sharing within, distributed communities. Resources can be computers, storage space, instruments, software applications, and data, all connected through the Internet. Since multiple applications may require numerous resources which often are not available for them so that in order to allocate resources to input jobs, having a scheduling system is essential. Because of the vastness and separation of resources in the computational grid, scheduling is one of the most important issues in grid environment [3]. Vast investigations have been done in this scope, which have led to theories and practical results [4, 5, and 6]. However new scheduling algorithms have been offered with emergence of grid computing. Objectives of scheduling algorithm are increasing system throughput [6], efficiency, and decreasing job completion time.

Scheduling systems for traditional distributed environments do not work in Grid environments because the two classes of environments are radically distinct. Scheduling in Grid environments is significantly complicated by the heterogeneous and dynamics nature of Grids. Compared to traditional scheduling systems such as clustering computing, Grid scheduling systems have to take into account diverse characteristics of both various Grid applications and various Grid resources. The different performance goals also place great impacts on the design of scheduling systems. To overcome the heterogeneous and dynamic nature of Grids, the information service plays a highly important role. A successful scheduling system should accommodate all these issues.

The job of the grid scheduler is to automatically assign the suitable resources to the independent jobs to maximize the system utilization. It also reduces the average response time of the jobs. The efficient utilization of grid computing resources can improve the overall job-throughput due to load balancing of the tasks between the grid resources. A Grid scheduler is different from local scheduler in that a local scheduler only manages a single site or cluster and usually owns the resource. A Grid scheduler is in charge of resource discovery, Grid scheduling (resource allocation and task scheduling), and job execution management over multiple administrative domains. The jobs will take different execution time on different machines. So, job scheduling in grid environment is a problem to schedule a stream of applications from different users to a set of computing resources to minimize the total completion time. This scheduling requires the matching of different jobs with the machines that satisfy their resource requirement. There are two different goals for job scheduling:

i. Increasing computing performance, its aim is to minimize the execution time of each application that is considered in parallel processing.
ii. Increasing overall throughput, its purpose is to schedule a set of independent tasks in such a way that it increases the processing capacity of the systems for long period of time.

There are relatively a large number of task scheduling algorithms to minimize the total completion time of the tasks in distributed systems [7, 8, 9, 10, 11, and 12]. These algorithms try to minimize the overall completion time of the tasks by finding the most suitable resources to be allocated to the tasks. It should be noticed that minimizing the overall completion time of the tasks does not necessarily result in the minimization of execution time of each individual task.

The rest of the paper is organized as follows. Section 2 presents the background and related work. Section 3 discusses the details of the proposed scheduling algorithm i.e MJ_CDT_{min} (multiple jobs based on the minimum cumulative departure time).Section4 provides the results of the experiment carried out using our own grid simulated environment.Section5 gives the comparison results of MJ_CDT_{min} based scheduling with the commonly used FCFS (first come first serve) algorithm. Finally Section 6 concludes the paper by summarizing our contributions and future works.

2. RELATED WORK

Due to relatively high communication costs in grid environments most of the well known scheduling algorithms are not applicable in large scale distributed systems such as grid environments [7, 13, 14 and 15]. There has been an ongoing attempt to build scheduling algorithms specifically within grid environments. Various algorithms have been proposed which in recent years each one has particular features and capabilities. In this section we review several scheduling algorithms which have been proposed in grid environment. In [16] a scheduling algorithm which is based on HQ-GTSM is presented. This algorithm not only takes into account the input jobs but also considers the resource migration time in deciding on the scheduling. One of the most important features of this algorithm is that it guarantees the grid quality of service. At the present time, job scheduling on grid computing is not only aims to find an optimal resource to improve the overall system performance but also to utilize the existing resources more efficiently.

X. He et al. have presented a new algorithm based on the conventional Min-min algorithm [7].The proposed algorithm which is called QoS guided Min-min, schedules tasks requiring high bandwidth before the others. Therefore, if the bandwidth required by different tasks varies highly, the QoS guided Min-min algorithm provides better results than the Min-min algorithm. Whenever the bandwidth requirement of all of the tasks is almost the same, the QoS guided Min-min algorithm acts similar to the Min-min algorithm.

E. Elmroth et al. have proposed a user oriented algorithm for task scheduling in grid environments, using advanced reservation and resource selection [17]. The algorithm minimizes the total execution time of the individual tasks without considering the total execution time of all of the submitted tasks. Therefore, the overall makespan of the system does not necessarily get small.

F. Dong et al. have proposed a similar algorithm called QoS priority grouping scheduling [18]. This algorithm, considers deadline and acceptation rate of the tasks and the makespan of the wholes system as major factors for task scheduling. In comparison with Min-min and QoS guided Min-min, the QoS priority grouping scheduling algorithm achieves better acceptance rate and completion time for the submitted tasks.

K. Etminani et al. have proposed a new algorithm which uses Max-min and Min-min algorithms [12]. The algorithm determines to select one of these two algorithms, dependent on the standard deviation of the expected completion times of the tasks on each of the resources

B. Yagoubi et al. have offered a model to demonstrate grid architecture and an algorithm to schedule tasks within grid resources [19]. The algorithm tries to distribute the workload of the grid environment amongst the grid resources, fairly. Although, the mechanism used in [19] and other similar strategies which try to create load balancing within grid resources can improve the throughput of the whole grid environment, the total makespan of the system does not decrease, necessarily.

3 MINIMUM CDT BASED SCHEDULING ALGORITHM (MJ_CDT_{min})

In this section, New Grid Job Scheduling algorithm MJ_CDT_{min} (multiple jobs based on the minimum cumulative departure time) is discussed in detail. The proposed algorithm allows multiple job requests to be processed on a single node i.e each node consists of four processors. The MJ_CDT_{min} is based on the rule that the cumulative arrival time of the next job arriving at the processor is compared with the minimum cumulative departure time of the processor. The main aim of proposed scheduling algorithm is to increase the system efficiency and to satisfy the job requirements from the available resources.

3.1 Assumptions

Let us consider the arrival time and service time of K jobs. Let these jobs be marked as $J_1, J_2, J_3 \ldots\ldots, J_k$.Let the interarrival time AT_k denotes the time gap between the arrivals of the $(k-1)^{th}$ job and the K^{th} job into the system. These times will be generated randomly. Similarly, let ST_K denotes the service time of the K^{th} Job. The service times are also generated randomly.

- ✓ We will use CAT_k to denote the cumulative arrival time of the $K^{th}\ job$.

$$CAT_k = CAT_{k-1} + AT_k$$
$$CAT_1 = AT_1 = 0$$

Fig1: Cumulative Arrival time Representation

We will assume that initially there is no queue, and all the processors are free.

- ✓ CDT_k is the time elapsed at the departure of the $K^{th}\ job$ from processor.

$$CDT_k = CAT_k + ST_k + WT_k$$

Fig2: Cumulative Departure time Representation

- ✓ Idle time is the amount of the time the processor spends while waiting for the Job no *K* to arrive.

$$IDT_k = CAT_k - MNDT \ \ (CAT_k > MNDT)$$

Fig3: Idle time Representation

- ✓ Wait time denote the waiting time of the k^{th} job in the queue.

$$WT_k = MNDT\text{-}CAT_k (MNDT > CAT_k)$$

Fig4: Wait time Representation

- ✓ QL_k (Queue Length) = number of jobs waiting for processing scheduling in the queue. If the wait time

for the job comes then the queue length is incremented by 1.

3.2 Algorithm (pseudo code)

Step1: cList= list of all individual requests by validating the client specification(s);
Step2: nodeList =list of all nodes. Each Grid node comprises a number of computational resources (processors, denoted by:
$P_{ni}\left\{\sum_{n=1}^{x} n = \text{node no.} \middle| \sum_{i=1}^{4} i = \text{processors no. in node}\right\}$);
Step3: For each job do the following steps;
Step4: Filter out the resources that do not fulfill the job requirements. Contact GRD (GridResource Database) to obtain a list of available resources;
Step5: $k = \text{no of jobs}$
Step6: cat=0, nd=1, et=0;
Step7: $for(i = 1; i <= 100; i++)$
{
$q[i] = 0, x[i] = 0, wt[i] = 0, cdt[i] = 0, idt[i] = 0;$
}
Step8: $if(k == 0)$
{
$go\ to\ step\ 18$;
}
/ checking Jobs*/*
Step9: $for(j = 1; j <= k; j++)$
{
*/*randomly take the values of arrival time and store them in array*/*
$at[j] = random(100);$
$cat = cat + at[j];$
$st[j] = random(100);$ */*checking..... Processor with Min CDT (cumulative departure time)*/*
Step10: $minCDT = cdt[1], nd = 1;$
Step11: $for(i = 1; i <= 4; i++)$
{
$If\ (cdt[i] < minCDT);$
{
$minCDT = cdt[i];$
$nd = i;$
} */*End If */*
} */*End For */*
Step12: $x[nd] = cdt[nd];$ */*Processor with Min CDT*/*
Step13: $cdt[nd] = cat + st[j];$
/ comparing the arrival time of the next job with the min CDT store in step 14*/*

Step14: $If(cat >= x[nd])$
{
$idt[nd] = cat - x[nd];$
/ idle time for the processor*/*
$x[nd] = cdt[nd];$
$wt[nd] = 0;$ ** wait time for the job*/*
$cdt[nd] = cdt[nd] + wt[nd];$
} */*End If */*
$Else$
{
$idt[nd] = 0;$
/ idle time for the processor*/*
$wt[nd] = x[nd] - cat;$
$q[nd] = q[nd] + 1;$
/ increment the queue length*/*
$cdt[nd] = cdt[nd] + wt[nd];$
} */*End Else */*
Step15: $Processor\ Allocated = nd;$
/ calculate the total Execution time for all the jobs*/*
Step16: $if(j == 1)$
{
$et = cdt[1];$ */* execution time*/*
}
$if(cdt[nd] > et)$
{
$et = cdt[nd];$
}
} */*End For */*
Step17:Total Execution Time = et
Step18:exit

3.3 Algorithm Description

In step 1 of Algorithm given in section 3.3, the user's request is processed and split into individual job requests. Step 2, consists of list of all nodes available in the grid. The actual scheduling process starts from step 3 that is repeated for each job request. In step 4, the scheduler discovers the available resources by contacting GRD (GridResource Database).Here resources are evaluated according to the requirements in the job request and only the appropriate resources are kept for further processing.Step5 gives the number of job request.Step8 randomly take the values for the arrival times (at) of the jobs by calling the random function and store them in an array ($at[j]$).Step9 calculate the cumulative arrival time(cat) for the jobs. Step10 take the random values for the services time(st) of the jobs and store them in an array($st[j]$).Step 11, 12, 13 check the processors with minimum cumulative departure time. Then the Processor with minimum value of cumulative departure time (min CDT) is stored in the Step14.The processor with min CDT is selected for the allocation of the next job arriving. Step 15 calculates the cumulative departure time.Step16 compare the cumulative arrival time of the next job arriving with the min CDT store in step 14. If the value of cumulative arrival time is greater than the min CDT then the idle time for the processor comes i.e. $idt[nd]$ is the amount of idle time processor spends while waiting for the job to arrive. Otherwise wait time for the job comes i.e. $wt[nd]$ is the amount of time which job has to wait in the queue to be serviced by the processor. As a result of which queue length is incremented by one. In Step17 job is assigned to the processor.Step18 calculates the total execution time for executing the jobs.

3.4 Flowchart

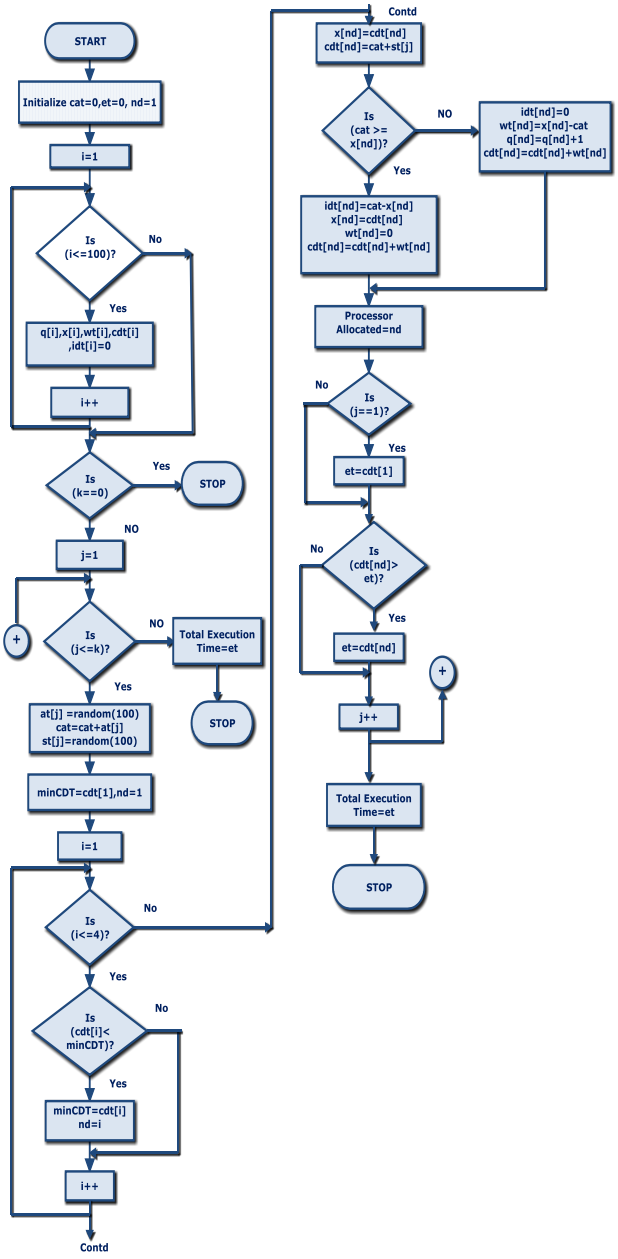

Fig5: Flowchart of MJ_CDT_{min} based scheduling algorithm

4 EXPERIMENTAL RESULTS FOR (MJ_CDT_{min})

Experiments have been carried out by using the simulated Grid computing environment by using, multiple jobs arriving at single node. In the simulation of the experiment, the arrival time and service time of jobs are generated randomly in the range of {0,100}. Figure6 and Figure7 shows the frequency distribution plot of arrival time (AT_k) and service time (ST_k) of the jobs respectively. For the experiments, all the machines are assumed to have four processing element only. Results have been obtained by scheduling independent sets of jobs generated randomly. First time the simulation has been carried out by varying the number of jobs from 1 to 1000.

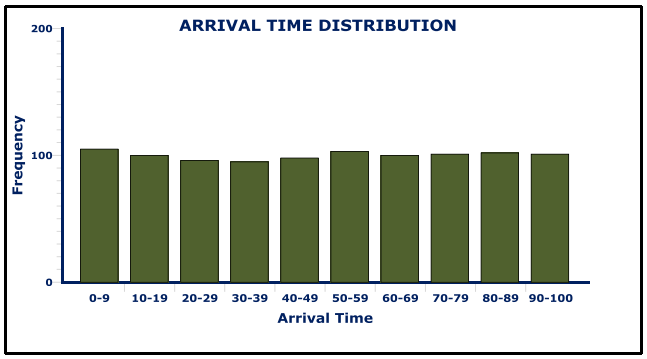

Fig6: Distribution of AT_k

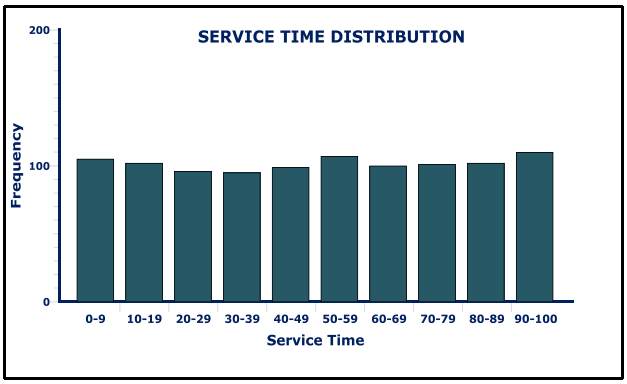

Fig7: Distribution of ST_k

Table 1 gives the results of the experiment carried out for MJ_CDT_{min} based scheduling in Grid simulated environment. Graph in Figure 8 gives the experiment results of the MJ_CDT_{min} based scheduling.

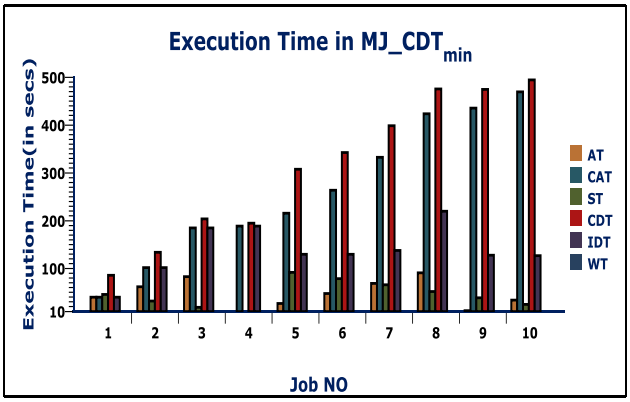

Fig8: MJ_CDTmin based Scheduling

Table 1: Simulation Test Runs for MJ_CDT

Job No	AT (sec)	CAT (sec)	ST (sec)	min CDT Processor		CDT (sec)	IDT (sec)	WT (sec)	QL	PA
				Processor no	MCDT (sec)					
1	40	40	46	1	0	86	40	0	0	1
2	62	102	32	2	0	134	102	0	0	2
3	83	185	19	3	0	204	185	0	0	3
4	04	189	6	4	0	195	189	0	0	4
5	27	216	92	1	86	308	130	0	0	1
6	48	264	79	2	134	343	130	0	0	2
7	69	333	66	4	195	399	138	0	0	4
8	91	424	52	3	204	476	220	0	0	3
9	12	436	39	1	308	475	128	0	0	1
10	34	470	25	2	343	495	127	0	0	2

5 COMPARISON BETWEEN MJ_CDT$_{min}$ AND EXISTING ALGORITHM (FCFS)

Next experiments were carried on to compare proposed MJ_CDT$_{min}$ algorithm with the already existing FCFS (First Come First Serve) algorithm in terms of the execution time (in secs). Since there were no results for minimizing the execution time in for existing algorithms. Hence the comparison is done only for the proposed algorithms. This is achieved with the experimental test bed by specifying deadline, while submitting the jobs. Simulation was carried out by different number of jobs varying from 1000 to 10,000. In the experimental testing heterogeneous machines were used and tested for different number of tasks/jobs. During the experiment, the comparison was carried out by considering the six different values for the Service time (ST_k) of the jobs which are as given below:-

5.1 i) Simulation with $ST_k = 50$

First time, the two scheduling algorithms were compared by taking ST_k ,of integer point number generated randomly in the range{0,50} and the different simulated results were generated as shown in Table 2. ST_k denote the service time of k^{th} job.

Table2: Execution time (in secs) for MJ_CDT$_{min}$ and FCFS with (ST_k {0, 50})

No. of Jobs	MJ_CDT$_{min}$	FCFS
1000	13575	13576
2000	13608	13611
3000	13768	13770
4000	13774	13779
5000	14004	14008
6000	13981	13987
7000	14060	14063
8000	14083	14087
9000	14091	14099
10000	14112	14117

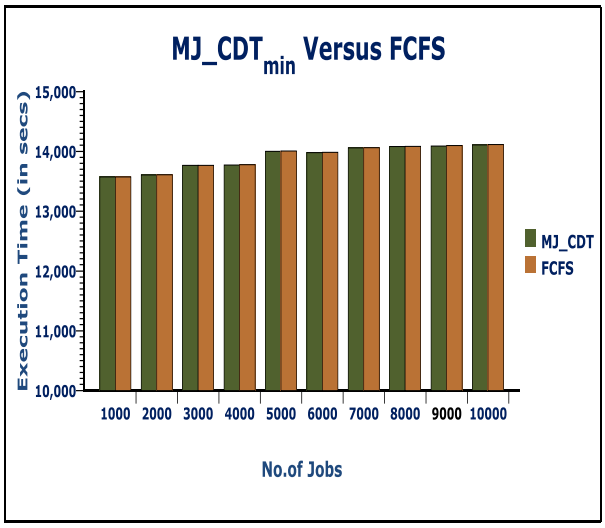

Fig 9: Comparison between MJ_CDT$_{min}$ and FCFS (ST_k {0, 50})

Graph in Figure 9 depicts the comparison among proposed MJ_CDT$_{min}$ scheduling algorithm and the existing FCFS algorithm in terms of execution time (in secs) with ST_k in the range{0,50}.

ii) Simulation with $ST_k = 100$

Next time the two scheduling algorithms were compared by taking ST_k , of integer point number generated randomly in the range{0,100} and the different simulated results were generated as shown in Table 3 and Figure 10.It has been analyzed from the graph plotted in Figure10 that as the service time increases from 50 to 100 the difference between the proposed MJ_CDT$_{min}$ scheduling algorithm and the existing FCFS algorithm in terms of execution time (in secs) also increases.

Table3: Execution time (in secs) for MJ_CDT_{min} and FCFS with ($ST_k \{0, 100\}$)

No. of Jobs	MJ_CDT_{min}	FCFS
1000	13701	13727
2000	13718	13770
3000	14000	14054
4000	14441	14498
5000	14578	14635
6000	14741	14803
7000	14511	14571
8000	14508	14572
9000	14776	14841
10000	15046	15113

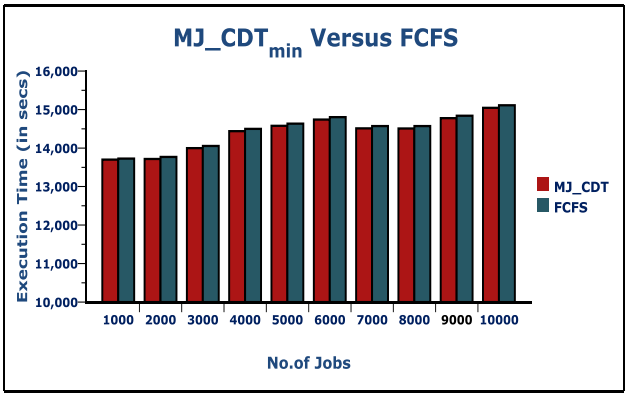

Fig 10: Comparison between MJ_CDT_{min} and FCFS ($ST_k \{0, 100\}$)

iii) Simulation with $ST_k = 150$

In the third time, value of service time is changed from 100 to 150. Table 4 and Figure 11 shows the different results generated during the experiment with $ST_k = 150$

Table 4: Execution time (in secs) for MJ_CDT_{min} and FCFS with ($ST_k \{0, 150\}$)

No. of Jobs	MJ_CDT_{min}	FCFS
1000	13686	13713
2000	13911	13968
3000	13780	13836
4000	14190	14253
5000	14041	14102
6000	14872	14943
7000	14744	14815
8000	14402	14472
9000	15375	15452
10000	15645	15724

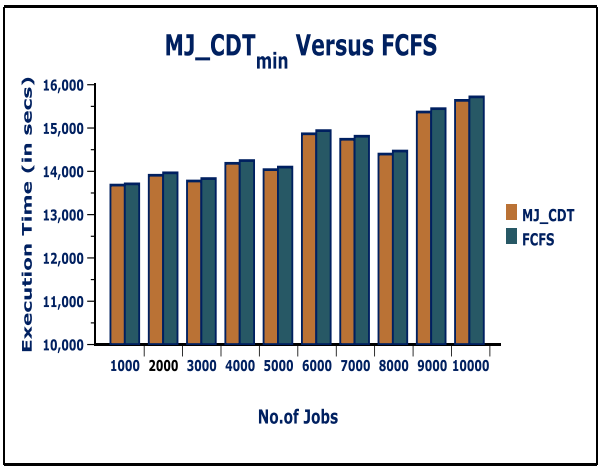

Fig11: Comparison between MJ_CDT_{min} and FCFS ($ST_k \{0, 150\}$)

iv) Simulation with $ST_k = 200$

Table 5: Execution time (in secs) for MJ_CDT_{min} and FCFS with ($ST_k \{0, 200\}$)

No. of Jobs	MJ_CDT_{min}	FCFS
1000	13832	13862
2000	13847	13907
3000	13820	13878
4000	14111	14176
5000	14326	14390
6000	14434	14507
7000	15009	15085
8000	15342	15416
9000	15404	15485
10000	15524	15606

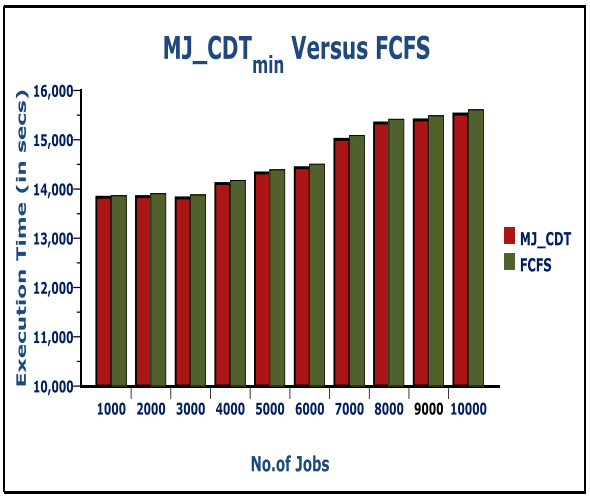

Fig12: Comparison between MJ_CDT_{min} and FCFS ($ST_k \{0, 200\}$)

Table 5 and Figure12 shows the experiment results generated during the simulation with the value of $ST_k = 200$.

v) Simulation with $ST_k = 250$

Table 6: Execution time (in secs) for MJ_CDT_{min} and FCFS with ($ST_k \{0, 250\}$)

No. of Jobs	MJ_CDT_{min}	FCFS
1000	14069	14109
2000	14205	14275
3000	14136	14201
4000	15112	15190
5000	15776	15852
6000	15203	15292
7000	15212	15303
8000	15271	15369
9000	15559	15670
10000	15640	15765

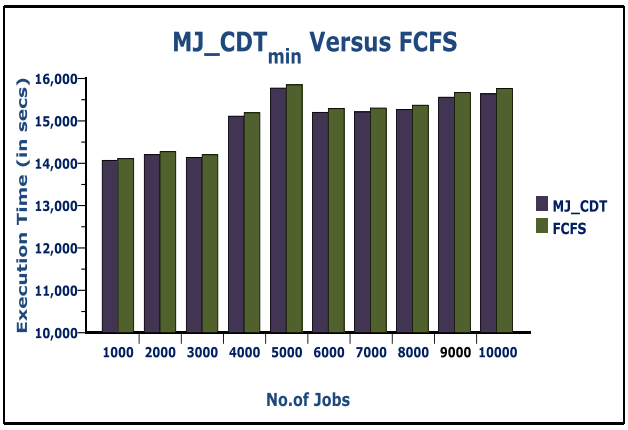

Fig13: Comparison between MJ_CDT_{min} and FCFS ($ST_k \{0, 250\}$)

Graph in Figure13 shows that as the value of service time increases, the performance of proposed MJ_CDT_{min} based schedule also increase as compared to the already existing FCFS scheduling algorithm.

vi) Simulation with $ST_k = 300$

Table 7: Execution time (in secs) for MJ_CDT_{min} and FCFS with ($ST_k \{0, 300\}$)

No. of Jobs	MJ_CDT_{min}	FCFS
1000	14299	14354
2000	14380	14461
3000	14997	15075
4000	15069	15168
5000	15915	16002
6000	15903	16015
7000	15233	15354
8000	16059	16189
9000	16432	16577
10000	16534	16691

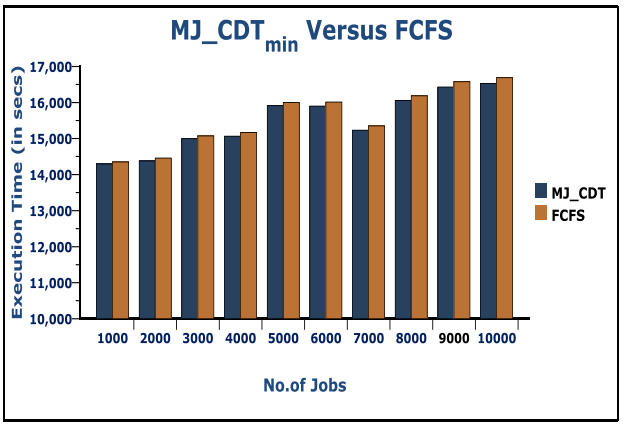

Fig14: Comparison between MJ_CDT_{min} and FCFS ($ST_k \{0, 300\}$)

Graphs in Figure 9- 14 depicts that the Execution time for the given number of jobs is shorter in case of proposed MJ_CDT_{min} based scheduling algorithm as compared to the already existing FCFS scheduling algorithm.

5.2 Performance Analysis of MJ_CDT_{min} and FCFS at Different Values of ST_K

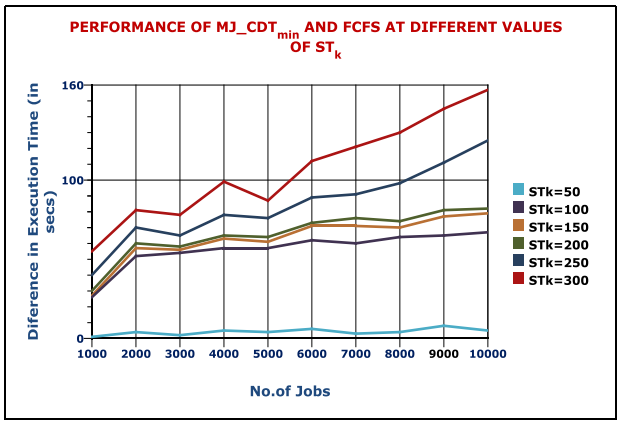

Figure15: Performance of MJ_CDT_{min} and FCFS with varying ST_K

It is clear from the graphs shown in Figure 15 that for the same number of jobs, the difference between the execution time in the proposed minimum cumulative departure time based scheduling algorithm (MJ_CDT_{min}) algorithm and existing FCFS algorithm always goes on increasing as the value of service time (ST_K) increases. Graphs in the Figure 15 also depict that as the service time goes on increasing, the proposed algorithm works better than existing scheduling algorithm. Thus, (MJ_CDT_{min}) shows a noticeable increase in performance than existing scheduling algorithm

6. CONCLUSION AND FUTURE WORK

Resource sharing in grid environment is an inevitable task and the scheduling concept is one of the most important issues in grid computing. In this paper a job scheduling algorithm in grid environment have been presented in order to enhance the average Makespan of input jobs.This newly proposed scheduling algorithm achieves high efficiency in the Grid computing. A simulation system was developed to test the minimum cumulative departure time based scheduling algorithm in a simulated Grid environment. We used the makespan/Execution time of batch jobs as the comparison criteria. This research paper also provides the experimental details of the comparison made between the results of MJ_CDT_{min} (minimum cumulative departure time) based scheduling algorithm and the already existing FCFS scheduling algorithm in the Grid environment. Results show that proposed meta-heuristic based algorithms perform better than the existing algorithms for Grid scheduling. The main aim of these implementations is to evaluate and validate the proposed algorithms against a benchmark to demonstrate their usability. As memory is an important resource, In future research can be done considering others factors such as memory as the resource requirement in task scheduling algorithm.

7. REFERENCES

[1] Deepti Malhotra, Devanand and Anik Gupta.2012. Simulation of MJ_CDT_{min} Based Scheduling Algorithm in Grid Environment,International Journal of Computer Applications (0975 – 8887) Vol 42– No.11,pp.24-29 March 2012.

[2] I. Foster and C. Kesselman (editors).1999. The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann Publishers, USA.

[3] Rajkummar Buyya.2002.Economic-based Distributed Resource Management and Scheduling for grid computing.PhD thesis, Monash university, Melborn, Australia.

[4] K. Al-Saqabi, S. Sarwar, and K. Saleh.1997. Distributed gang scheduling in networks of heterogeneous workstations, Computer Communications Journal, pp.338-348.

[5] Maheswaran M, Ali S, Siegel H J, et al.1999.Dynamic mapping of a class of independent tasks on to heterogeneous computing systems. In the 8th IEEE Heterogeneous Computing Workshop (HCW '99),San Juan, Puerto Rico,(Apr. 1999), pp.30-44.

[6] XiaoShan He, XianHe Sun, and Gregor von Laszewski.2003.QoS Guided Min-Min Heuristic for Grid Task Scheduling, Computer Science and Technology, 18(4):442-451.

[7] X. He, X-He Sun, and G. V. Laszewski.2003.QoS Guided Min-min Heuristic for Grid Task Scheduling, Journal of Computer Science and Technology, Vol. 18, pp. 442-451.

[8] M. Maheswaran, Sh. Ali, H. Jay Siegel, D. Hensgen, and R. F. Freund.1999. Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems, Journal of Parallel and Distributed Computing, Vol. 59, pp. 107-13.

[9] T. D. Braun, H. Jay Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao.2001.A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems,Journal of Parallel and Distributed Computing, Vol. 61, pp. 810-837.

[10] F. Dong, J. Luo, L. Gao, and L. Ge.2006.A Grid Task Scheduling Algorithm Based on QoS Priority Grouping," In the Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06), IEEE.

[11] E. Ullah Munir, J. Li, and Sh. Shi.2007. QoS Sufferage Heuristic for Independent Task Scheduling in Grid. Information Technology Journal, 6 (8): 1166-1170.

[12] K. Etminani, and M. Naghibzadeh.2007.A Min-min Max-min Selective Algorithm for Grid Task Scheduling,The Third IEEE/IFIP International Conference on Internet, Uzbekistan.

[13] Deepti Malhotra.2013. SCH_ACR and SCH_LD Based Job Scheduling Algorithm in Grid Environment, International Journal of Computer Applications (0975 – 8887) Vol 64– No.13, pp.35-41 February 2013

[14] B.T. Benjamin Khoo, B. Veeravalli, T. Hung, and C.W. Simon See.2007.A multi-dimensional scheduling scheme in a Grid computing environment," Journal of Parallel and Distributed Computing, Vol. 67, pp. 659-673.

[15] B. Yagoubi, and Y. Slimani.2007.Task Load Balancing Strategy for Grid Computing, Journal of Computer Science, Vol. 3, No. 3, pp. 186-194.

[16] Huyn zhang, chanle wu, Q.xiong, and L.Wu,G. Ye. 2006. Research on an Effective Mechanism of Task Scheduling in Grid Environment. In IEEE, Fifth International

Conference on Grid and Cooperative Computing (GCC'06).

[17] E. Elmroth, and J. Tordsson.2008.Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions, Journal of Future Generation Computer Systems, Vol. 24, pp. 585-593.

[18] F. Dong, J. Luo, L. Gao, and L. Ge.2006.A Grid Task Scheduling Algorithm Based on QoS Priority Grouping, In the Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06), IEEE.

[19] B. Yagoubi, and Y. Slimani.2007.Task Load Balancing Strategy for Grid Computing, Journal of Computer Science, Vol. 3, No. 3, pp. 186-194.