American Journal of Applied Sciences, 2012, 9 (12), 2046-2051
ISSN: 1546-9239

doi:10.3844/ajassp.2012.2046.2051 Published Online 9 (12) 2012 (http://www.thescipub.com/ajas.toc)

An Efficient Fault Tolerant Scheduling Approach for Computational Grid

[1]P. Keerthika and [2]N. Kasthuri

[1]Department of Computer Science and Engineering,
Faculty of Computer Science and Engineering,
[2]Faculty of Electronics and Communication Engineering,
Kongu Engineering College, Perundurai-638052, Erode, Tamilnadu, India

Received 2012-06-26, Revised 2012-09-03; Accepted 2012-09-03

ABSTRACT

Grid computing serves as an important technology to facilitate distributed computation computational grids solve large scale scientific problems using heterogeneous geographically distributed resources. Problems like dispatching and scheduling of tasks are considered as major issues in computational grid environment. The Grid Scheduler must select proper resources for executing the tasks with less response time. 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 and communication time with scheduling gains much importance. In this study, a new fault tolerance based scheduling approach Fault Tolerant Min-Min (FTMM) 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 min-min scheduling policy using GridSim and the results shows that the proposed policy performs better with less makespan in the presence of failures. The number of tasks successfully completed is also more when compared to the non-fault tolerant min-min scheduling policy. Thus the proposed FTMM algorithm not only achieves better hit rate but also improved makespan.

Keywords: Fault Tolerance, Communication Time, Min-Min, 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. The 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. Similarly the key parameters for a resource include speed, failure rate, maximum load it can handle, queue length. In this study we try to find out the common parameters that are being

Corresponding Author: Keerthika, P., Department of Computer Science and Engineering, Faculty of Computer Science and Engineering, Kongu Engineering College, Perundurai-638052, Erode, Tamilnadu, India Tel: +91(0)4294 226560

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 attempt to do so by integrating scheduling and fault management, in order to properly schedule both faulty and non-faulty tasks.

The consideration of makespan value is because of its improved efficiency over Min-Min algorithm. 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 study is to design a new scheduling algorithm that reduces the makespan 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.

1.1. 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 by Malarvizhi and Uthariaraj (2009) 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. This 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.

Other related works includes fault tolerant algorithms discussed by Garg and Singh (2011) surveys the importance of fault tolerance for achieving reliability by all possible mechanisms such as Replication, Check pointing and job migration.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.

The model proposed by Anne *et al*. (2005) 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. Based on the user's quality of services requirements, the resources for t heir 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. Zheng *et al*. (2007) 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 by He *et al.* (2003) which is based on general adaptive scheduling heuristics including QoS guidance. The results 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 by Lee *et al*. (2009). It also gives a resource scheduling service, detection of faults and over usage of resources and fault management service.

Suresh and Balasubramanie (2012) proposes a static heuristic approach for scheduling independent tasks in grid environment. The requirements of tasks are necessary to identify resources such as computational nodes and data resources. The proposed scheduling algorithm considers both system and application aspects i.e., the factors to improve the performance and utilization of the resources and throughput.

Rodero *et al*. (2009) 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. In order to allocate grid tasks in minimum time and to increase toleration of faults, Modiri *et al*. (2011) uses DAG mechanism to enter tasks and thereby brings out an efficient algorithm namely ant colony optimization algorithm. Garg and Singh (2011) surveys the importance of fault tolerance for achieving reliability by all possible mechanisms such as Replication, Check pointing and job migration.

Nska *et al*. (2006) proposes system architecture for Distributed Networks providing a wide area scheduler prototype and uses divide and conquer strategy for overcoming crashes of one or more nodes and concentrates on minimizing the redundant work.

2. MATERIALS AND METHODS

2.1. Problem Formulation

The problem of job scheduling with heterogeneous distributed resources is discussed which follows the grid scheduling model explained in **Fig. 1** below.

A centralized broker is the single point for the whole infrastructure and manages directly the resource manager interfaces that interact directly with the local resource managers. All the users submit the tasks to the centralized broker. Each resource differs from other resources by many ways that includes number of processing elements, processing speed, internal scheduling policy and its load factor. Similarly each job differs from other jobs by execution time, deadline, time zone.

The static mapping of meta tasks is done in which each machine executes one task at a time. It is assumed that the size of the meta tasks, number of resources, expected execution time of each task in each machine are known priori. An ETC matrix (Expected Tim e to Compute) is constructed using the EET which is the estimated execution time of task i on resource j. The experimental results are based on Braun *et al*. (2001) wherein the scheduling problem is defined by:

- A number of independent tasks to be allocated to the available grid resources
- Number of resources is available to participate in the allocation of tasks
- Workload of each task (MI)
- Computing capacity of each resource
- (MIPS)
- RT(R_j) represents the ready time of the resource after completing the previously assigned jobs

2.2. Proposed FTMM Algorithm

The brief description of the proposed FTMM algorithm is presented.

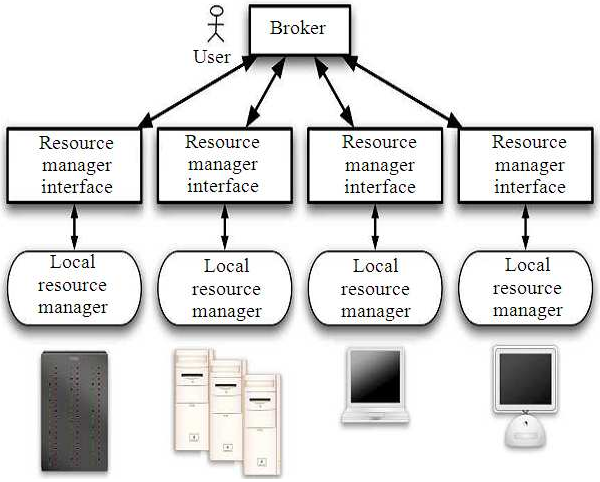

Fig. 1. Basic grid scheduling model

This scheduling algorithm is based on transmission time and fault rate. System performance is also achieved by reducing the idle time of the resources and distributing the unmapped tasks equally among the available resources.

Step 1: Construct ETC(T_i,R_j) matrix of size m × n where m represents the number of tasks and n represents the number of resources involved.
Step 2: Construct RT(R_j) matrix of size 1× n.
Step 3: For each task T_i in the queue and
For each resource R_j where j∈ n
Step 3.1: Construct CT(T_i,R_j) matrix of size m × n which is the completion time of each task i ∈ m on each resource j∈ n and it is given by:

$$CT(T_i,R_j) = ETC(T_i,R_j) + RT(R_j)$$

Step 3.2: Construct CMT(T_i,R_j) matrix of size m × n which is the communication time of each task i ∈ m on each resource j ∈ n and is given by:

$$CMT(T_i,R_j) = ipt(T_i,R_j) + opt(T_i,R_j)$$

where,
ipt(T_i,R_j) = Time taken by T_i for transfer of input files to the resource R_j
opt(T_i,R_j) = Time taken by T_i for transfer of output files to the user from the resource R_j

Step 3.3: Construct TCT(T_i,R_j) matrix of size m × n which is the Total Completion Time of each task i ∈ m on each resource j ∈ n and is given by:

$$TCT(T_i,R_j) = CT(T_i,R_j) + CMT(T_i,R_j)$$

Step 3.4: Compute Failure rate $FR(R_j) = T_f / T_{sub}$ for all $j \in n$ where
T_f is the number of tasks failed to be executed previously in resource j
T_{sub} is the number of tasks submitted to be executed previously in resource j

Step 3.5: Compute Fitness Value:

$$FV(T_i,R_j) = FFR + FTCT$$

where,
FFR is the Failure Rate Fitness function and
FTCT is the Total Completion Time Fitness function and they are given by:

$$FFR = (FR(R_j) - FR_{min}) / 2$$

And:

$$FTCT = (TCT(T_i,R_j) - TCT_{imin}) /2$$

where, FR_{min} is the minimum of failure rates of all the resources and
TCT_{imin} is the minimum TCT of task i in resource $j \in n$.

Step 3.6: Set task T_i , Resource R_j and lowest fitness value to $CANDIDATE(T_i,R_j,FV_{min})$

Step 3.7: Choose task T_{min} with lowest fitness value from $CANDIDATE(T_i,R_j,FV_{min})$

Step 3.8: Dispatch task T_{min} to Resource R_j and remove T_{min} from task list.

Step 3.9: Update $RT(R_j)$ where j is the resource to which the task T_{min} is dispatched.

Step 3.10: Update $FR(R_j)$, if resource R_j fails, where j is the resource to which the task T_{min} is dispatched.

Step 4: If there are tasks in Task_list, repeat step 3.
Else Compute
Makespan = max {RT (R_j)} and
Hit Rate = T_{succ}/T_{sub} for all $j \in n$ where
T_{succ} is the number of tasks successfully completed without any failure.

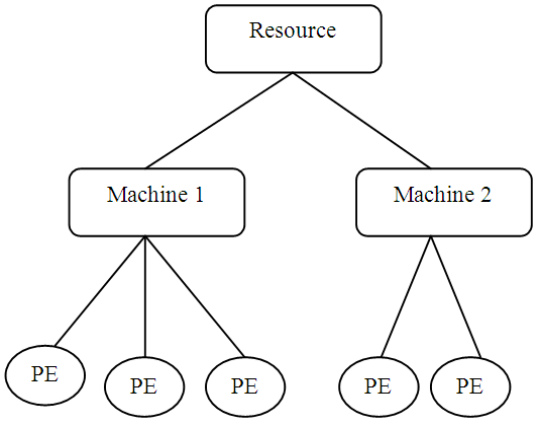

Fig. 2. Arrangement of grid resource in GridSim

2.3. Simulation Setup

The main aim of the proposed scheduling algorithm is to minimize the makespan and to improve fault tolerance of the system proactively and that is achieved by increasing the Hit rate. The simulation is done with GridSim 5.0 toolkit.

Number of Resources	: 16
Number of Tasks	: 512
Initial FR of resources	: 0 to 1

The arrangement of grid resources in GridSim 5.0 and the hierarchy of resources used for evaluating the proposed scheduling algorithm is given in **Fig. 2**. Each resource is characterised by number of machines and each machine is characterised by number of processing elements.

3. RESULTS

The proposed FTMM algorithm is simulated with 512 tasks and 16 machines for 5 different inputs using GridSim5.0 Toolkit with the above mentioned setup and compared with min-min algorithm (without fault tolerant measures) based on makespan and hit rate.

The percentage of improvement of makespan values of FTMM over min-min is given in **Table 3**.

4. DISCUSSION

The results show that the proposed FTMM algorithm outperforms traditional min-min algorithm with better makespan and better hit rate. The makespan and hit count values for 512 tasks in 16 resources is given in **Table 1-4**.

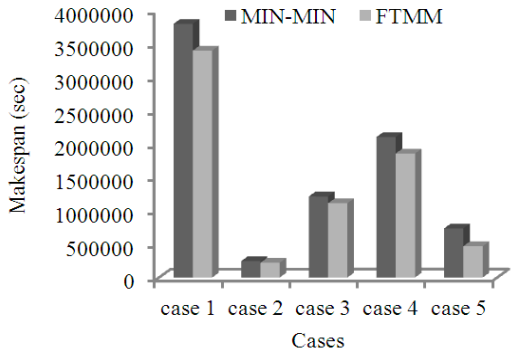

Fig. 3. Comparison chart based on Makespan (sec)

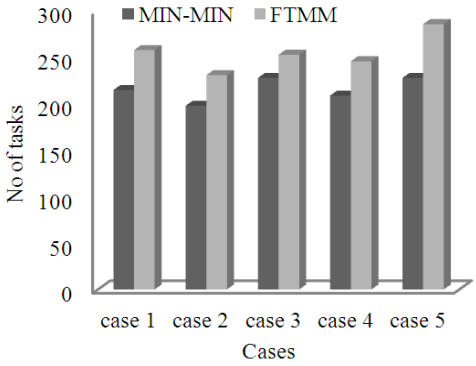

Fig. 4. Comparison chart based on Hit count

Table 1. Comparison based on Makespan (sec)

Cases	MIN-MIN	FTMM
1	3804441	3403760
2	245245	222041
3	1215524	1113854
4	2106230	1860084
5	735913	469402

Table 2. Comparison based on Hit count (No. of tasks)

Cases	MIN-MIN	FTMM
1	215	258
2	198	231
3	228	253
4	209	246
5	228	286

Table 3. Improvement of FTMM over min-min based on makespan

Cases	Improvement (%)
1	10.530
2	9.469
3	8.360
4	11.680
5	36.210

Table 4. Improvement of FTMM over min-min based on hit count

Cases	Improvement (%)
1	8.40
2	6.45
3	4.88
4	7.23
5	11.33

The graphical representation of both the makespan and hit rate is given in **Fig. 3 and 4**. The average percentage improvement of five different sets of 512 tasks and 16 resources based on makespan is 15.25% and based on hit count is 7.66%.

5. CONCLUSION

5.1. Conclusion and Future Work

The problem of grid scheduling is addressed in this study with a solution of providing fault tolerance along with scheduling. The simulation results shows that the proposed FTMM scheduling algorithm with fault tolerance shows high hit rate and minimized makespan. 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 and other criterias like user deadline can also be included.

6. REFERENCES

Anne, B., C. Murray, G. Stephen and H. Jane, 2005. Enhancing the effective utilisation of grid clusters by exploiting on-line performability analysis. Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, May 9-12, ACM Press, USA., pp: 317-324. DOI: 10.1109/CCGRID.2005.1558571

Braun, T.D., H.J. Siegel, N. Beck, L.L. Boloni and M. Maheswaran *et al*., 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput., 61: 810-837. DOI: 10.1006/jpdc.2000.1714

Garg, R. and A.K. Singh, 2011. Fault Tolerance in grid computing: State of the art and open issues. Int. J. Comput. Sci. Eng. Survey, 2: 88-97. DOI: 10.5121/ijcses.2011.2107

He, X., X. Sun and G.V. Laszewski, 2003. Qos guided min-min heuristic for grid task scheduling. J. Comput. Sci. Technol., 18: 442-451. DOI: 10.1007/BF02948918

Lee, H., D. Park, M. Hong, S.S. Yeo, S.K. Kim and S.H. Kim, 2009. A resource management system for fault tolerance in grid computing. Proceedings of the IEEE International Conference on Computational Science and Engineering, Aug. 29-31, IEEE Xplore Press, pp: 609-614. DOI: 10.1109/CSE.2009.257

Malarvizhi, N. and V.R. Uthariaraj, 2009. A minimum time to release job scheduling algorithm in computational grid environment. Proceedings of the IEEE 5th International Joint Conference on INC, IMS, IDC, Aug. 25-27, IEEE Xplore Press, Seoul, pp: 13-18. DOI: 10.1109/NCM.2009.373

Modiri, V., M. Analoui and S. Jabbehdari, 2011. Fault tolerance in grid using ant colony optimization and directed acyclic graph. Int. J. Grid Comput. Appli., 2: 14-26. DOI: 10.5121/ijgca.2011.2102

Nska, G.W., R.V. Van Nieuwpoort, J. Maassen, T. Kielmann and H.E. Bal, 2006. Fault-tolerant scheduling of fine-grained tasks in grid environments. Int. J. High Perform. Comput. Appli., 20: 103-114. DOI: 10.1177/1094342006062528

Rodero, I., F. Guim and J. Corbalan, 2009. Evaluation of coordinated grid scheduling strategies. Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications, Jun. 25-27, IEEE Xplore Press, Seoul, pp: 1-10. DOI: 10.1109/HPCC.2009.28

Suresh, P. and P. Balasubramanie, 2012. User demand aware scheduling algorithm for data intensive tasks in grid environment. Eur. J. Sci. Res., 74: 609-616.

Zheng, Q., B. Veeravalli and C. Tham, 2007. Fault-tolerant scheduling for differentiated classes of tasks with low replication cost in computational grids. Proceedings of the 16th International Symposium on High Performance Distributed Computing, Jun. 25-29, ACM Press, USA., pp: 239-240. DOI: 10.1145/1272366.1272409