**Journal of Computer Science 10 (6): 1069-1076, 2014**
ISSN: 1549-3636

doi:10.3844/jcssp.2014.1069.1076 Published Online 10 (6) 2014 (http://www.thescipub.com/jcs.toc)

# A NEW-THRESHOLD BASED JOB SCHEDULING FOR GRID SYSTEM

**L. Ramaparvathy**

Department of Computer Applications, SSN College of Engineering, Chennai, India

Received 2014-01-02; Revised 2014-01-30; Accepted 2014-02-03

## ABSTRACT

In heterogeneous distributed systems, utility grids have emerged as a new model of service. In this service, workflow scheduling is one of the challenging problems for satisfying user's quality requirements. One of the main issues in work flow scheduling is to minimize the workflow execution cost in terms of time and makespan. In this study, we propose a new workflow scheduling algorithm based on a novel concept called New-Threshold Based Scheduling (NTBS) that attempts to minimize the cost of workflow execution time and provides service fairness. It works under two phases. The scheduler computes threshold in first phase and in second phase it schedules the grid jobs to reduce the execution time without affecting fairness. From the simulation results it is observed that NTBS gives better performance in terms of reduced makespan and consistent turnaround time as compared to FCFS, EDF and other scheduling algorithms.

**Keywords:** New-Threshold Based Scheduling (NTBS), Makespan, Grid Jobs and Turnaround Time

## 1. INTRODUCTION

Traditional grid scheduling problems are addressed in many papers; there are only a few works on this problem. In these utility grids, it is difficult to solve the scheduling problems because of its multi-objective nature, especially in workflows. In distributed system, workflows establish a model for application description. Yu and Buyya (2005) described the grid workflow taxonomy is described in their work. The problem of task mapping to a suitable resource is done in workflow job scheduling. Task ordering on each resource to satisfy the performance is another important task of workflow job scheduling. As workflow job scheduling is a known problem, many methods have been proposed for homogeneous (Kwok and Ahmad, 1999) and heterogeneous distributed grid systems by several authors (Topcuoglu *et al*., 2002; Bajaj and Agrawal, 2004; Daoud and Kharma, 2008). These scheduling methods try to minimize the execution time of the workflows and suitable for community grids. Current community grid systems concerns are about time, for example the makespan, which is the time spent from the beginning of the first task in a job to the end of the last task of the job. As economic models (Buyya *et al*., 2002) are introduced into grid computing; the economic cost becomes concern for some grid user. Most of the above current workflow scheduling systems uses different workflow job scheduling to minimize the makespan. In this study we have proposed new-threshold based job scheduling algorithm for grid system to reduce the makespan and turnaround time. This new-threshold based job scheduling first estimates the threshold, which is based on user's expected execution time. Then it schedules the user's jobs fairly whose expected execution time is less than the calculated threshold value.

This study is organized as follows: In section 2, we present our system model. Section 3 describes the various Grid scheduling algorithms. The proposed algorithm is illustrated in section 4. Simulation results and its discussion are presented in section 5. Section 6 concludes the study observation.

## 2. SYSTEM MODEL

**Figure 1** describes the system model of grid scheduling model. Grid Information Server (GIS) in the grid system maintains the information about all the available resources like resource id, resource capacity, resource availability, nature of resource and so forth. The grid user submits the job request to scheduler.

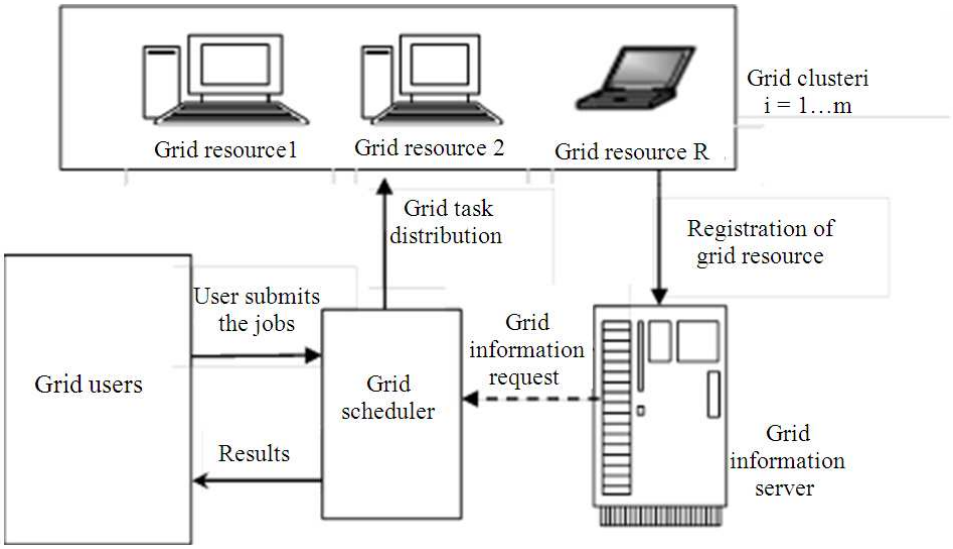


**Fig. 1.** Grid scheduling model

Then the scheduler schedules the job based on scheduling algorithms and assign resources to the grid job with the help of GIS. After job is completed, the available resource is updated in GIS and the job result is send to the grid user.

The resources differ from each other in system processing speed, processing element ids, scheduling policy. The user jobs are also differs in arrival time, execution time, deadline. A task is an atomic unit to be scheduled by the scheduler and assigned to a resource. The properties of a task are parameters like CPU/memory requirement, deadline, priority. A job is a set of atomic tasks that will be carried out on a set of resources. A resource is something that is required to carry out an operation. The job failure model assumed is similar to (Song *et al*., 2006). In this model, the risks between grid jobs and resources are considered. Job failure is generated randomly.

## 3. GRID SCHEDULING ALGORITHMS

Grid scheduling is an important aspect of computational grid. In scheduling, grid scheduler plays the major role in submitting the tasks based on user requirements. Grid scheduling is a framework, with which the scheduler collects resource state information, selects appropriate resources, predicts the potential performance for each candidate schedule and determines the best schedule for the applications to be executed on a grid system subject to some performance goals (Abrishami *et al*., 2012). In principle, scheduling in grids means two things (Khoo *et al*., 2007), ordering and mapping. When there are more than one applications waiting for execution, ordering is performed in order to determine by which order the pending jobs are arranged. Ordering is necessary if jobs with priority or deadline are involved. Mapping is the process of selecting a set of appropriate resources and allocating the set of resources to the grid jobs. For each mapping, the performance potential is estimated in order to decide the best schedule. In general, a scheduling system of grid computing environments aims at delivering better performance. Desirable performance goals of grid scheduling includes: Maximizing resource utilization, minimizing the execution time (Abramson *et al*., 1995) and fulfilling economic constraints (Buyya *et al*., 2000). A task sheduling using ant colony optimization is proposed (Ramesh and Krishnan, 2012). In this study we have proposed a new-threshold based job scheduling to minimize the execution time and turnaround time. Execution time refers to the time duration spent on executing the job. Turnaround time is also called response time. It is defined as the sum of waiting time and executing time. The makespan and turnaround time performance of NTBS is compared with various scheduling algorithms. These scheduling algorithms are described below.

### 3.1. First Come First Served (FCFS) Scheduling

In FCFS (Ahmad *et al*., 2004; Doulamis *et al*., 2007) scheduling algorithm, new set of jobs are added in the job

scheduling queue. The resource is assigned based on queue order to maintain the fairness among grid users. Let N be the total number of user jobs to be executed. Let R be the total number of resource available in a cluster for executing jobs. Here N is being generated randomly. If N is greater than R, workflow scheduling schedules the jobs based on its arrival. The algorithm of FCFS is given by:

- User submits their jobs to the grid scheduler
- The scheduler schedule this jobs in order as it arrived
- Scheduler assigns resource for first R number of jobs and keeps the remaining jobs in the waiting queue
- After completing each job, scheduler assigns the next job in the waiting queue to the available resource
- Turnaround time and Makespan of the scheduler is evaluated

### 3.2. Early Deadline First (EDF) Scheduling

EDF scheduling algorithms (Doulamis *et al*., 2007) compare all the new set of jobs and sort based on deadline time in ascending order. The resources are assigned as per this sorted priority order to achieve better performance. The fairness of the grid user of this scheduler is poor. The algorithm of EDF is given by:

- User submits their jobs to the grid scheduler
- The scheduler schedule this jobs based on jobs deadline time
- Scheduler assigns resource for first R number of sorted jobs and keeps the remaining jobs in the waiting queue
- After completing each job, scheduler assigns the next job in the waiting queue to the available resource
- Turnaround time and makespan of the scheduler is evaluated

### 3.3. Easy Backfilling Scheduling

While the job at the head of the queue is waiting, it is possible for other small jobs to be scheduled especially if they would not delay the start of the job on the head of the queue. It is done in Easy backfilling scheduling (Wong and Goscinski, 2007). Here small jobs in queue are allowed to go ahead to run on idle resources. This scheduling satisfies the user jobs order and hence eliminate the starvation problems. The algorithm of Easy backfilling scheduling is given by:

- User submits their jobs to the grid scheduler
- The scheduler schedule this jobs in order as it arrived
- Scheduler assigns resource for first R number of jobs and keeps the remaining jobs in the waiting queue
- After completing each job, scheduler assigns the next job in the waiting queue to the available resource. If any resource is idle, then suitable small jobs in queue are assigned to those idle resources
- Turnaround time and Makespan of the scheduler is evaluated

### 3.4. Fastest Processor to Largest Task First (FPLTF) Scheduling

Fastest Processor to Largest Task First (FPLTF) (Silva *et al*., 2003) scheduling may require two parameters such as workload of job and resource speed. This schedule collects this information before scheduling. During scheduling process, this scheduler assigns fastest processor resource to the largest job then next largest job and so on. This scheduler is modified version of EDF scheduler. This scheduler performance becomes poor if more number of jobs has heavy workload. The algorithm of FPLTF scheduling is given by:

- User submits their jobs to the grid scheduler along with its workload status
- The scheduler collects resource CPU speed information from GIS
- It schedules these jobs according to largest job to fastest processor nature
- Scheduler assigns resource for first R number of jobs and keeps the remaining jobs in the waiting queue
- After completing each job, scheduler assigns the next job in the waiting queue to the available resource
- Turnaround time and Makespan of the scheduler is computed and evaluated

### 3.5. Min-Min (Min-Min) Scheduling

Min-Min algorithm starts with a set of all unmapped tasks (Song *et al*., 2006). The machine that has the minimum completion time for all jobs is selected. Then the job with the overall minimum completion time is selected and mapped to that resource. The ready time of the resource is updated. This process is repeated until all the unmapped tasks are assigned. The algorithm of Min-Min scheduling is given by:

- User submits their jobs to the grid scheduler
- The scheduler selects the resource that has the minimum completion time for all jobs
- Scheduler assigns this resource to the job with the overall minimal completion time
- Update the ready time of the resource and the process is repeated until all the jobs are assigned
- Turnaround time and makespan of the scheduler is evaluated

### 3.6. Largest Task First (LTF) Scheduling

This largest task first (Menasce *et al*., 1995) scheduler schedules the task from the task domain which has largest task size. Length comparator is used to identify the task size. The algorithm of LTF scheduling is given by:

- User submits their jobs to the grid scheduler
- The scheduler schedules the jobs based on job's size
- Scheduler assigns resource for first R number of sorted jobs and keeps the remaining jobs in the waiting queue
- After completing each job, scheduler assigns the next job in the waiting queue to the available resource
- Turnaround time and makespan of the scheduler is evaluated

## 4. PROPOSED ALGORITHM-NEW THRESHOLD BASED SCHEDULING (NTBS)

In New-Threshold based scheduling, the scheduler computes the threshold value based on current active grid user's jobs execution time. This threshold value is calculated based on the following Equation (1):

$$T = \frac{1}{N}\sum_{i=1}^{N} E_i \quad (1)$$

where, $E_i$ is the execution time of $i^{th}$ job. This new-threshold based scheduler arranges the grid user jobs based on this threshold value. It organizes the user jobs in a fair manner whose execution time is below this threshold value. Hence the fairness of this scheduler is as better than EDF scheduler. Then it assigns the resources fairly to scheduled user jobs. This the jobs which are not in queue are assigned with set of fast processor to complete the job fast. The algorithm of NTBS is given by:

- User submits their jobs to the grid scheduler
- The scheduler estimates the threshold T
- Scheduler schedules the user jobs in a first come first serve manner if the user jobs execution time is less than this T
- Scheduler assigns resource for first R number of sorted jobs and keeps the remaining jobs in the waiting queue. If the all resources are not used, then scheduler assigns the free fast processor resource to the jobs whose execution time is above T
- Turnaround time and makespan of the scheduler is evaluated and compared with other scheduling algorithms

## 5. SIMULATION DESIGN

We have used Grid simulator (Gridsim) for simulation. Gridsim (Buyya and Murshed, 2002) is a simulation tool which is used to simulate Grid environment. Gridsim based simulations contain entities for the users resources, information service, statics and network-based I/O. Job represents user's application. Jobs are described with additional information like job deadline, the maximum time limit for execution, necessary machine architecture. In our grid simulation we have taken MetaCentrum data set. This data set contains 103656 jobs with execution time. Number of users is randomly selected and performs different scheduling algorithms based on resource availability. Each result presented is the average value that is derived from 5 simulation experiments with different seeds of random numbers. The scheduling algorithms performed in our papers are FCFS, EDF, Easy back filling, FPLTF, MIN-MIN, LTF and NTBS.

### 5.1. Performance Metrics

To evaluate the performance of the scheduling algorithm, we use the following performance metrics.

#### 5.1.1. Turnaround Time

Let the total number of jobs be N, the completion time for job $j_i$ be $c_i$ and job arrival time is denoted by $a_i$. The turnaround time is defined as Equation (2):

$$\text{Turnaround time} = \frac{1}{N}\sum_{i=1}^{N}(c_i - a_i) \quad (2)$$

#### 5.1.2. Makespan

Makespan is defined as the time spent from the beginning of the first task to the end of the last task in the schedule. It assumes that the jobs are ready at time zero and resources are continuously available during the whole scheduling. Then the makespan is obtained by Equation (3):

$$\text{Makespan } C_{max} = \max\left\{C_1', C_2', ..., C_n'\right\} \quad (3)$$

where, $C`_i$ is the completion of task i. Lesser the makespan means more efficient is the the algorithm, i.e., less time is taken to execute the algorithm. Simulation parameters of our work are shown in **Table 1**.

## 6. RESULTS AND DISCUSSION

The simulation results are shown in **Fig. 2-5**. **Figure 2** describes the simulation results of Turnaround Time performance of the FCFS, EDF, EASY backfilling, FPLTF, Min-Min, LTF and NTBS. This performance is simulated under random job failure condition.

It is observed from the **Fig. 2** that the turnaround time of FCFS scheduler is found to be high because it assign resource based on first come first serve basis without considering resource capability of executing the assigned job in time. The EDF scheduler schedules the jobs based on execution time in ascending order and assigns suitable resource to complete the jobs. Hence the turnaround time of this scheduler is found to be low as compared to FCFS. When the number of grid user jobs increases, the turnaround time of this scheduler slightly increases. New-Threshold based scheduler turnaround time performance is found to be improved and it is comparatively low when the number of jobs increases. The performance of Min-Min, FPLTF and LTF is good when the available resource is closer to number of jobs and it decays under resource constraint situation. In our simulation the availability of cluster of resources are considered as fourteen.

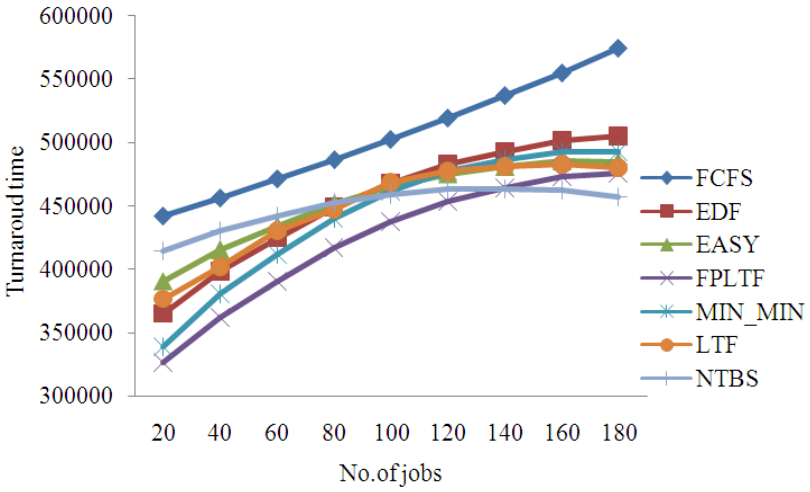


**Fig. 2.** Turnaround Time of different scheduler with failure

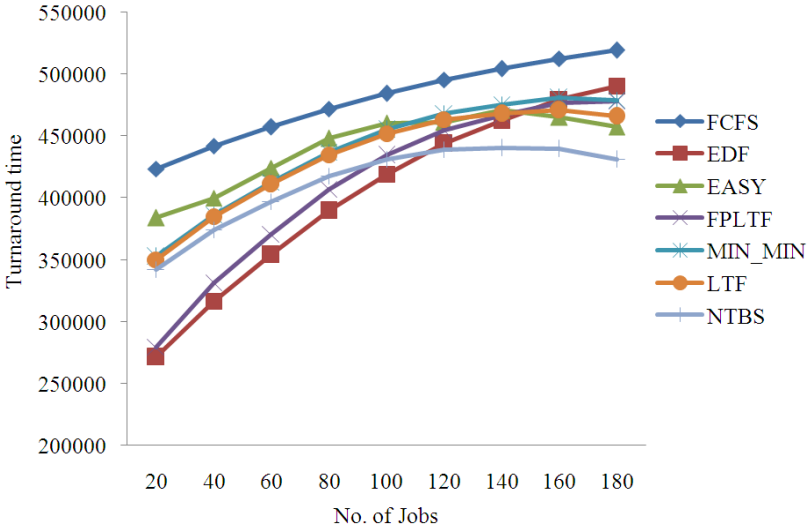


**Fig. 3.** Turnaround Time of different scheduler without failure

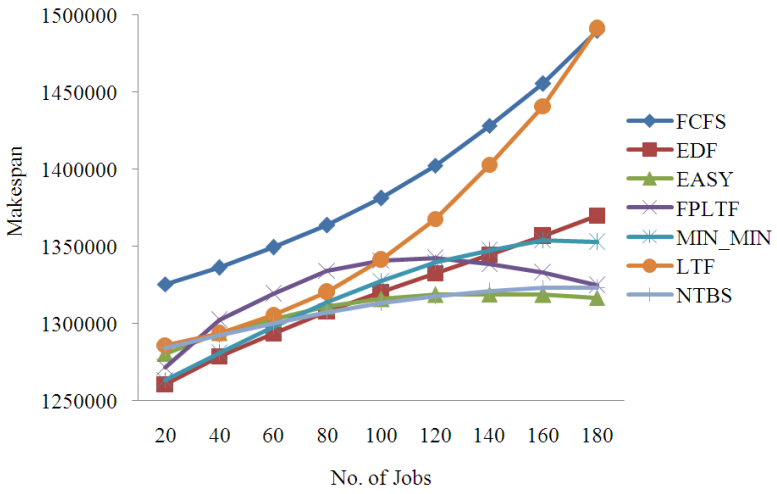


**Fig. 4.** Makespan of different scheduler with failure

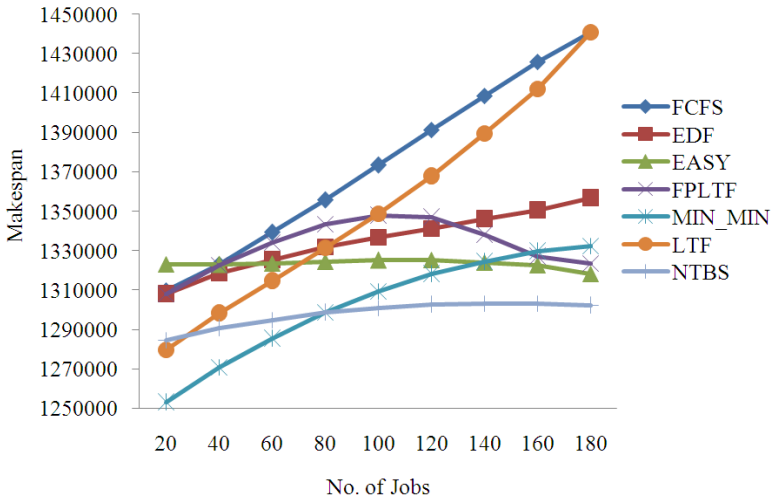


**Fig. 5.** Makespan of different scheduler without failure

**Table 1.** Simulation parameters

| Parameter | Values |
|---|---|
| Number of jobs N | 20 to 180 |
| Number of resources | 14 |
| Job workload | 50-1000 (billion instruction) |
| Node processing speed | 20-200 MIPS |
| Network bandwidth | 2-8 (Mbps) |

**Figure 3** illustrate the Turnaround Time simulation performance of various schedulers without job failure. From this figure, it is observed that the performance of NTBS is good as compared to FCFS, Easy, Min-Min and LTF and it is poor as compared to EDF, FPLTF when the number of jobs closer to number of resources available. Also it is observed from this figure that NTBS turnaround time performance is found improved as compared to all scheduling algorithms when the number of available resource becomes constraint.

**Figure 4** gives the makespan performance comparison of NTBS with different schedulers. From this figure, it is observed that the makespan of new-threshold based scheduler is found to be improved as compared to FCFS, EDF, LTF, Min-Min and FPLTF. All the scheduler gives similar performance, when the number of grid jobs closer to number of available grid resources. The makespan performance of Easy backfilling scheduler and NTBS scheduler are found to

be enhanced when the number of jobs exceeds the 80, i.e., the available resource becomes less than 17%.

**Figure 5** gives the makespan performance comparison of NTBS with different schedulers without job failure condition. From **Fig. 5**, it is observed that the makespan of new-threshold based scheduler is found to be always good as compared to FCFS, EDF, Easy backfilling, LTF and FPLTF. The makespan performance of NTBS scheduler is found to be enhanced by 2% when the available resource becomes less than 17% of the number of resources.

## 7. CONCLUSION

In this study we have proposed new-threshold based scheduling for grid system. The simulation was carried out in grid simulator with MetaCentrum data set. The makespan performance of this scheduler is compared with FCFS, EDF, Easy backfilling, LTF, Min-Min and FPLTF scheduling. NTBS makespan performance is found improved by 2% as compared to EDF and easy backfilling scheduling, 2.89% as compared to FPLTF scheduling, 3.1% as compared to Min-Min scheduling and 4.3% as compared to FCFS and LTF.

## 8. ACKNOWLEDGMENT

The researcher thanks the management of SSN College of Engineering, Chennai, India, for funding the High Performance Computing Lab (HPC Lab) where this research was carried out. The author expresses the sincere thanks to Prof. Dr. Chandrabose Aravindan for his encouragement and valuable guidance.

## 9. REFERENCES

Abramson, D., R. Sosic, J. Giddy and B. Hall, 1995. Nimrod: A tool for performing parametrised simulations using distributed workstations. Proceedings of the 4th IEEE International Symposium on High Performance Distributed Computing, Aug. 2-4, IEEE Xplore Press, Washington, DC., pp: 112-121. DOI: 10.1109/HPDC.1995.518701

Abrishami, S., M. Naghibzadeh and Dick H.J. Epema, 2012. Cost-driven scheduling of grid workflows using partial critical paths. IEEE Trans. Parallel Distrib. Syst., 23: 1400-1414. DOI: 10.1109/TPDS.2011.303

Ahmad, I., Y.K. Kwok, M.Y. Wu and K. Li, 2004. Experimental performance evaluation of job scheduling and processor allocation algorithms for grid computing on metacomputers. Proceedings of the 18th International Symposium Parallel Distributed Processing, Apr. 26-30, IEEE Xplore Press, pp: 170-177.

Bajaj, R. and D.P. Agrawal, 2004. Improving scheduling of tasks in a heterogeneous environment. IEEE Trans. Parallel Distrib. Syst., 15: 107-118. DOI: 10.1109/TPDS.2004.1264795

Buyya, R. and M. Murshed, 2002. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. J. Concurrency Comput. Pract. Exp., 14: 1175-1220. DOI: 10.1002/cpe.710

Buyya, R., D. Abramson and J. Giddy, 2000. Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid. Proceedings of the 4th International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region, May 14-17, IEEE Xplore Press, Beijing, China, pp: 283-289. DOI: 10.1109/HPC.2000.846563

Buyya, R., D. Abramson, J. Giddy and H. Stockinger, 2002. Economic models for resource management and scheduling in Grid computing, J. Concurrency Comput. Practice Exp., 14: 1507-1542. DOI: 10.1002/cpe.690

Daoud, M.I. and N. Kharma, 2008. A high performance algorithm for static task scheduling in heterogeneous distributed computing systems. J. Parallel Distrib. Comput., 68: 399-409. DOI: 10.1016/j.jpdc.2007.05.015

Doulamis, N.D., A.D. Doulamis, E.A. Varvarigos and T.A. Varvarigou, 2007. Fair scheduling algorithms in grids. IEEE Trans. Parallel Distrib. Syst., 18: 1630-1648. DOI: 10.1109/TPDS.2007.1053

Khoo, B.T.B., B. Veeravalli, T. Hung and C.W.S. See, 2007. A multi-dimensional scheduling scheme in a Grid computing environment. J. Parallel Distrib. Comput., 67: 659-673. DOI: 10.1016/j.jpdc.2007.01.008

Kwok, Y.K. and I. Ahmad, 1999. Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput. Surveys, 31: 406-471. DOI: 10.1145/344588.344618

Menasce, D., D. Saha, S.C.D. Porto, V.A.F. Almeida and S.K. Tripathi, 1995. Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures. J. Parallel Distrib. Comput., 28: 1-18. DOI: 10.1006/jpdc.1995.1085

Ramesh, D. and A. Krishnan, 2012. An optimal load sharing technique for grid computing. Am. J. Applied Sci., 9: 1101-1106. DOI: 10.3844/ajassp.2012.1101.1106

Silva, D.P.D., W. Cirne and F.V. Brasileiro, 2003. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. Proceedings of the International Euro-Par Conference Klagenfurt, Aug. 26-29, Springer Berlin Heidelberg, Klagenfurt, Austria, pp: 169-180. DOI: 10.1007/978-3-540-45209-6_26

Song, S., Y.K. Kwok and K. Hwang, 2006. Risk-resilient heuristics and genetic algorithms for security as sured grid job scheduling. IEEE Trans. Comput., 55: 703-719. DOI: 10.1109/TC.2006.89

Topcuoglu, H., S. Hariri and M. Wu, 2002. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst., 13: 260-274. DOI: 10.1109/71.993206

Wong, A.K.L. and A.M. Goscinski, 2007. Evaluating the easy-backfill job scheduling of static workloads on clusters. Proceedings of the IEEE International Conference Cluster Computing, Sep. 17-20, pp: 64-73. DOI: 10.1109/CLUSTR.2007.4629218

Yu, J. and R. Buyya, 2005. A Taxonomy of workflow management systems for grid computing. J. Grid Comput., 3: 171-200. DOI: 10.1007/s10723-005-9010-8