Journal of Theoretical and Applied Information Technology

www.jatit.org

A DYNAMIC ERROR BASED FAIR SCHEDULING ALGORITHM FOR A COMPUTATIONAL GRID

[1] DAPHNE LOPEZ, [2] S. V. KASMIR RAJA

[1]School of Computing Sciences, VIT University, Vellore, India 632006

[2] Dean Research, SRM University, Kattankulathur, Chennai, India 603203

E-mail: daphnelopez@vit.ac.in , svkr@yahoo.com

ABSTRACT

Grid Computing has emerged as an important new field focusing on resource sharing. One of the most challenging issues in Grid Computing is efficient scheduling of tasks. In this paper, we propose a new algorithm for fair scheduling, and we compare it to other scheduling schemes such as the First Come First Served and the Round Robin schemes for a computational grid. It aims at addressing the fairness issue by reducing the service time error..The algorithm assigns to each task enough computational power to complete it within its deadline. The resources that each user gets are proportional to the user's weight or a share. The weight or share of a user may be defined as the user's contribution to the infrastructure or the price he is willing to pay for services. Scheduling of tasks is based on an error called the Service time error which fairness among users. Fairness is defined as the proportional allocation of resources to tasks as per their demand. Simulated results and comparisons with the conventional scheduling schemes such as the FCFS and Round Robin are presented.

Keywords: *Computational Grid, Scheduling, Fairness, Proportional Allocation, Service Time Error, Shares*

1. INTRODUCTION

"Grid" computing has emerged as an important new field, distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation.[11],[12]. The constant growth of communications, in terms of quality and availability, is increasing the interest on grid computing paradigm [28], by which computing resources geographically distributed can be logically coupled together working as a computational unit. Various types of Grids have been developed to support these applications and are categorized as Computational Grids, Data Grids and Service Grids. Computational Grid (CG) represents a new computational framework whose efficient use requires schedulers that allocate user's tasks to the grid resources in an acceptable amount of time.

An efficient use of distributed resources is highly dependent on the resource allocation by grid schedulers, where user requirements and job characteristics must be also considered. Moreover, due to the changeability of a CG, machines and jobs to be scheduled may vary over time, and therefore, any grid scheduler must generate optimal schedules at a minimal amount of time in order to rapidly adapt itself to the changes of the grid. Major issues that can be easily handled in conventional computing environments become seriously challenging problems in grids mainly because a grid consists of multiple administrative domains [3]. Two very crucial issues among them are security and scheduling [8]. They have been investigated and researched over time.

The demand for scheduling is to achieve high performance computing [5]. The motivation of this paper is to develop a good scheduling algorithm that can perform effectively and efficiently in terms of minimizing the error to achieve fairness and reduce the cost and time. This paper proposes a fair scheduling algorithm based on the service time error [16]. Fair Share is a widely used queueing algorithm for prioritorizing jobs on the basis of a "share" [27]. The first part explains the algorithm and secondly the simulation of the experiment with GridSim toolkit is presented. The simulator defines the workload of resources, the arrival time of independent jobs, length of each job and other parameters. Finally we compare the performance with FCFS and Round Robin.

2. RELATED WORK

In the past, researchers have proposed various scheduling algorithms in a grid environment which is a complex one. The heuristic algorithms proposed for job scheduling in [4], [27] and [9] rely on static environment and the expected value of execution times. H. Casanova et al. [7] and R.Baraglia et al. [2] proposed the heuristic algorithms to solve the scheduling problem based on the different static data, for example, the execution time and system load. Unfortunately, all information such as execution time and workload cannot be determined in advance of dynamic grid environments.

Scheduling algorithms dealing with preemptable tasks have also been reported in the literature [20], [10], [23], [14]. Enterprise [22] shows the effectiveness of a bidding model for a decentralized scheduling framework. Genetic algorithm methods are presented in [26] and [18] for minimizing the total task completion time. The algorithms model the scheduling process as a genetic evolution and estimate at which Grid resource a task should be assigned for execution so that the completion time is minimized. A survey evaluation of scheduling algorithms is presented in [15]. Stochastic evaluation of fair scheduling algorithms is also presented in [13], where networking issues are discussed. Finally, evaluation of different scheduling mechanisms for Grid computing is also presented in [1], such as the First Come First Served (FCFS), the Largest Time First (LTF), the Largest Cost First (LCF), the Largest Job First (LJF), the Largest Machine First (LMF), 1the Smallest Machine First (SMF), and the Minimum Effective Execution Time (MEET). R. Buyya [6] have proposed an economic based scheduling technique that optimizes cost and time. A drawback of the previously mentioned approaches is that scheduling is performed without taking into account fair considerations. Doulamis et al in Fair Scheduling Algorithms in the Grids [24] have considered the fairness issue but it is a non-preemptive algorithm.

The main objective of this algorithm would be to improve the fairness and decrease the total completion time by minimizing the service time error. The tasks are assigned the processors according to the error value of each job.

3. GRID MODEL

The grid *G* in our study consists of a site in each of which a set of N computational hosts is participating in a grid. More formally, the hosts are represented as *{N1, N2… Nr}* .Let N = *{N1, N2…, Nr}* denote a set of all hosts in *G* and each host consists of a number of processors.

3.1. Problem Formulation

The problem of Job Scheduling on Computational Grids [8] basically consists of a dynamic set of T independent tasks to be scheduled on a dynamic set of N resources. An instance of the problem consists of:

• A set of T independent tasks to be scheduled. Each job has associated with it a workload (in million of instructions). Every job must be entirely executed in a unique machine.

• A set of M number of processors which has its corresponding computing capacity (in mips)

A multiprocessor system of M processors and that the computation capacity of processor j is equal to cj units of capacity. The total computation capacity C of the Grid G is defined as

$$C = \sum_{j=1}^{M} c_j \qquad (1)$$

The earliest time a task would be started on a processor j is defined as the maximum delay in making a decision to assign the task to a processor j and the time the task gets the processor exactly.

4. FAIR SCHEDULING BASED ON ERROR

The scheduling algorithms described in the previous section do not address the issue of fairness. A precise definition of fairness is essential before further discussion of fair scheduling of tasks. The classic notion [17] of fairness in the allocation of resource among multiple requesting entities with equal rights to the resource but unequal demands, is as follows.

- The resource is allocated in order of increasing demand
- No requesting task gets a share of the resource larger than its demand
- Requesting tasks with unsatisfied demands get equal shares of the resources

Tasks with a higher demand are favored against the remaining tasks in the case of other existing algorithms which means that such tasks are given a higher priority than the others which leads to starvation that increases the completion time of tasks and no fairness is guaranteed. These issues are addressed in the algorithm that we propose which alllocate resources fairly to all tasks based on the

error [21]. The algorithm is oriented towards large scale computing in which multiple processes are taken into account.

4.1. Service Time Error Algorithm

Proportional share scheduling for a given set of tasks have associated weights[16] or shares, and a proportional share scheduler should allocate resources to each task in proportion to its respective weight. More specifically we assume that each task is assigned an integer share determined, for example by the user's contribution to the grid infrastructure or by the price he is willing to pay for the services he receives. The process of scheduling is modeled in two steps as

1) the scheduler orders the tasks in a queue

2) the scheduler runs the first task in the queue for its *time quantum*, which is the maximum time interval the client is allowed to run before another scheduling decision is made.

As mentioned earlier the algorithm works towards perfect fairness defined as an ideal state in which each task has received service exactly proportional to its weight. This algorithm is based on an error that occurs during the service of the request. The probability that a request has been serviced for *t* units of time will terminate in the next *dt* units of time. In preemptive scheduling this applies every time a request is scheduled to run after an interruption.

We denote the proportional share of task A as SA, and the time interval would be the difference of time between its arrival t1 and the execution time t2. The *amount* of service received by task A during the time interval (t1, t2) is represented as WA (t1, t2). If an ideal system exists wherein all tasks could consume their resources allocation simultaneously, then the scheduler can maintain the perfect state at all intervals. However in a time multiplexing environment it is not possible to be proportionally fair at all intervals and also in the real world no algorithm maintains perfect fairness. The idea is to quantify how close an algorithm works towards perfect fairness.

A slight variation of equation 2 defines the *service time error* [16] for a task A ,EA (t1, t2) is the difference between the amount of service allocated to the task during an interval (t1, t2) and the amount of time that would have been allocated under an ideal scheme that maintains perfect fairness for all tasks over all intervals. Mathematically service time error is represented as

$$E_A(t_1,t_2) = W_A(t_1,t_2) - (t_2-t_1)\, S_A/\Sigma S_i \qquad (2)$$

The computation of this error could be a positive value which indicates that a task has received more than its ideal share over an interval; a negative value indicates that a task have received less than what it deserves and a zero value indicates that it has received its ideal share. The main objective of the algorithm is to minimize the error and reduce the completion time of the tasks.

The input consists of T number of tasks submitted to Grid G consisting of N number of computation nodes and each node having M multiple processors.

A queue is created for each node and the tasks are placed in the queue for execution. Each task is assigned a weight depending on various factors depending on the infrastructure.The tasks are ordered in the decreasing order of their weights. Initially when the processors are free the first task in the queue is assigned to the processor for the first time quantum that is fixed by the grid infrastructure. The service time error is calculated for the current task in execution denoted by Ec, the task that is in the head of the queue which is denoted by Ef and also for the job that is the second task in the queue represented as En

To determine as to which job will be given the next time quantum depends on the error value if the error is positive then it has been given enough of resources so it is moved to the end of the queue. The error values of the first job and the second job in the queue are compared and the job that has a lower error value gets the resource for the next time quantum. This process repeats until there are no more jobs in the queue for that node.

Input: A set of tasks T, a set N computation node with multiple processors

Output: A schedule of *T* onto N

1 Create a set *Q* of N queues

2. *qsize* = |*T*| / |N|

3 **for** each queue, *qi* in *Q*

4.. Remove *qsize* tasks in *T* and enqueue them to *qi*

5. While there are tasks in the queues do

6.Assign weights to the tasks

7. Arrange the tasks in decreasing order of their weights

8. The first task in queue is executed initially for the required time quantum to the node that is available

9. Calculate the service time error Ec for the current job

10. If the Ec > 0 move the job to the end of the queue then calculate the error value of the first job in the queue Ef and the next job in the queue then

11. If En >= Ef 6

12. j = first job of the queue

13. else

14. j = next job in the queue

15. end if /* jobs are assigned to the free nodes*/

16. else

17. j= current job in the queue
18. end if
19. end while
20. end for

Fig.1Pseudocode for Service Time Error Algorithm

4.2. Objective Evaluation

The tasks arrival is modeled as an application of a queuing system [19]. Modelling each queue it can be shown that the average length of the queue

$$Q = \frac{\rho^2}{2(1-\rho)} \tag{3}$$

The average waiting time is given by

$$WT = \frac{\rho}{2\mu(1-\rho)} \tag{4}$$

The Task Completion Time is given by

$$TC = \frac{2-\rho}{2\mu(1-\rho)} \tag{5}$$

Another criterion would be to minimize the error using root mean square

$$Z = \sqrt{\left(\frac{\sum_{n=1 to m} \left(W_n - \{t_{2n} - t_{1n}\} * S_n / \sum_{n=1}^{m} S_n \right)^2}{m} \right)} \tag{6}$$

subject to the contraints

$W_{n+1}=f(t_{1n},t_{2n})$

$S_{n+1}=g(t_{1n},t_{2n})$

$t_{2n} \geq t_{1n}$

$\sum_{n=1}^{m} S_n = a(const.)$

$m \geq 2$

$t_{1n},t_{2n} \geq 0$

As the primary function of a scheduler is to select a client to execute when the resource is available. A key benefit of this algorithm is that it can select a task to execute in $O(1)$ time.

5. EXPERIMENTAL RESULTS

The proposed algorithm is simulated using GridSim toolkit [25] which provides the implementation of a grid infrastructure as depicted in Fig 2. Section 5.1 gives a brief description about the scheduler architecture as adopted by the simulator. It allows the basic functionality to create common entities such as computation resources, users, processing elements of varying capacity.

5.1. Scheduler Architecture

Fig 2 depicts an architecture adopted in the GridSim [25] infrastructure. When the simulator starts, Grid Resource Entities sends an event to GIS entities for registration. Hence, GIS entities return a list of registered resources and their details to Metascheduler. Therefore, Grid Client Entities submit the jobs and its details such as arrival time, releasing time, length of job and the request of resources configuration, properties, etc, to Meta-scheduler. The Meta-scheduler responds with dynamic information such as resources workload, available resources, capability, and the other properties. With each computation node is associated a local queue where the jobs are placed for execution.

5.2. Simulated Results

In this section, the proposed scheduling scheme is simulated against 1) a large set of tasks 2) a large and varying number of processors. Table 1 and Table 2 shows the Grid Resource Infrastructure and different workloads respectively. The simulations were performed with the above mentioned infrastructure and the workload. The same set of data is tested for FCFS, Round Robin and rhe proposed Service Time Error algorithm. Fig 2 The performance is compared in terms of the average cost, Task completion time, Error values and the results are shown in Table 3. Fig 2 and Fig 3 depict the results in terms of task completion time and the error values.

Table 1 The Grid Resources Attributes

Parameters	Values	Notations
Total Number of Resources	10-20	Machines
Speed	200-400	Million Instructions per Second
Number of Processors	5-6	Processing Elements

Table 2 Workload Attributes

Parameters	Values	Notations
Total number of jobs	100 – 2000	
Length of a job	1,000 – 5,000	Million Instructions (MI)
Number of processors Required	5-6	Million Instruction Per second(MIPS)

Table 3

Comparison of performance in terms of Average Cost, Task Completion Time, Maximum and Minimum Error Value for FCFS, Round Robin and the Service Time Error.

Number of Tasks 100

Parameters	Average Cost	Completion Time	Min Error	Max Error
FCFS	236232.0	9321	-1.20000	0.899999
Round Robin	210345.2	8102	-0.79999	0.83333
Error Based	210345.2	6766	-0.5	0.5

Number of Tasks 1000

Parameters	Average Cost	Completion Time	Min Error	Max Error
FCFS	420452.9	8251.93	-5.46000	6.941818
Round Robin	355897.6	6169.05	-4.56765	5.636500
Error Based	311192.8	5509.62	-3.27989	3.970909

In all experiments the arrival model described in Section 4.2 is adopted. For a minimal set of tasks all the algorithms efficiently schedule the tasks but as the number of tasks increase Service Time Error algorithm yields the best performance and Round Robin the next highest. On the contrary FCFS shows the worst performance.

6. CONCLUSIONS

In this paper we have proposed a scheduling algorithm for the Grid environment that could be used to implement scheduling in a fair way. This algorithm has proved to outperform the results in terms of Cost, completion time and the error. In particular, the algorithm allocates the tasks to the available processors so that no requesting task gets a share of the resource larger than its demand and requesting tasks with unsatisfied demands get equal shares of the resources. It also guarantees that all tasks are considered for execution. This algorithm can be integrated in the existing Grid computing systems to improve the task allocation performance.

7. REFERENCES

[1] I. Ahmad, Y.-K. Kwok, M.-Y. Wu, and K. Li, "Experimental Performance Evaluation of Job Scheduling and Processor Allocation Algorithms for Grid Computing on Metacomputers," Proc. IEEE 18th Int'l Parallel and Distributed Processing Symp. (IPDPS '04) , pp 170-177, 2004.

[2] R. Baraglia, R. Ferrini, and P. Ritrovato, "A static mapping heuristics to map parallel applications to heterogeneous computing systems", Research articles. *Concurrency and Computation : Practice and Experience*, 17(13):1579–1605, 2005.

[3] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski, "The GrADS Project: Software Support for High-Level Grid Application Development," Int'l J. High Performance Computing Applications, vol. 15, no. 4, pp. 327- 344, Winter, 2001.

[4] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I.Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen and R.F.Freund (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(6): Pages 810-837.

[5] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/G : An Architecture for a Resource Management and Scheduling System in a Global Computational Grid," Proc. Fourth Int'l Conf. High Performance Computing in Asia-Pacific Region, 2000

[6] R. Buyya, Economic - based Distributed Resource Management and Scheduling for Grid Computing, PhD Thesis, Monash University, Melbourne, Australia, April 12, 2002.

[7] H. Casanova, A. Legrand, D. Zagorodnov and F. Berman, "Heuristics for scheduling parameter

sweep applications in Grid environments", *in Heterogeneous Computing Worksho p 2000*, IEEE Computer Society Press, 2000, pp. 349–363.

[8] K. Czajkowski, I. Foster, and C. Kesselman. Resource co-allocation in computational grids". In Proceedings of the The Eighth IEEE International Symposium on High Performance Distributed Computing. IEEE Computer Society Washington, DC, USA, 1999.

[9] H. Dail, H. Casanova, and F. Berman, "A Decoupled Scheduling Approach for the GrADS Environment" Proc. Conf. Supercomputing (SC'02), Nov. 2002.

[10] X. Deng, N. Gu, T. Brecht, and K.-C. Lu, "Preemptive Scheduling of Parallel Jobs on Multiprocessors," SIAM J. Computing, vol. 30, no. 1, pp. 145-160, 2000.

[11] I. Foster and C. Kesselman. "The Grid – Blueprint for a New Computing Infrastructure". Morgan Kaufmann Publishers, 1998.

[12] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," Int'l J. Supercomputer Applications, vol. 15, no. 3, 2001.

[13] M. Hawa, "Stochastic Evaluation of Fair Scheduling with Applications to Quality- of - Service in Broadband Wireless Access Networks," PhD dissertation, Univ. of Kansas, Aug. 2003.

[14] L.E.Jackson and G.N. Rouskas, "Deterministic Preemptive Scheduling of Real Time Tasks," Computer, vol. 35, no. 5, pp. 72- 79, May 2002

[15] R. Jain, A Survey of Scheduling Methods . Nokia Research Center, Sept. 1997.

[16] Jason Nieh Chris Vaill, Hua Zhong, "Virtual - Time Round Robin: An O(1) Proportional Share Scheduler", Proc . of the 2001 Usenix Technical Annual Technical Conference, Boston, USA, 2001

[17] S. Keshav, An Engineering Approach to Computer Networks Reading Mass: Addison- Wesley, 1997

[18] S. Kim and J.B. Weissman, " A Genetic Algorithm Based Approach for Scheduling Decomposable Data Grid Applications," Proc. Int'lConf. Parallel Processing (ICPP '04), pp. 406- 413, 2004.

[19] L.Kleinrock and Arnne Nilsson, "On Optimal Scheduling Algorithms fo a Time Shared Systems" , Journal of the Assocla Uon for Computing Machinery, Vol 28, No 3. July 1981, pp 477-486

[20] J.Y-T. Leung and M.L. Merrill, "A Note on Preemptive, S cheduling of Periodic, Real-Time Tasks," Information Processing Letters, pp. 115- 118, Nov. 1980.

[21] D.Lopez and Rasika, "A Service Time Error Algorithm for a Computational Grid", Proc. Int'l conf.

[22] T. W. Malone, R. E. Fikes, K. R. Grant and M. T. Howard " Enterprise : A Market - Like Task Scheduler for Distributed Computing Environments", The Ecology of Computation, B. A. Huberman,ed.,pp. 177-205, 1988

[23] G. Manimaran and C.S.R. Murthy , " An Efficient Dynamic Scheduling Algorithm for Multiprocessor Real -Time Systems," IEEE Trans. Paralleland Distributed Systems, vol. 9 no. 3, pp. 312- 319, Mar. 1998.

[24] Nikolaos D. Doulamis, Anastasios D. Doulamis, "Fair Scheduling Algorithms In Grid"," IEEE Trans. Parallel and Distributed Systems, vol. 18, no. 11, Nov2007.

[25] Rajkumar . Buyya and M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for Grid Computing", *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, 14:1175-1220, 2002.

[26] D.P. Spooner, S.A. Jarvis, J. Cao, S. Saini, and G.R.Nudd, "Local Grid Scheduling Techniques Using Performance Prediction," IEE Proc. Computers and Digital Techniques, vol. 150, no. 2, pp. 87-96, Mar. 2003

[27] Stephen D. Kleban Scott H. Clearwater " Fair Share on High Performance Computing Systems: What Does Fair Really Mean?" Proceedings of the 3rd IEEE/ ACM International Symposium on Cluster Computing and the Grid (CCGRID.03)

[28] H. Topcuoglu, S. Hariri, and M. Wu, "Performance- effective and low - complexity task scheduling for heterogeneous computing. IEEE transactions on Parallel and Distributed Systems 13,(3): 260-274, March 2002. Middleware for Grid Computing"

9. FIGURES

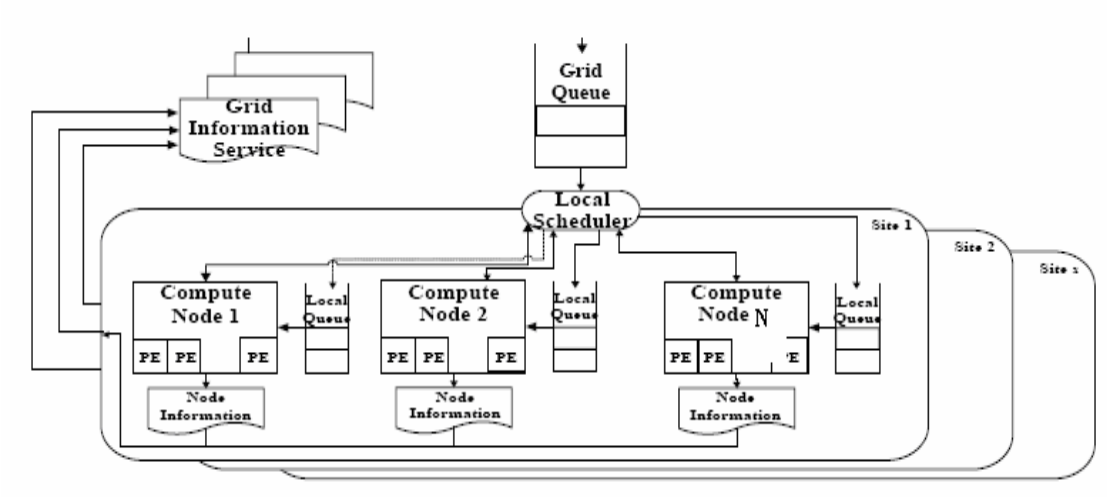

Fig 2: The Scheduler Architecture Adopted From The Gridsim Infrastructure

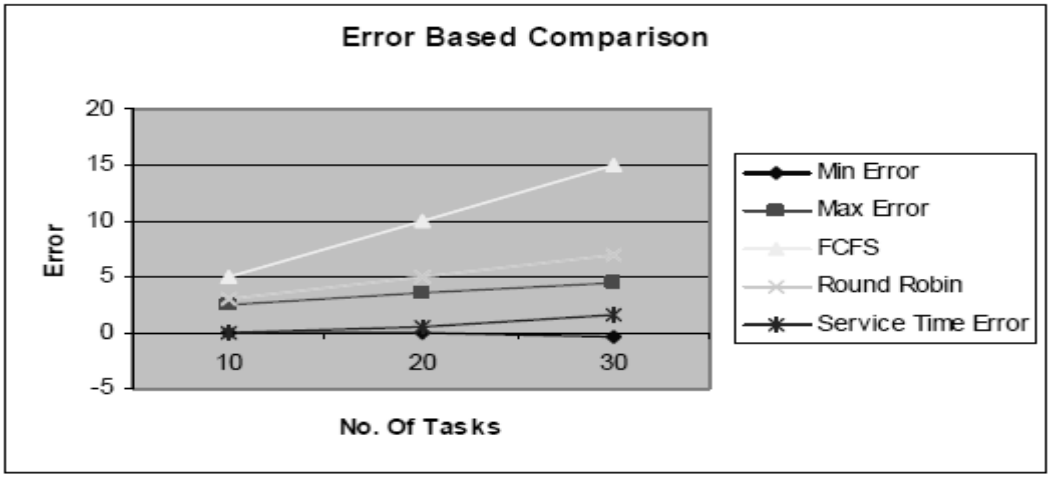

Fig 3: Comparison Based On Error Values

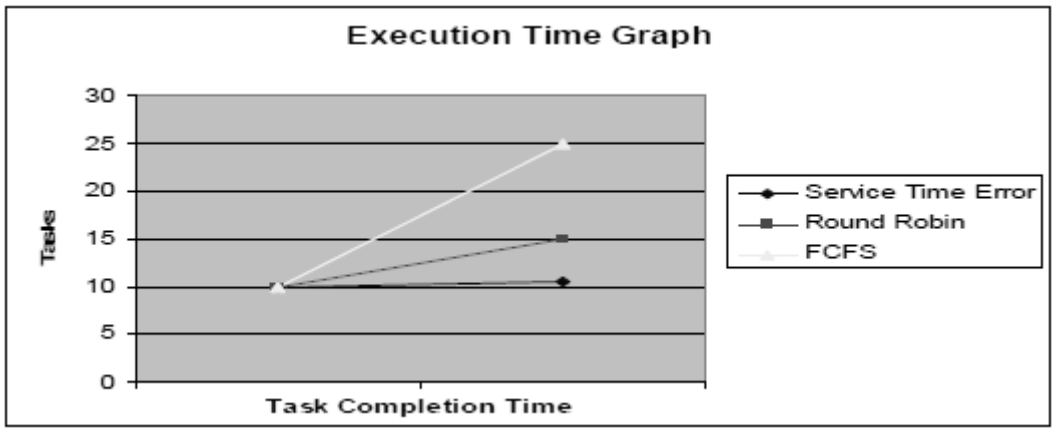

Fig 4 : Comparison Based On The Task Completion Time