Research Journal of

Information Technology

ISSN 1815-7432

Academic Journals Inc.

www.academicjournals.com

Research Journal of Information Technology 6 (4): 379-388, 2014
ISSN 1815-7432 / DOI: 10.3923/rjit.2014.379.388
© 2014 Academic Journals Inc.

An Effective Hybrid Scheduling for Checkpoint Based Grid Environment

L. Rama Parvathy
Department of Computer Applications, SSN College of Engineering, Kalavakkam, Chennai, Tamil Nadu, India

ABSTRACT

The main roles of utility grids are to carry out the user applications dynamically. To meet the users' quality requirement, scheduling the user jobs to proper resource in existence of fault, is the one of the challenge in these grids. Providing fault tolerance is as important as scheduling to minimize the workflow execution time and makespan. In this study, we propose an Effective Hybrid Scheduling (EHS) algorithm for checkpoint based grid systems. It combines the best features of various scheduling and checkpointing. It works under two phases namely training phase and deployment phase. During training phase, EHS runs various schedulers for bags of task and build the model based on their performance. In the deployment phase, this algorithm uses the model developed during training phase for proper resource allocation. From the simulation results, it is found that EHS always gives better performance in terms of reduced makespan and consistent turnaround time with improved reliability.

Key words: Effective hybrid scheduling, makespan, utility grid and turnaround time

INTRODUCTION

Grid computing effectively uses the heterogeneous resources to get rid of computational problems. It has the utmost concern in handling the enormous data produced by parallel applications. Feasibility of multi-node failures gets increased as the number of computer nodes increases. In order to reduce the multi-node failures, a checkpoint is needed that ends up in performance degradation. In case of any failure, the job will be passed on to some other computational node and begin the execution from previously stored checkpoint. One of the main issues in distributed environment is to provide fault tolerance, while optimizing execution time. Research efforts in design and implementation of fault detection services (Hwang and Kesselman, 2003; Subbiah and Blough, 2004) have been dedicated to fault tolerance. Failure prediction (Derbal, 2006) and recovery schemes (Dogan and Ozguner, 2002; Da Silva *et al.*, 2003) were analyzed. In the failure situation, job checkpointing in combination with migration, improve system performance. Their effectiveness mostly depends on optimizing runtime parameters such as the checkpointing interval and the number of replicas (Oliner *et al.*, 2005; Bossie and Fiorini, 2006). Determining optimal values for these parameters, requires good knowledge of the distributed system. Most of these papers address the issues of reliability. The effectiveness in combining with scheduling was not well addressed in those studies. As workflow job scheduling is a known problem, many methods have been proposed for homogeneous (Kwok and Ahmad, 1999) and heterogeneous distributed grid systems (Topcuoglu *et al.*, 2002; Bajaj and Agrawal, 2004; Daoud and Kharma, 2008). In this study, an effective hybrid scheduling for checkpoint based grid environment is presented that provide suitable scheduler selection to reduce the runtime without compromising reliability.

The proposed hybrid scheduling algorithm for checkpoint based grid system reduces the makespan and turnaround time as compared to the performance of checkpoint based grid system technique without scheduling algorithms. This algorithm works under two phases. They are training phase and deployment phase. In training phase, EHS runs various scheduling algorithms with job retry and checkpoint mechanism. It bulid a model based on their performance. In deployment phase, it suitably allocates proper resources according to model developed in training phase. The hybrid scheduler combines the advantages of New Threshold Based Scheduling (NTBS), Easy backfilling and Fast Processor to Largest Task First (FPLTF).

SYSTEM MODEL

As shown in the Fig. 1, the grid model consists of 4 components. They are grid information system, grid resources, grid user and grid job-scheduler. For resource allocation, the user sends request to the scheduler. The GIS registers the availability of all grid resources. Depending on the proposed algorithm, the job-scheduler chooses the most perfect resources to execute the job. The user receives the processed results at the end. The resources differ from each other in system processing speed, processing element ids, scheduling policy etc. The user jobs also differs in arrival time, execution time, deadline, etc. The job failure model follows exponential distribution. In this model, the risks between grid jobs and resources are considered. Job failure is generated randomly. Job checkpointing is an intellectual scheme adopted for inserting fault tolerance in computational grids. In case of any failure, the job will migrate to another computational node and starts the execution from the previously stored checkpoint. For reliable computation, any computational node can utilize one or more multiple fault-tolerance mechanisms.

CHECKPOINT SERVER AND GRID SCHEDULERS

Checkpoint server: The main functions of checkpoint server is described below:

- It receives resource failure history from GIS
- It prepares job checkpoints
- It dispatches jobs with checkpoints to grid resource

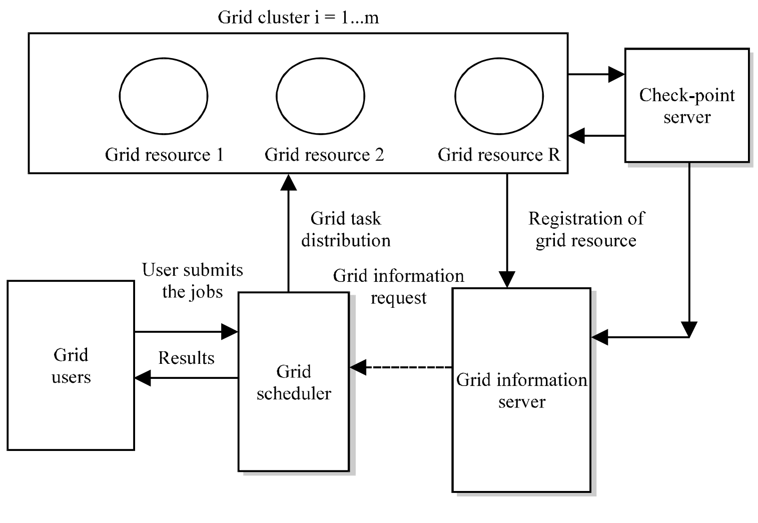

Fig. 1: Checkpoint based grid system model

- Grid resource sends the checkpoint status to checkpoint server
- In case of any failure, it is reported to GIS by checkpoint server
- GIS resubmit the job to grid scheduler to assign new resource along with last checkpoint status

Grid schedulers: Grid scheduler plays a vital role among the components of checkpoint based grid system. The scheduler collects the job information like job number, job type and job size from the users. Users surrender its QoS requirements of each job that comprises the deadline to complete its execution, the number and type of required resources. Scheduler performance is to detect and sort the appropriate resources that satisfy the user QoS requirements and job execution. Grid scheduler joins with Grid Information Server (GIS) to gather the information of available grid resources for job execution. In general, a scheduling system of grid computing environments aims to minimize the execution time (Abramson *et al.*, 1995) and fulfilling economic constraints (Buyya *et al.*, 2000) without affecting resource utilization. In this study, we have proposed an effective hybrid scheduling to minimize the execution time and turnaround time without affecting reliability. When checkpointing is done that takes an extra run time. By combining proper scheduler for checkpoint based grid system, this extra runtime is compensated. The makespan and turnaround time performance of EHS is compared with various scheduling algorithms. In our model, N is the total number of user jobs to be executed which are randomly generated and R is the total number of resource available chosen from various cluster for executing jobs. The various existing scheduling algorithms (Ramaparvathy, 2014) described below are modeled along with our proposed algorithms for performance comparison.

First Come First Served (FCFS) scheduling: In FCFS (Ahmad *et al.*, 2004; Ramaparvathy, 2014) scheduling algorithm, the resource is assigned based on queue order to maintain the fairness among grid users. If the number of job is greater than available resource, FCFS schedules the jobs based on its arrival. Then it assigns resource for first R number of jobs and keeps the remaining jobs in the waiting queue.

Early Deadline First (EDF) scheduling: EDF scheduling algorithms (Doulamis *et al.*, 2007; Ramaparvathy, 2014) 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.

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; Ramaparvathy, 2014). 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.

Fastest Processor to Largest Task First (FPLTF) scheduling: Fastest processor to largest task first (Da Silva *et al.*, 2003; Ramaparvathy, 2014) 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.

Min-Min (Min-Min) scheduling: Min-Min algorithm (Song *et al.*, 2006; Ramaparvathy, 2014) starts with a set of all unmapped tasks. 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.

Largest Task First (LTF) scheduling: This largest task first (Menasce *et al.*, 1995; Ramaparvathy, 2014) scheduler schedule the task from the task domain which has largest task size. Length comparator is used to identify the task size.

New Threshold Based Scheduling (NTBS): In New-Threshold based scheduling (Ramaparvathy, 2014), 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:

$$T = \frac{1}{N}\sum_{i=1}^{N} E_i \tag{1}$$

where, E_i is the execution time of ith 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. It keeps the other jobs in weighting queue in fair manner. Hence, the fairness of this scheduler is as better than EDF scheduler. Then it assigns the resources fairly to scheduled user jobs first and then it assigns resources for the remaining user jobs those are in waiting queue fairly.

PROPOSED ALGORITHM: AN EFFECTIVE HYBRID SCHEDULING (EHS)

The basic aim of computational grids is to carry out the user applications or jobs. For this reason, grid scheduler receives the user's jobs with their QoS requirements and then it assigns each job to proper resource. Depending upon the response time, the present scheduling systems select resources for execution of the jobs. When a grid resource is not able to finish its job within a specified duration, a fault occurs. In case of failure of the grid resource at the time of execution, the job is rescheduled to some other resource in which the execution of the job begins from scratch. This method is described as Job Retry mechanism. This method consumes more time to complete the job than expected, which, in turn, fails to satisfy the user's QoS requirements. The job checkpointing mechanism is ultimately used to explain this problem. While checkpointing mechanism is adopted, the partially completed job can be retrieved from the last checkpoint stored and thereby eliminates to begin the job from scratch. The main prospective of checkpointing is to enhance the performance of the grid in the presence of faults. In our simulation, the both job retry and job checkpointing mechanisms are modeled. Generally, the performance of grid system will be better when there is no resource failure. In this condition, depends upon scheduler, the run time performance varies. Hence, choosing the best scheduler for checkpoint based grid model will tend to give better run time response. This is done in our proposed effective hybrid scheduling algorithms. Proposed hybrid scheduler works under two phases named as training phase and deployment phase. In first phase known as training phase, the scheduler runs the various schedulers to build the better performance model. This model is referred in second phase known as deployment phase for proper resource allocation to achieve better performance. Therefore, it adoptively chooses the best scheduler to give better runtime performance. Since, this scheduling algorithm combines the quality of scheduling and checkpointing, it achieves the better runtime performance and better job reliability. The algorithm of EHS is given by:

- User submits their jobs to the grid scheduler
- The scheduler runs various scheduling algorithms with and without checkpointing
- Scheduler stores the computational results in its performance table
- Scheduler assigns resource according to suitable scheduling algorithms based on number of current active user jobs

Turnaround time and makespan of the scheduler is evaluated and compared with other scheduling algorithms.

RESULTS AND DISCUSSION

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, etc. In our grid simulation, we have taken MetaCentrum data set and the failure model follows hyper exponential distribution. In this failure model, the number of resource failure, failure time and its duration are defined randomly. There are 103656 jobs in defined data set. Here, the number of active user jobs is generated randomly. 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 studies are FCFS, EDF, Easy back filling, FPLTF, MIN-MIN, LTF and NTBS and proposed EHS. These scheduler's performances are observed under Job retry and migration with checkpointing condition. One migration node is taken into consideration in our simulation. The following performance metrics (Ramaparvathy, 2014) are used for performance comparison.

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:

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

Makespan: Makespan (Ramaparvathy, 2014) 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:

$$\text{Makespan} = C_{\max} = \max\{C'_1, C'_2, \ldots, C'_n\} \tag{3}$$

where, c'_i is the processing time of task j_i by the resource R_k, lesser the makespan means more efficient is the algorithm i.e., less time is taken to execute the algorithm. The simulation parameters (Ramaparvathy, 2014) are shown in Table 1.

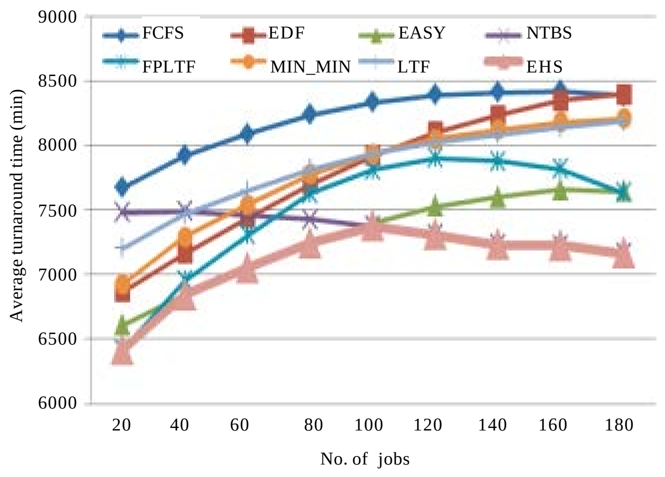

Fig. 2: Turnaround time of EHS scheduler compared to other scheduling with job retry mechanism

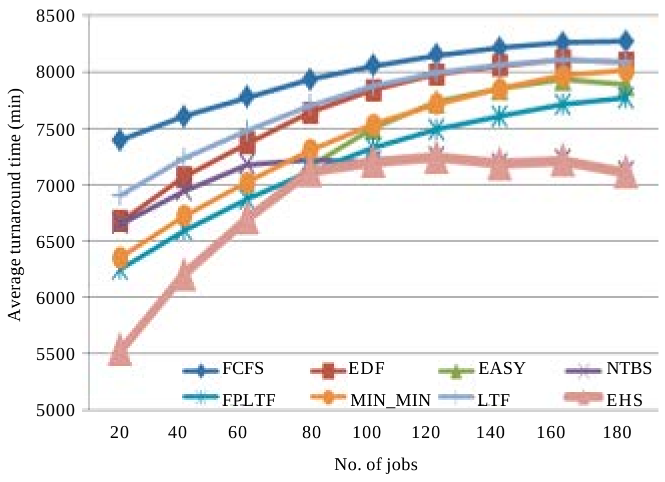

Fig. 3: Turnaround time of EHS with job checkpointing mechanism

Table 1: Simulation parameters

Parameters	Values
No. of jobs (N)	20-180
No. of resources	14
Job workload	50-1000 (billion instruction)
Node processing speed	20-200 (MIPS)
Network bandwidth	2-8 (Mbps)

Figure 2-5 describe the experimental results. Turnaround time performance of the EHS compared with other scheduling algorithms with job retry mechanism is shown in Fig. 2. From the Fig. 2, it is observed that the FCFS scheduler turnaround time is high because it assigns 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 deadline time in ascending

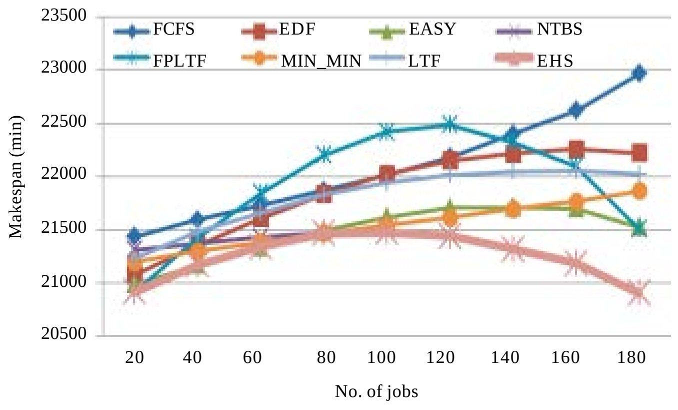

Fig. 4: Makespan of different scheduler with job retry mechanism

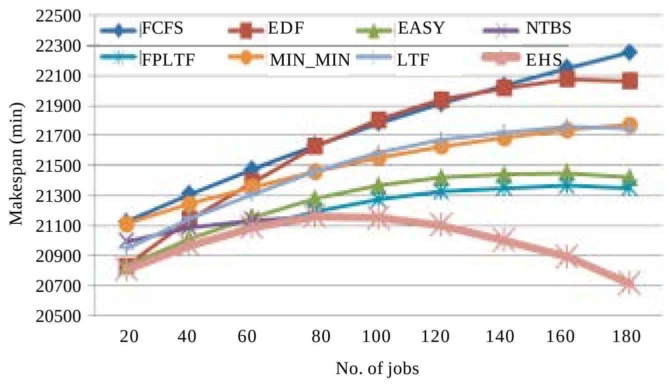

Fig. 5: Makespan of different scheduler with job checkpointing mechanism

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. This scheduler performance decays when the number of user jobs increases.

Also, it is observed from the Fig. 2 that the Min-Min, FPLTF and LTF performance found to be good when the number of active user jobs is nearby the available resource and its performance falloffs when the resource is very less as compared to the user jobs. Fourteen resources of nodes from various clusters are considered in our simulation. The turnaround time performance of EHS scheduler found to be good in all condition because it combines the best features of all these schedulers. The Turnaround Time performance of EHS compared with various schedulers with job checkpointing mechanism is shown in Fig. 3.

From Fig. 3, it is observed that the performance of EHS is good compared to FCFS, Min-Min, Easy, EDF FPLTF and LTF. When the number of jobs is 80, the turnaround time of EHS with job checkpointing is found to be 7000 min, whereas it is 7250 min in EHS with job retry mechanism. This is around 3.5% improvement. The makespan performance of various schedulers with job retries and job checkpointing mechanism is shown in Table 2 and 3, respectively.

From the table it is clearly observed that the EHS scheduler always gives better performance in terms of low makespan. Figure 4 and 5 shows the makespan performance of EHS with job retry mechanism and job checkpointing mechanism, respectively.

Table 2: Makespan performance in minutes of various schedulers with job retry mechanism

	No. of jobs								
Schedulers	20	40	60	80	100	120	140	160	180
FCFS	21435	21596	21731	21870	22018	22180	22400	22623	22970
EDF	21081	21355	21610	21836	22021	22151	22218	22260	22223
EASY	20986	21170	21340	21500	21621	21710	21711	21701	21528
NTBS	21313	21376	21431	21473	21476	21440	21320	21185	20903
FPLTF	20918	21425	21843	22201	22420	22490	22325	22100	21506
MIN_MIN	21196	21296	21383	21466	21543	21616	21695	21770	21863
LTF	21225	21481	21663	21830	21943	22018	22048	22061	22025
EHS	20918	21170	21340	21466	21476	21440	21320	21185	20903

Table 3: Makespan performance in minutes of various schedulers with job checkpointing mechanism

	No. of jobs								
Schedulers	20	40	60	80	100	120	140	160	180
FCFS-CP	21125	21311	21475	21635	21780	21913	22033	22150	22255
EDF-CP	20830	21143	21393	21626	21805	21940	22016	22076	22063
EASY-CP	20830	21016	21155	21280	21366	21423	21443	21453	21421
NTBS-CP	20995	21088	21135	21165	21151	21101	21003	20890	20715
FPLTF-CP	20806	20968	21088	21198	21275	21328	21351	21366	21350
MIN_MIN-CP	21113	21246	21356	21463	21551	21626	21685	21738	21770
LTF-CP	20950	21148	21311	21463	21581	21670	21720	21758	21745
EHS-CP	20806	20968	21088	21165	21151	21101	21003	20890	20715

From Fig. 4, it is observed that the makespan performance of Min-Min algorithm, Easy algorithm and NTBS is found to be improved compared to FCFS and EDF. It is also observed that the NTBS performance is improved when the number of available resource less than 17% of active users. Here, the 14 number of resources are considered. When the number of jobs is 80 and above, NTBS performance is found to be good. Under resource constraint enviroinment, the performance of NTBS is good. In other region, Min-Min and Easy algorithm gives good makespan performance. EHS makespan performance is found good always since it combines the best scheduler.

The makespan performance of various schedulers along with EHS with job chckpointing mechanism is shown in Fig. 5. From Fig. 5, it is observed that the makespan of new-threshold based scheduler is found to be good compared to FCFS, EDF, Easy backfilling, LTF and FPLTF when the number of jobs is 60 and above. FPLTF scheduler performance is found to good when the numbers of active jobs are 60 and less. The makespan performance of EHS scheduler is found to be enhanced by 0.5% when the number of jobs are 60 and less. When it is above 60, the performance of EHS is improved by 9%.

CONCLUSION

Scheduling and job migration is combined in our proposed effective hybrid scheduling to achieve better runtime performance with improved job reliability. The EHS scheduling with job retry mechanism and job checkpointing mechanism is implemented using GridSim and the runtime performance like turnaround time and makespan is computed for MetaCentrum data set. The makespan performance of EHS scheduler is compared with various scheduling algorithms. It is found that the proposed EHS achieves 9% improvement compared to other scheduling algorithms.

ACKNOWLEDGMENT

The author 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.

REFERENCES

Abramson, D., R. Sosic, J. Giddy and B. Hall, 1995. Nimrod-G, A tool for performing parametised simulations using distributed workstations. Proceedings of the 4th Symposium on High Performance Distributed Computing, August 2-4, 1995, Washington, DC., USA., pp: 112-121.

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 IEEE 18th International Parallel and Distributed Processing Symposium, April 26-30, 2004, Santa Fe, New Mexico, USA., pp: 170-177.

Bajaj, R. and D.P. Agrawal, 2004. Improving scheduling of tasks in a heterogeneous environment. IEEE Trans. Parallel Dist. Syst., 15: 107-118.

Bossie, C. and P.M. Fiorini, 2006. On checkpointing and heavy-tails in unreliable computing environments. SIGMETRICS Perform. Eval. Rev., 34: 13-15.

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

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 and Exhibition on High Performance Computing in Asia-Pacific Region, May 14-17, 2000, Beijing, China, pp: 1-7.

Da Silva, D.P., 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 9th International Euro-Par Conference, August 26-29, 2003, Klagenfurt, Austria, pp: 169-180.

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.

Derbal, Y., 2006. A new fault-tolerance framework for grid computing. Multiagent Grid Syst., 2: 115-133.

Dogan, A. and F. Ozguner, 2002. Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing. IEEE Trans. Parallel Distrib. Syst., 13: 308-323.

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.

Hwang, S. and C. Kesselman, 2003. A flexible framework for fault tolerance in the grid. J. Grid Comput., 1: 251-272.

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

Menasce, D.A., 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.

Oliner, A.J., R.K. Sahoo, J.E. Moreira and M. Gupta, 2005. Performance implications of periodic checkpointing on large-scale cluster systems. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, April 4-8, 2005, Denver, CO., USA.

Ramaparvathy, L., 2014. A new threshold based job scheduling for grid system. J. Comput. Sci., 10: 1069-1076.

Song, S., K. Hwang and Y.K. Kwok, 2006. Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling. IEEE Trans. Comput., 55: 703-719.

Subbiah, A. and D.M. Blough, 2004. Distributed diagnosis in dynamicfault environments. Parallel Distrib. Syst., 59: 453-467.

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

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 on Cluster Computing, September 17-20, 2007, Austin, TX., pp: 64-73.