A New Proactive Fault Tolerant Approach for Scheduling in Computational Grid

P.Keerthika,
Assistant Professor(Sr.G),
Kongu Engineering College,
Perundurai, Erode,
Tamilnadu.

Dr.N.Kasthuri,
Professor,
Kongu Engineering College,
Perundurai, Erode,
Tamilnadu.

ABSTRACT
Grid Computing provides non-trivial services to users and aggregates the power of widely distributed resources. Computational grids solve large scale scientific problems using distributed heterogeneous resources. The Grid Scheduler must select proper resources for executing the tasks with less response time and without missing the deadline. There are various reasons such as network failure, overloaded resource conditions, or non-availability of required software components for execution failure. Thus, fault-tolerant systems should be able to identify and handle failures and support reliable execution in the presence of failures. Hence the integration of fault tolerance measures with scheduling gains much importance. In this paper, a new fault tolerance based scheduling approach for scheduling statically available meta tasks is proposed wherein failure rate and the fitness value are calculated. The performance of the fault tolerant scheduling policy is compared with a non-fault tolerant scheduling policy and the results shows that the proposed policy performs better with less TTR in the presence of failures. The number of tasks successfully completed is also more when compared to the non-fault tolerant scheduling policy.

Keywords
Fault tolerance, Failure rate, Grid scheduling, Meta task

1. INTRODUCTION
Grid computing is sharing of coordinated resources in a dynamic environment where multi-institutional virtual organization involves and open standards becomes the key underpinning. In grid environments they does not prefer to rely on centralized control; instead they provide coordination among the resources. The use of open standards, protocols and frameworks provides interoperability facilities. To achieve the full potential of grid environment we should perform the grid scheduling in an effective manner.
Grid scheduling is the process of making scheduling decisions involving resources over multiple administrative domains. This process can include searching multiple administrative domains to use a single machine or scheduling a single job to use multiple resources at a single site or multiple sites.
Job scheduling involves mapping of ‘n’ tasks to ‘m’ processors. It is a NP- complete problem. Scheduling is done by using a software application called scheduler. Scheduler software enables an enterprise to schedule and, in some cases, monitor computer "batch" tasks. It can initiate and manage jobs automatically by processing prepared task control language statements or through equivalent interaction with a human operator.
When a task is considered, the key parameters includes deadline, memory space required, waiting time, process time, turn-around time etc. Similarly the key parameters for a resource include speed, failure rate, maximum load it can handle, queue length etc. In this paper we try to find out the common parameters that are being shared by both task as well as resource like memory space, speed to filter out the capable tasks. With the above assumptions we perform the scheduling through time to release and failure rate values.
Fault tolerant mechanisms are needed to hide the occurrence of faults, or the sudden unavailability of resources. Although scheduling and fault tolerance have been traditionally considered independently from each other, there is a strong correlation between them. As a matter of fact, each time a fault-tolerance action must be performed. Fault-tolerant schedulers [19,20] attempt to do so by integrating scheduling and fault management, in order to properly schedule both faulty and non-faulty tasks.
The consideration of TTR value is because of its improved efficiency over Min-Min and FCFS algorithms. The addition of transmission time in scheduling criteria enables the sight over the transmission cost of data’s or packets where the actual grid resources being distributed in nature. When this is being integrated with fault tolerant measures then the reliability of the algorithm would increase. In the proposed algorithm the above is achieved in efficient way with the fitness value which is calculated and considered while scheduling when the task can hold with the available specifications of the resource.
The main objective of this paper is to design a new scheduling algorithm that reduces the TTR which is the total time taken to complete a set of jobs. Also, the idle time of the resources should be less which assures that no resources are kept idle for a long time. It also ensures that fault tolerant measures are satisfied. The tasks are scheduled after the fault rate of all the resources is calculated. The proposed algorithm considers both system performance and user satisfaction. Hence, most of the jobs are completed within their expected completion time with minimum number of failures.

2. RELATED WORKS
There are many scheduling algorithms that perform better and some of the algorithms concentrate on fault tolerance. Some of those scheduling algorithms are discussed below.
Minimum Time to Release Scheduling Algorithm has been discussed in [1] in which the Time to Release (TTR) is calculated. Based on the TTR value all the tasks are arranged in descending order. The tasks are submitted in that order. This algorithm performs better when compared to First Come First Serve Scheduling and min-min algorithms.
Other related works includes time and cost optimization algorithms discussed in [3]. It extends the cost-optimisation

algorithm to optimise the time without incurring additional processing expenses. This is accomplished by applying the time-optimisation algorithm to schedule task farming or parameter-sweep application jobs on distributed resources having the same processing cost.
Computational grid environment involves problems in effective scheduling of jobs. [1] brings out a way in solving this problem through grid scheduling architecture and job scheduling algorithm. This architecture is scalable and eliminates control of local site resources. In this algorithm the grid scheduler which selects the computational resources based on job requirements, job characteristics and information given by resources, performs resource brokering and job scheduling.
The purpose of this scheduler is to minimize the total time to release (TTR) for individual applications. The work [1] provides the grid architecture with dispatcher, grid scheduler and load balancer which interface modules, matches resources and clients, reschedules for optimizing resources respectively. Grid scheduling systems obtain user applications through interface application profiling and the services are provided by the resources which differ in number of processors, cost and speed of processing, etc. This model is efficient enough to handle large range of components being integrated into the Grid and supports varied policies of users. This architecture brings out interoperation of different schedulers, which considers user objectives. Grid resources must register with GIS initially.

The grid scheduling algorithm starts with splitting of user requests and checking for available identified specific resources. Then the resources and their requirements are verified with appropriate resources saved for future processing. This algorithm proceeds with calculation of TTR and mapping of jobs based on it. Based on time assignment some mappings are removed and a updated list is made which is sorted. From the list first job is released and assigned to selected resource if possible. This algorithm is repeated till completion of jobs. This proposed model implements grid simulation toolkit software GridSim Toolkit 4.0. Through this a virtual network connection was established among various entities with an exclusive bandwidth between every two entities. On the whole the results obtained seem to be efficient when compared with some benchmark algorithms.

The model proposed in [2] brings out modeling execution of jobs on grid compute clusters with the assistance of PEPA model. It involves approximation of state space and representing it as a set of ordinary differential equations. [3] proposes a new scheduling algorithm called DBC cost time optimization extending the DBC cost optimization algorithm. Based on the user's quality of services requirements, the resources for their applications are allocated, by regulating the supply and demand. This is brought through a framework including economy driven deadline and budget constrained algorithms for satisfying user's requirements. [4] addresses fault-tolerant scheduling for differentiated classes of independent tasks through various simulation experiments. It proposes two algorithms such as MRC-ECT and MCT-LRC which provides optimal backup schedule in terms of replication cost and minimum completion time respectively.

A QoS guided task scheduling algorithm is put forth in [5] which is based on general adaptive scheduling heuristics including QoS guidance. The results of [5] show that general adaptive scheduling heuristics that includes QoS guidance provides significant performance gain. A fault tolerance service based on different types of failures satisfying the QoS requirements is explained in [6]. It also gives a resource scheduling service, detection of faults and over usage of resources and fault management service.

[7] gives an evaluation of coordinated grid scheduling strategy with the FCFS job scheduling policy and the matchmaking approach for the resource selection as a reference. [8] tests and compares coordinated checkpoint and pessimistic message log based on frequency of faults and concludes that an uncoordinated checkpoint is better for large scale cluster computational grids.[9] implements (LA-MPI) for toleration of network related failures and present a distinguished view its features in network transmission. [10] studies various failures in computational grid environment and presents a statistical view on all factors related to failures.

Through effective mechanisms for fault tolerance using over-provisioning algorithms as basic steps and achieving balance reliability in real time constraints has been proposed in [11]. In order to allocate grid tasks in minimum time and to increase toleration of faults, [12] uses DAG mechanism to enter tasks and thereby brings out an efficient algorithm namely Ant Colony Optimization algorithm.[13] approaches scheduling mechanism with dynamic priority for tasks according to the execution time and multi queues are formed.RPC and Proactive failure detection are being used for accurate efficiency.[14] surveys the importance of fault tolerance for achieving reliability by all possible mechanisms such as Replication, Check pointing and job migration.

Overcoming the disadvantages of existing fault tolerance techniques,[15] proposes "Smart Failure" technique for reducing failover and increasing performance.[16] studies various fault occurrences and investigates the reason for faults thereby providing a clear view on faults. [19] proposes a system architecture for Distributed Networks providing a wide area scheduler prototype.[20] uses divide and conquer strategy for overcoming crashes of one or more nodes and concentrates on minimizing the redundant work.

3. PROPOSED FAULT TOLERANT BASED SCHEDULING POLICY

In this section, the brief description of the proposed algorithm is presented. This scheduling algorithm is based on transmission time and fault rate. User deadline is taken into account and the job is made to be executed within the expected deadline by assigning it to the most suitable resource. System performance is also achieved by reducing the idle time of the resources and distributing the unmapped tasks equally among the available resources. TTR value is calculated for all the jobs with every available resource. Secondly, the job with the minimum deadline is considered. The deadline of the selected job given by the user is compared with different TTR values. Then the waiting time of the resource and the TTR are recalculated for the remaining unmapped jobs. Above steps are repeated until all the jobs are scheduled.

3.1 Calculation of TTR.

TTR = TW +TE + TI + TO

TW: Time for waiting in resource queue
TE: Expected time to execute the job

TI: Time taken for transfer of input files and executable to the resource
TO: Time taken for transfer of output files to the user.

Hit Rate = JSucc/Jsub

JSucc : No. of jobs successfully completed
Jsub : Total No. of jobs submitted

Miss Rate = Jf/Jsub

Jf : No. of jobs failed for execution
Jsub : Total No. of jobs submitted

4. ESTIMATION OF FAULT RATE AND FITNESS VALUE

In this algorithm fault rate is considered. Based on fault rate and TTR value, fitness value is calculated.

FR =Jf / Js

where
FR = Fault rate
Jf = Number of jobs failed
Js = Number of jobs submitted

FV = Ff + Ft

where

Ff =((fc – fmin) / 2)
Ft =((tc – tmin) / 2)
FV= Fitness value
fc = current fault-rate
fmin =overall minimum fault-rate
tc = current ttr
tmin = overall minimum ttr

5. SIMULATION RESULTS

The above proposed algorithm is simulated with 512 jobs and 16 machines for 5 different inputs using GridSim5.0 Toolkit. Fault Tolerant Time To Release scheduling algorithm is compared with Time To Release scheduling algorithm on the basis of TTR, hit rate and miss rate. By analyzing various possible TTR values, the performance of the fault tolerant time to release algorithm is found to be better than the Time to Release scheduling algorithm.

CASE	TIME TO RELEASE SCHEDULING ALGORITHM	FAULT TOLERANT TIME TO RELEASE SCHEDULING ALGORITHM
1	69	65
2	61	57
3	66	63
4	65	62
5	63	59

Table 5.1 Comparison based on TTR

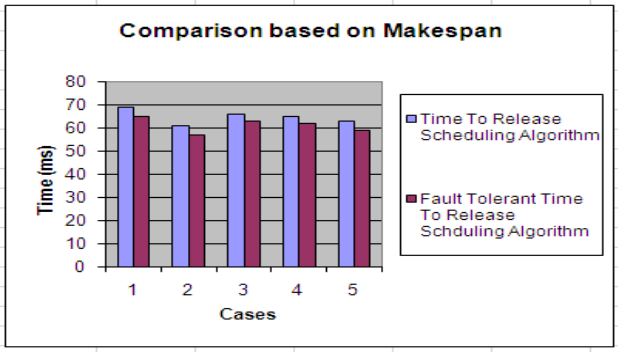

Figure 5.1 Comparison chart based on TTR

From the table 5.1 and figure 5.1, it is inferred that Fault tolerant scheduling has less TTR than scheduling without fault tolerance. TTR is the total time taken to complete a set of jobs.

CASE	TIME TO RELEASE SCHEDULING ALGORITHM	FAULT TOLERANT TIME TO RELEASE SCHEDULING ALGORITHM
1	215	220
2	198	199
3	228	230
4	209	213
5	228	229

Table 5.2 Comparison based on Hit count

From the table 5.2 and figure 5.2, it is inferred that Fault tolerant scheduling has better hit rate than scheduling without fault tolerance. Hit count is calculated as the ration between number of jobs executed successfully to the total number of jobs submitted.

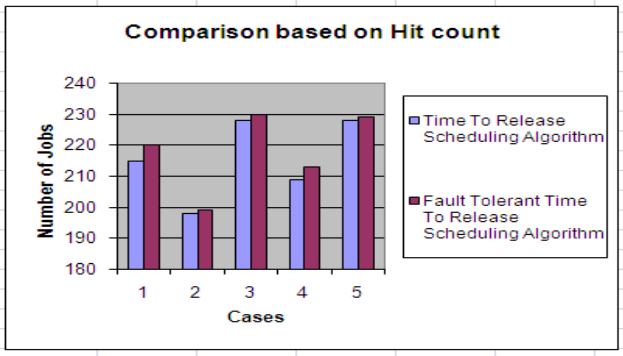

Figure 5.2 Comparison chart based on Hit count

CASE	TIME TO RELEASE SCHEDULING ALGORITHM	FAULT TOLERANT TIME TO RELEASE SCHEDULING ALGORITHM
1	297	292
2	314	313

3	284	282
4	303	299
5	284	283

Table 5.3 Comparison based on Miss count

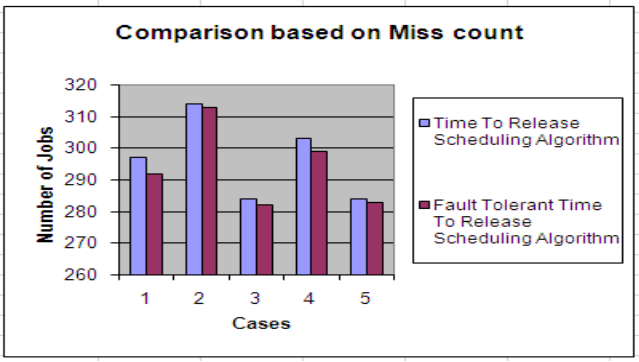

Figure 5.3 Comparison chart based on Miss count

From the table 5.3 and figure 5.3, it is inferred that Fault tolerant scheduling has less miss rate than scheduling without fault tolerance. Miss count is calculated as the ration between number of jobs failed for execution to the total number of jobs submitted.

6. CONCLUSION AND FUTURE WORK

The problem of grid scheduling is addressed in this paper with a solution of providing fault tolerance along with scheduling. The simulation results shows that the proposed scheduling algorithm with fault tolerance shows high hit rate and less miss rate. This approach is successful for static scheduling and it can be extended for dynamic scheduling. The proposed technique is a proactive fault tolerance technique and it can also be merged with passive techniques through which fault tolerance can be achieved to a greater extent.

7. REFERENCES

[1] N.Malarvizhi, Dr.V.Rhymend Uthariaraj. (2009): A Minimum Time To Release Job Scheduling Algorithm in Computational Grid Environment, IEEE Fifth International Joint Conference on INC, IMS, IDC.

[2] Benoit Anne, Cole Murray, Gilmore Stephen and Hillston Jane. (2005): Enhancing the effective utilization of Grid clusters by exploiting on-line performability analysis, IEEE International symposium on Cluster Computing and the Grid (CCGRID), pp. 317-324.

[3] Buyya. R, Murshed. M, Abramson. D. (2002): A deadline and budget constrained cost-time optimization algorithm for Scheduling task farming applications on global grids, In Proceedings of the international conference on parallel and distributed processing techniques and applications, Las Vegas, USA, pp. 24–27.

[4] Q. Zheng, B. Veeravalli, and C. Tham.(2007): Fault-tolerant Scheduling for Differentiated Classes of Tasks with Low Replication Cost in Computational Grids, ACM, HPDC'07, June 25–29, 2007, Monterey, California, USA.

[5] He X , Sun, X., Laszewski, G.V., (2003). Qos guided min-min heuristic for grid task scheduling, Journal of Computer Science and Technology 18, 442-451.

[6] H.Lee, D.Park, M.Hong, Sang-Soo Yeo, SooKyun Kim, SungHoon Kim, (2009): A Resource Management System for Fault Tolerance in Grid Computing, IEEE International Conference on Computational Science and Engineering, DOI 10.1109/CSE.2009.257.

[7] Ivan Rodero, Francesc Guim, Julita Corbalan, 2009, Evaluation of Coordinated Grid Scheduling Strategies, 11th IEEE International Conference on High Performance Computing and Communications, DOI 10.1109/HPCC.2009.28.

[8] A. Bouteiller, P.Lemarinier, G.Krawezik, F.Cappello, Coordinated checkpoint versus message log for fault tolerant MPI, IEEE International Conference on Cluster Computing (Cluster 2003). IEEE CS Press, December 2003.

[9] R. L. Graham, S.-E. Choi, D. J. Daniel, N. N. D. nd Ronald G. Minnich, C. E. Rasmussen, L. D. Risinger, and M. W. Sukalski, "A network failure-tolerant message-passing system for terascale clusters," in International Conference on Supercomputing(ICS'02). New York City, NY, USA: ACM, June 2002, pp. 77–83.

[10] B. Schroeder, and G. Gibson, "A Large Scale Study of Failures in Highperformance-Computing Systems," International Symposium on Dependable Systems and Networks, 2006.

[11] Gopi Kandaswamy, Anirban Mandal, and Daniel A. Reed, "Fault Tolerance and Recovery of Scientific Workflows on Computational Grids"

[12] VahidModiri, Morteza Analoui and Sam Jabbehdari, Fault tolerance in grid using Ant colony optimization and Directed acyclic graph, (2011), International Journal of Grid Computing & Applications (IJGCA) Vol.2, No.1. DOI: 10.5121/ijgca.2011.2102

[13] S.ThamaraiSelvi, Ponsy R.K.SathiaBhama, S.Architha, T.Kaarunya and K.Vinothini, (2010) "Scheduling inVirtualized Grid Environment Using Hybrid Approach" International Journal of Grid Computing & Applications (IJGCA) Vol.1, No.1.

[14] Ritu Garg, Awadhesh Kumar Singh, (2011), "Fault Tolerance in grid computing: state of the art and open issues", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.1. DOI : 10.5121/ijcses.2011.2107

[15] K Limaye, B. Leangsuksun, Z. Greenwood, S. L. Scott, C. Engelmann, R. Libby and K. Chanchio, (2005), "Job-Site Level Fault Tolerance for Cluster and Grid environments" In Proceedings of the IEEE international conference on cluster computing, , pp. 1-9.

[16] R. Medeiros, W. Cirne, F. Brasileiro, J. Sauve, (2003), "Faults in grids: why are they so bad and what can be done about it?" In proceedings of the 4th international workshop, pp 18–24.

[17] J. H. Abawajy,(2004), "Fault-tolerant scheduling policy for grid computing systems", In Proceedings of the International Parallel and Distributed Processing Symposium, IEEE Computer Society, Los Alamitos, United States, pp.3289–3295.

[18] J. Weissman and D. Womack,(1996), Fault Tolerant Scheduling in Distributed Networks. Technical Report TR CS-96-10, Department of Computer Science, University of Texas, San Antonio.

[19] Gosia WrzesinNska, Rob V. van Nieuwpoort, Jason Maassen, Thilo Kielmann, Henri E. Bal, (2006), Fault-

Tolerant Scheduling Of Fine-Grained Tasks In Grid Environments, The International Journal of High Performance Computing Applications,Volume 20, No. 1, Spring 2006, pp. 103–114,DOI: 10.1177/1094342006062528, SAGE Publications.

[20] Meenakshi Bheevgade, Manik Mujumdar, Dr. Rajendra Patrikar, Latesh Malik, (2008), Achieving Fault Tolerance in Grid Computing System, Proceedings of 2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008).

[21] Sameer Singh Chauhan, R. C. Joshi, (2010), QoS Guided Heuristic Algorithms for Grid Task Scheduling, International Journal of Computer Applications (0975 – 8887),Volume 2 – No.9

[22] Leyli Mohammad Khanli, Maryam Etminan Far, Ali Ghaffari , (2010), Reliable Job Scheduler using RFOH in Grid Computing , Journal of Emerging Trends in Computing and Information Sciences.