A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling

U.Karthick Kumar[1]
[1]**Department of MCA & Software Systems,VLB Janki Ammal Arts and Science College, Coimbatore,TamilNadu – 641 042,India**

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 Load balancing algorithm for fair scheduling, and we compare it to other scheduling schemes such as the Earliest Deadline First, Simple Fair Task order, Adjusted Fair Task Order and Max Min Fair Scheduling for a computational grid. It addresses the fairness issues by using mean waiting time. It scheduled the task by using fair completion time and rescheduled by using mean waiting time of each task to obtain load balance. This algorithm scheme tries to provide optimal solution so that it reduces the execution time and expected price for the execution of all the jobs in the grid system is minimized. The performance of the proposed algorithm compared with other algorithm by using simulation.

Keywords: *Computational Grid, Scheduling, Load balancing, Fair scheduling, Mean Waiting Time, Execution Cost*

1. Introduction

Grid computing has been increasingly considered as a promising next-generation computing platform that supports wide area parallel and distributed computing since its advent in the mid-1990s [1]. It couples a wide variety of geographically distributed computational resources such as PCs, workstations, and clusters, storage systems, data sources, databases, computational kernels, and special purpose scientific instruments and presents them as a unified integrated resource [2].

In computational grids, heterogeneous resources with different systems in different places are dynamically available and distributed geographically. The user's resource requirements in the grids vary depending on their goals, time constraints, priorities and budgets. Allocating their tasks to the appropriate resources in the grids so that performance requirements are satisfied and costs are subject to an extraordinarily complicated problem. Allocating the resources to the proper users so that utilization of resources and the profits generated are maximized is also an extremely complex problem. From a computational perspective, it is impractical to build a centralized resource allocation mechanism in such a large scale distributed environment [3].

A computational grid is less expensive than purchasing more computational resources while obtaining the same amount of computational power for their computational tasks. A key characteristic of Grids is resources are shared among various applications, and therefore, the amount of resources available to any given application highly varies over time.

1.1 Dynamic Load Balancing

Load balancing is a technique to enhance resources, utilizing parallelism, exploiting throughput improvisation, and to reduce response time through an appropriate distribution of the application. Load balancing algorithms can be defined by their implementation of the following policies [15]

Information policy: It states the workload of a task information to be collected, when it is to be collected and from where.

Triggering policy: It determines the appropriate period to start a load balancing operation.

Resource type policy: It order a resource as *server* or *receiver* of tasks according to its availability status.

Location policy: It uses the results of the resource type policy to find a suitable partner for a server or receiver.

Selection policy: defines the tasks that should be migrated from overloaded resources (source) to most idle resources (receiver).

Load balancing algorithms are defined by two types such as static and dynamic [16]. Static load balancing algorithms allocate the tasks of a parallel program to workstations. Multicomputers with dynamic load balancing allocate or reallocate resources at runtime based on task information, which may determine when and whose tasks can be migrated. In this paper Dynamic Load Balancing Algorithm is implemented to multicomputers based on resource type policy.

The remaining section of this paper is organized as follows. Section 2 explains the related work. Section 3 detailed Problem formulation, Section 4 explained Fair Scheduling and Section 5 detailed the Dynamic Load Balancing Algorithm and section 6 Results and Discussion are detailed and conclusion and future work is presented in section 7.

2. Related Work

Fair Share scheduling [4] is compared with Simple Fair Task Order Scheduling (SFTO), Adjusted Fair Task Order Scheduling (AFTO) and Max-Min Fair Share Scheduling (MMFS) algorithm are developed and tested with existing scheduling algorithms. Somasundaram, S. Radhakrishnan compares Swift Scheduler 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[5]. For a multiprocessor system, the authors in [6] have shown that heuristic schemes that takes into account both the task deadline and EST better performs than the EDF, LLF, and MPTF algorithms. Finally, evaluation of different scheduling mechanisms for Grid computing is also presented in [7], 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), the Smallest Machine First (SMF), and the Minimum Effective Execution Time (MEET).

Pal Nilsson and Michal Pioro have discussed Max Min Fair Allocation for routing problem in a communication Network [8]. Hans Jorgen Bang, Torbjorn Ekman and David Gesbert has proposed proportional fair scheduling which addresses the problem of multi-user diversity scheduling together with channel prediction[9]. Daphne Lopez, S. V. Kasmir raja has described and compared Fair Scheduling algorithm with First Come First Serve (FCFS) and Round Robin(RR) schemes[10].

Load Balancing is one of the big issues in Grid Computing [11], [12]. Grosu and Chronopoulos [13], Penmatsa and Chronopoulos [14] considered static load balancing in a system with servers and computers where servers balance load among all computers in a round robin fashion. Qin Zheng, Chen-Khong Tham, Bharadwaj Veradale to address the problem of determining which group an arriving job should be allocated to and how its load can be distributed among computers in the group to optimize the performance and also proposed algorithms which guarantee finding a load distribution over computers in a group that leads to the minimum response time or computational cost [12]. Saravanakumar E. and Gomathy Prathima, discussing A novel load balancing algorithm in computational Grid [17]. M.Kamarunisha, S.Ranichandra, T.K.P.Rajagopal, dicuss about Load balancing Algorithm types and three policies are Information policy, Triggering Policy, and Selection Policy in Grid Environment[15][16].

3. Problem Formulation

Let the number of tasks be N that have to be scheduled as T_i, i=1, 2… N, is the duration of the task when executed on a processor in million instruction per second (MIPS). Let number of processors is M and its total computation capacity C is defined as

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

Let M is the multiprocessor and its computation capacity of processor j is defined by c_j. The earliest time of task i started from processor j is the maximum of communication delay and completion time between i^{th} task and j^{th} processor. The completion time of task is zero, when no task allocated to processor j, otherwise it estimated the remaining time that are already allocated to processor j.

In the fair scheduling algorithm, the demanded computation rate X_i of a task T_i will play an important role. It estimated by the computation capacity that the Grid should allocate to task T_i for it to finish just before its requested deadline

4. Fair Scheduling

The scheduling algorithms do not adequately address congestion, and they do not take fairness considerations into account. Fairness [4] is most essential for scheduling of task. In Fair Scheduling, the tasks are allocated to multiple processors so that the task with unsatisfied demand get equal shares of time is as follows:

- Tasks are queued for scheduling according to their fair completion times.
- The fair completion time of a task is estimated by its fair task rates using a max-min fair sharing algorithm.
- The tasks are assigned to processor by increasing order of fair completion time.

In this algorithm, tasks with a higher order are completed first which means that tasks are taken a higher priority than the others which leads to starvation that increases the completion time of tasks and load balance is not guaranteed.

For this issue we propose a Load Balance (LB) Algorithm to give uniform load to the resources so that all task are fairly allocated to processor based on balanced fair rates. The main objective of this algorithm is to reduce the overall makespan.

5. Dynamic Load Balancing Algorithm

Dynamic load balancing algorithms make changes to the distribution of work among workstations at run-time; they

use current or recent load information when making distribution decisions. Multicomputers with dynamic load balancing allocate/reallocate resources at runtime based on a priori task information, which may determine when and whose tasks can be migrated. As a result, dynamic load balancing algorithms can provide a significant improvement in Performance over other algorithms.

Load balancing should take place when the scheduler schedules the task to all processors. There are some particular activities which change the load configuration in Grid environment. The activities can be categorized as following:

- Arrival of any new job and queuing of that job to any particular node.
- Scheduler schedules the job to particular processor.
- Reschedule the jobs if load is not balanced
- Allocate the job to processor when its free.
- Release the processor after it complete the whole job

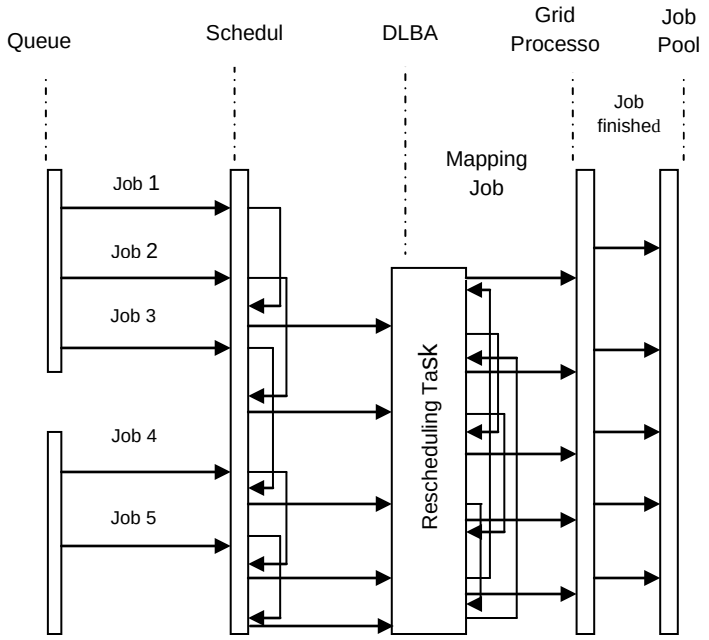

Fig.1: An Event Diagram for Dynamic Load Balancing Algorithm

Initialization of algorithm: N number of tasks that have to be scheduled and workload $w_i(x)$ of tasks are submitted to M number of processors.

Scheduling task: Scheduler allocates number of demanded tasks to M number of processors based on fair completion time of each task.

Load Balancing Algorithm: It applied when the processor task allocation is excessive than the other after scheduling the task.

Balancing criterion: Rescheduled the task for upper bound and lower bound processor based on $W_t(x)$.

Termination: This process is repeated until all the processor is balanced. Finally, obtain the optimal solution from the above process.

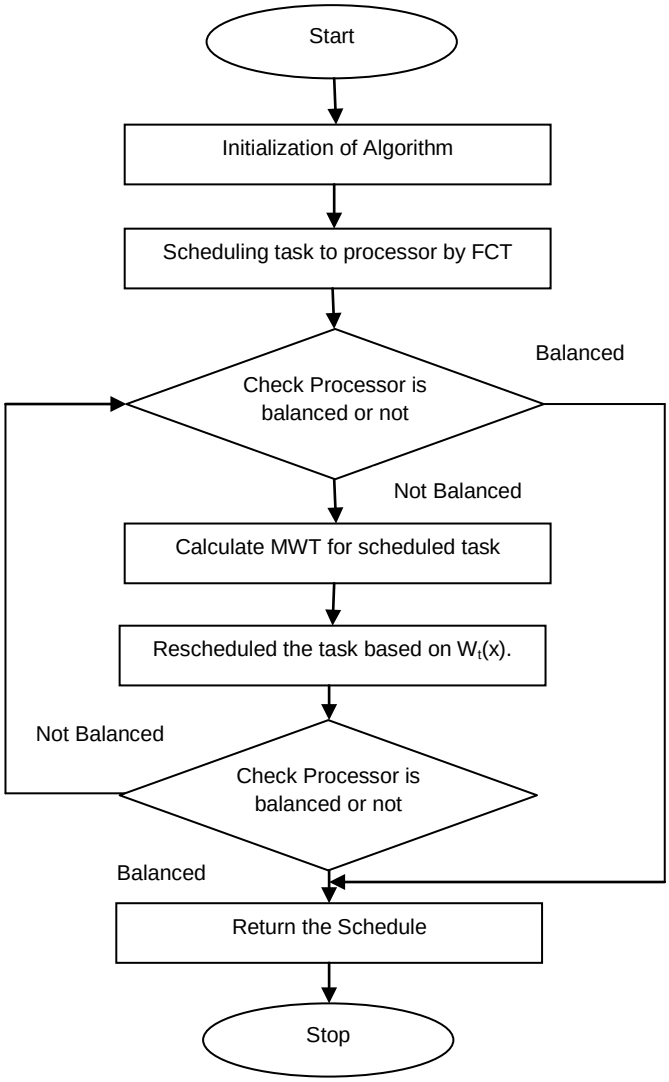

Fig.2 Flow Chart of Algorithm

5.1 Segment of code related to Algorithm

Input: A set of N task and M number of processor with computational capacity c_j.
Output: A schedule of N task
1. Create set of Queues.
2. qsize < N/M.
3. For each queue q_i in Q
4. While there are tasks in the queue do,
5. Assign demand rate of the task, X_i
6. k= C/N

7. If $X_i < k$
8. Assign X_i to i^{th} task as fair rate.
9. Else
10. Assign k to i^{th} task as fair rate.
11. Calculate fair completion time $t_i(x)$.
12. End while
13. End Loop
14. Arrange the task in increasing order based on their $t_i(x)$ and submitted to processor.
15. While (Load of any processor is greater than average load processor) do
16. Calculate mean waiting time for each scheduled task
17. If $Z_x^y > 0$
18 Migrated tasks are determined by using criteria of processor capacity.
19. Each processor which has least capacity is selected for migration.
20. End If
20. End While

5.2 Objective Evaluation

The task are scheduled by fair completion time $t_i(x)$, which is obtained by

$$t_i(x) = \int_1^M \left(d(xy)\, c(y) \Big/ c(y) \right) + w(x)/r(x)\ dx \qquad (2)$$

Here d(xy) is the earliest start time of i^{th} task to j^{th} processor i=0,1,…N and j=0,1,2,…,c(y) is the computational capacity of j^{th} processor, w(x) is workload of the task and r(x) is the fair rate of task computed by Max Min Fair Share approach.

Mean waiting time $W_t(x)$ is given by

$$W_t(x) = \sqrt{(\rho^n \lambda x^2 + x^n (1-\rho))/2(1-\rho)^2} + W(x) \qquad (3)$$

Where, W(x) is the constant delay made by the resource manager to assign to processor and arrival of all files necessary to run the task on processor.

$$W(x) = \rho x^2/2(1-\rho) \qquad (4)$$

To find the migration of processor by

$$Z_x^y = {}^1/_{W_t(x)} \int_1^M (W_t(x) - c(x))\, dx \qquad (5)$$

$$t(x_m) = max\{t[min(o, Z_x^y)], o\} \qquad (6)$$

$$t(y_n) = min\{t[max(o, Z_x^y)], o\} \qquad (7)$$

$$t(x_m) > t(y_n)\ and\ t(x_m), t(y_n) \geq 0 \qquad (8)$$

Based on mean waiting time task are rescheduled and allocated to processor. This is continued until all the processors are equally balanced to reach their minimum makespan.

5.3 Execution Cost

Our main objective is to reduce makespan and total execution cost by using load balancing algorithm. Specifically, we define the following for cost as

- $C(W(x), x)$ is cost incurred by a customer with seconds x ,if the expected constant delay is *W(x).*
- $W_t(x, l)$ is Mean Waiting time of processor with seconds x, if the rescheduling load balance algorithm is l .
- $Cost_{exe}(x, l)$ is Total execution cost of using load balance algorithm.

Cost optimization is defined by

$$Cost_{exe}(x) = \int_1^M C(W(x), x)\, dx \qquad (9)$$

The optimization problem is formulated by

$$Cost_{exe}(x, l) = \underbrace{Min}_{w_t(x,l)} \int_1^M C(W(x), x)\, W_t(x, l)\, dx \qquad (10)$$

As the primary function of a scheduler is to select a client to execute their tasks to processors when it is free. A key benefit of this algorithm is to reschedule the task by using $W_t(x)$ so that overall execution time and cost is reduced.

6. Result and Discussion

In this section proposed algorithm is simulated against

- Large set of Tasks as 256, 512, 1024, 2048 Million Instruction (MI).
- Large and varying number of processors as 8, 16, 32, 64 Million Instruction Per Second (MIPS).

Here, cost rate range is taken from 5 – 10 units is randomly chosen and assigned according to speed of the processor. Speed of the processor ranges from 0 – 1MIPS are randomly assigned to M processor. Below table shows the comparison results of proposed algorithm The work is approximately gives 45% - 25% less than EDF and 7% - 5% less than SFTO and AFTO and 5% - 2% less than MMFS for makespan. Also, LBA approximately show 30% - 25% less than EDF and 7% - 6% less than SFTO and AFTO 2% - 1% less than MMFS for Execution cost. The result shows better performance for Higher Matrix also. The following are the comparison result of existing and proposed method.

Table 1: Performance Comparison for 8 processors

Parameters	*Resource Matrix*	*EDF*	*SFTO*	*AFTO*	*MMFS*	*LBA*
Makespan	256 x 8	917.82	447.74	444.39	439.61	418.13
Cost		5506.91	4487.44	4468.54	4446.77	4023.57
Makespan	512 x 8	1121.32	1022.36	1010.09	858.54	836.72
Cost		7849.21	5111.8	5048.45	4292.71	4183.58
Makespan	1024 x 8	1825.33	1651.45	1686.17	1643.32	1599.82
Cost		10951.97	13211.63	11803.21	13180.96	12798.55
Makespan	2048 x 8	3596.42	3280.39	3247.63	3137.59	3095.82
Cost		28174.94	26243.11	25981.06	25100.75	24766.55

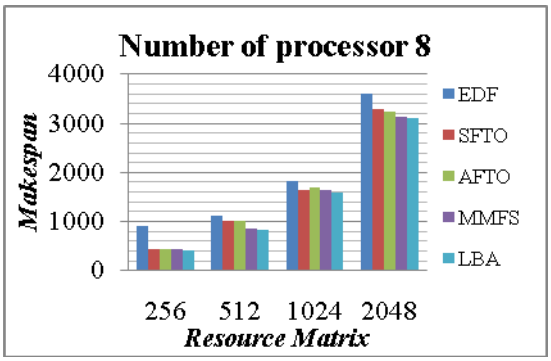

Fig 3: Performance Comparison for Makespan

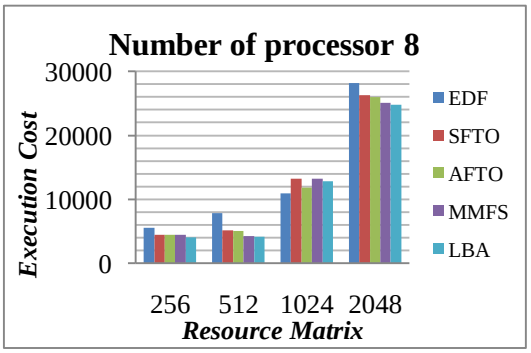

Fig 4 : Performance Comparison for Execution Cost

Table 2: Performance Comparison for 16 processors

Parameters	*Resource Matrix*	*EDF*	*SFTO*	*AFTO*	*MMFS*	*LBA*
Makespan	256 x 16	1466.72	304	300.65	295087	209
Cost		7332.11	1520	1511.0	1489.33	1045
Makespan	512 x 16	1366.48	553.89	555.37	545.76	483.57
Cost		8664.81	5868.91	5603..75	5508.24	4435.69
Makespan	1024 x 16	1540.27	1309.94	1296.35	1301.81	1231.43
Cost		9241.6	6549.72	6481.77	6519.05	6157.14
Makespan	2048 x 16	3352.67	2742.53	2761.98	2734.4	2641.04
Cost		26468.72	24682.76	27619.81	24652.09	23769.35

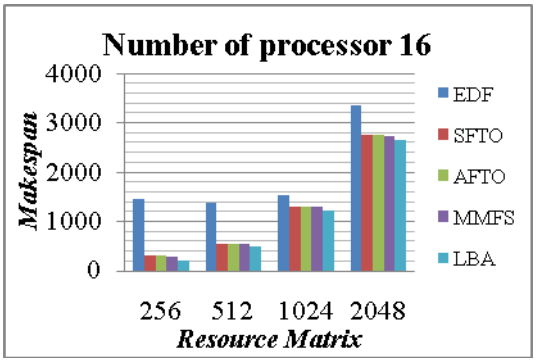

Fig 5: Performance Comparison for Makespan

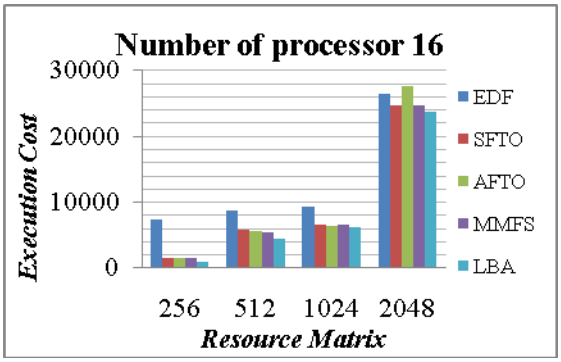

Fig 6: Performance Comparison for Execution Cost

Table 3: Performance Comparison for 32 processors

Parameters	*Resource Matrix*	*EDF*	*SFTO*	*AFTO*	*MMFS*	*LBA*
Makespan	256 x 32	206.05	183.43	180.08	175.30	114.64
Cost		1648.40	917.15	908.25	886.48	573.22
Makespan	512 x 32	744.80	580.60	577.25	574.27	464.54
Cost		5958.36	5225.43	5216.53	3445.64	2787.23
Makespan	1024 x 32	966.47	912.96	937.07	904.83	863.54
Cost		7731.78	4564.78	7512.53	4534.11	4317.72
Makespan	2048 x 32	1675.05	1427.97	1375.23	1370.11	1262
Cost		18725.38	11851.76	11377.09	11330.99	10359.85

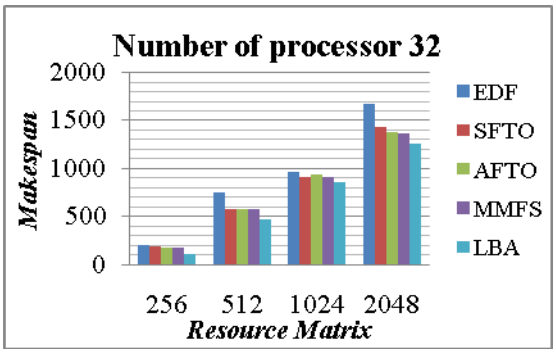

Fig 7: Performance Comparison for Makespan

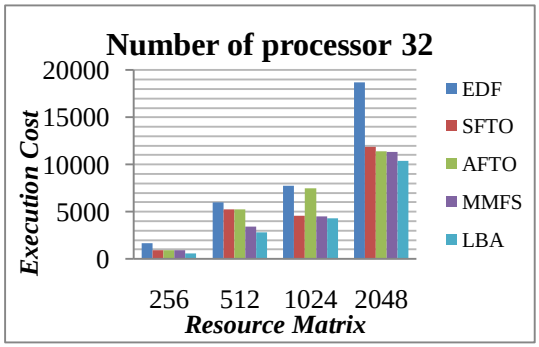

Fig 8: Performance Comparison for Execution Cost

Table 4: Performance Comparison for 64 processors

Parameters	*Resource Matrix*	*EDF*	*SFTO*	*AFTO*	*MMFS*	*LBA*
Makespan	256 x 64	305.35	281.93	278.80	273.80	211.45
Cost		2748.13	1691.60	1682.70	1660.93	1268.7
Makespan	512 x 64	966.67	600	596.65	591.87	450
Cost		5800.02	3000	2991.10	2969.33	2700
Makespan	1024 x 64	968.49	978.87	975.52	970.74	795.34
Cost		9810.95	9600.75	9265.85	8500.08	7735.36
Makespan	2048 x 64	1984.98	1630.08	1626.73	1621.95	1330.86
Cost		26879.85	26300.85	26291.95	26270.18	23308.63

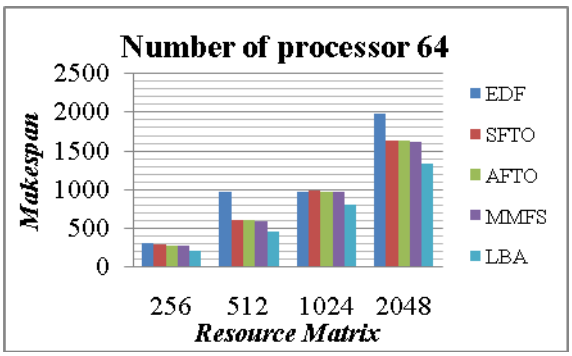

Fig 9: Performance Comparison for Makespan

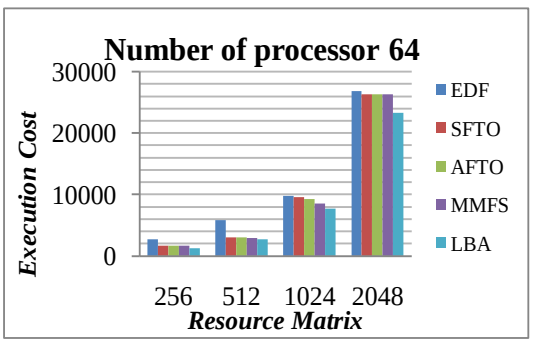

Fig 10: Performance Comparison for Execution Cost

7. Conclusion

In this paper we have proposed a Dynamic load balancing algorithm for the Grid environment that could be used to implement scheduling in a fair way. This algorithm has proved the best results in terms of makespan and Execution Cost In particular the algorithm allocates the task to the available processors so that all requesting task get equal amount of time that satisfied their demand.

Future work will focus on

- Fair scheduling can be applied to optimization techniques
- QoS Constrains such as reliability can be used as performance measure.

Reference

[1] Rajkumar Buyya, David Abramson, and Jonathan Giddy," A Case for Economy Grid Architecture for Service Oriented Grid Computing "**.**

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

[3] Wolski, R., Brevik, J., Plank, J., and Bryan, T., "Grid Resource Allocation and Control Using Computational Economies, In Grid Computing: Making the Global Infrastructure a Reality" Berman, F, Fox, G., and Hey, T. editors, Wiley and Sons, pp. 747--772, 2003.

[4] Doulamis, N.D.; Doulamis, A.D.; Varvarigos, E.A.; Varvarigou, T.A "Fair Scheduling Algorithms in Grids" IEEE Transactions on Parallel and Distributed Systems, Volume18, Issue 11, Nov. 2007 Page(s):1630 – 1648.

[5] K.Somasundaram, S.Radhakrishnan," Task Resource Allocation in Grid using Swift Scheduler", International Journal of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. IV, 2009.

[6] Ramamritham, J.A. Stankovic, and P.-F. Shiah, "Efficient Scheduling Algorithms for Real-Time Multiprocessor Systems,"IEEE Trans. Parallel and Distributed Systems, vol.1, no. 2, pp. 184- 194, Apr. 1990.

[7] 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.

[8] Pal Nilsson1 and Michał Pi´oro," Unsplittable max-min demand allocation – a routing problem".

[9] Hans Jorgen Bang, Torbjorn Ekman and David Gesbert," A Channel Predictive Proportional Fair Scheduling Algorithm".

[10] Daphne Lopez, S. V. Kasmir raja," A Dynamic Error Based Fair Scheduling Algorithm For A Computational Grid", Journal Of Theoretical And Applied Information Technology - 2009 JATIT.

[11] Qin Zheng, Chen-Khong Tham, Bharadwaj Veeravalli," Dynamic Load Balancing and Pricing in Grid Computing with Communication Delay," Journal in Grid Computing (2008).

[12] Stefan Schamberger,"A Shape Optimizing Load Distribution Heuristic for Parallel Adaptive FEM Computations," Springer-Verlag Berlin Heidelberg 2005.

[13] Grosu, D., Chronopoulos, A.T." Noncooperative load balancing in distributed systems",Journal of Parallel Distrib. Comput. **65**(9), 1022–1034 (2005).

[14] Penmatsa, S., Chronopoulos, A.T." Job allocation schemes in computational Grids based on cost optimization",In: Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium,Denver, (2005).

[15] M.Kamarunisha, S.Ranichandra, T.K.P.Rajagopal," Recitation of Load Balancing Algorithms In Grid Computing Environment Using Policies And Strategies An Approach," International Journal of Scientific & Engineering Research Volume 2, Issue 3, March-2011

[16] Prabhat Kr.Srivastava, Sonu Gupta, Dheerendra Singh Yadav," Improving Performance in Load Balancing Problem on the Grid Computing System", International Journal of Computer Applications (0975 – 8887) Volume 16– No.1, February 2011.

[17] Saravanakumar E. and Gomathy Prathima," A novel load balancing algorithm for computational grid," International Journal of Computational Intelligence Techniques, ISSN: 0976–0466 & E-ISSN: 0976–0474 Volume 1, Issue 1, 2010, PP-20-26

U.Karthick Kumar MSc., MCA., M.Phil.,He is a Post Graduate with M.Phil from Bharathiar University, Coimbatore.Now, he is working as a Assistant Professor in VLB Janaki Ammal Arts and Science College, Coimbatore. He has three years of experience in research. He presented paper in International Conference. His Interest areas are Grid Computing, Mobile Computing and Data Structures.