A Proactive Fault Tolerant Strategy for Desktop Grid

Geeta Arora[1],Dr. Shaveta Rani[2],Dr. Paramjit Singh[3]
[1]Research Scholar , Dept. of Computer Applications, I.K.Gujral Punjab Technical University, Kapurthala,India
[2, 3] Dept. of Computer Sci. & Engg., I.K.Gujral Punjab Technical University, Kapurthala,India
[1]mailtogeeta@gmail.com
[2]garg_shavy@yahoo.com
[3]param2009@yahoo.com

Abstract—Desktop Grid resources are volatile, heterogeneous and geographically distributed in nature, so scheduling and fault tolerance become the important challenges for Desktop grid systems. In this paper a proactive fault tolerant strategy is developed which also considers the unwanted delays in association with speed and load of resources in the Grid while scheduling jobs on Grid resources. Simulation experiments are conducted by using GridSim toolkit 5.2. The experimental results obtained from applying the proposed strategy considerably improves the performance in terms of Throughput, Turnaround time & Success Rate.

Keywords — Delay time, Desktop Grid, Fault Tolerance, Proactive, Success Rate, Throughput, Turnaround time.

I. INTRODUCTION

Grid Computing is a collection of heterogeneous computer resources distributed over remote locations to achieve a common objective. It allows the secure, transparent & coordinated access to these resources owned by multiple institutions by making Virtual Organizations [1]. On the other hand Desktop Grid objective is to make use of underutilized resources on desktop computers owned by individuals on the edge of internet [2]. Since Internet is basic foundation of Desktop grid, users have to face various kinds of crash and link failures. In addition volunteers are subject to volunteers autonomy failures in the middle of task execution & it will affect the execution of the jobs assigned to the failed resource. So a Fault Tolerance becomes a challenge while optimizing resource scheduling & job execution [3].This paper also considers the delay time of resources in addition to the speed & load of resources while scheduling jobs on Grid resources.

This paper is structured as follows. Section 2 gives an idea about the related work. In Section 3 proposed fault tolerant batch mode scheduling algorithm is discussed. Experimental results are presented and discussed in Section 4. Section 5 concludes the paper.

II. RELATED WORK

In the literature Fault handling mechanisms can be divided in to two types Pro-active [4, 5, 6, 7, 8] & Post-active mechanisms [9, 10, 11]. In pro-active mechanisms, the failures are considered before scheduling of jobs. Whereas, in post-active mechanisms failures of jobs are handled after it has occurred. Proactive mechanisms need more information about grid resources before scheduling and potentially reduce the failure rate & increase the capacity & throughput.

In [4] P. Keerthika et al. proposed an efficient fault tolerant scheduling algorithm (FTMM) in which all the unassigned tasks are uniformly assigned among the available resources to minimize the wasted time and improve the performance.

In [5] P. Keerthika et al. proposed a Bicriteria scheduling algorithm that handles resource failures and thus achieve user satisfaction. The main idea of this paper is to achieve user satisfaction along with fault tolerance to decrease makespan and improve resource utilization.

In[6] Huda et al. presents a Proactive agent based approach in which the agents maintains the information about H/w conditions, Executing process, memory consumption, available resources, component MTTF to deal with selected kinds of faults and thus improves the reliability of Grid services.

In [7] Benjamin et al. proposed three proactive fault handling strategies named Site availability based allocation (SAA), Node availability based allocation (NAA), and Node and Site based allocation (NSA) strategies to estimate the availability of resources in the grid and also the capacity of the grid is being calculated preemptively & then the backfill & replication algorithms are also extended by including proposed strategies to handle the job failures during execution.

In [8] Amoon et al. proposed a novel fault tolerant scheduling system that allocates the resources on the basis of a new factor called Scheduling indicator (SI) which comprises of the response time & failure rate of grid resources.

In [9] B. Nazir et al. proposed a novel task check pointing based job scheduling strategy for an economy based grid which schedules the jobs on the basis of fault index in addition to using a time optimization heuristic. A fault index is being used in proposed strategy to apply different intensity of task checkpointing .

Jiang et al. [10] proposed a security-aware fault tolerant algorithm based upon adaptive job replications which allocates the jobs by matching the user security demand and trust level of the resource.

In [11], Nandagopal and Uthariaraj combine the checkpoint replication based fault tolerance mechanism with minimum total time to release (MTTR) job scheduling algorithm. The scheduler uses the fault index and the response time of resources for making scheduling decisions.

III. PROPOSED WORK

In Traditional Grid System Client allocates their jobs to resource broker by specifying its QoS requirements i.e. deadline, processor capacity and type of operating system and so on. The resource broker further schedules the job to best available resource which meets the client requirements.

Such a computational framework environment has a significant drawback. In this environment, there are resources that accomplish the criterion of deadline, but they have a tendency toward faults. In such a situation, the grid scheduler continues to choose the faulty resource just because the resource meets user's prerequisites which ultimately degrade the performance of grid.

In our proposed strategy the selection of resources will based upon of delay time along with speed and load on the resource. For calculating delay time we maintain a resource history table on the basis of delay time.

Let T = {T1, T2, ..., TN} denote the set of N tasks and R = {R1, R2,..., RM} are set of M resources .The time taken to complete the task depends on size of task, processing speed of the resource and time taken for resource to become ready to execute and the reliability of resource that is calculated as delay time of resource.

Let N is the total number of independent tasks

M is the total number of resources

And i=1...N

J=1...M

Then Completion time (CTij) of the task Ti with respect to resource RJ is calculated by

$$CT_{ij} = ET_{ij} + RT_J + ADT_J$$

Here CTiJ is the Completion time of the task Ti on resource Rj

Estimated Execution Time (ETij) is the estimated amount of time taken by resource R j to execute task Ti.

Ready Time (RTj) is the time taken by resource Rj to become ready to execute any task after finishing the execution of all tasks assigned to it

Average Delay time (ADTj) of resource Rj is the average of delay times DTj of the tasks assigned to it .

$$ADT_J = \frac{\Sigma_{i=1..N \; j=1..M} DT_{ij}}{N}$$

DTij is the delay time of ith task on jth resource

N is the Total tasks has been executed on that resource

Here Delay time (DTij) is difference between estimated execution time and actual execution time of task Ti on a resource Rj. Hence delay time

$$DT_{ij} = (ET_{ij} - AT_{ij})$$

DTij is the delay time of ith task on jth resource

ETij is the estimated execution time of ith task on jth resource

ATij is the actual execution time of ith task on jth resource

---------Data Structures -----

Bt {is the queue of the tasks (Ti is the ith task)}

Tmax {is the task with maximum size from Bt}

R {is the set of resources (Rj is the jth resource)}

Rmin { is the resource with minimum completion time.}

CTij {is the Completion Time of ith task on jth resource)

ETij {is the Execution Time of ith task on jth resource)

RTj {is the Ready Time of jth resource)

ADTj { is the Average Delay Time of jth resource}

----------Proposed algorithm------

1. For all submitted tasks Ti in Bag of Tasks Bt
2. For all resources Rj
3. Construct ETij matrix of size N*M
4. Construct RTj matrix of size 1*M
5. Construct ADTj matrix of size 1*M from Resource History Table
6. Compute $CT_{ij} = ET_{ij} + RT_J + ADT_J$
7. End For
8. End For
9. While tasks Ti in Bag of Tasks Bt is not empty
8. Find the task Tmax from Bt with maximum Size and resource Rmin from Rj with minimum completion time.
9. Allocate task Tmax to the resource Rmin
10. Remove the task Tmax from Bag of Tasks Bt
11. Update RTmin for selected resource Rmin
12 Update delay time ADTmin of the resource history table
13. Calculate CTimin
14. End While

In the above algorithm it is assumed that task and resource attributes such as number of tasks, size of tasks, number of resources, and execution speed of resource are known in advance. The completion time matrix (CTij) is constructed by adding the expected execution time (ETij) of ith task on jth resource, ready time (RTj) and average delay time (ADTj) of Jth resource. Our algorithm find a most suitable resource for a task by calculating the average delay time of all the resources by maintaining a history of resources in resource history table .Based on the history our scheduler prefers resource having minimum average delay time and in this way it avoids the faulty resources and improve performance in terms of throughput, average turnaround time and success rate.

IV. EXPERIMENTAL RESULTS

Java based simulation toolkit GridSim5-2 [12] is used to simulate our proposed "Proactive fault tolerant Algorithm". It also facilitates fault-tolerant services by allowing injection of faults in the grid.

Experiments are conducted under four scenarios with variation of total number of tasks, resources with varying speed and tendency to fail. It is assumed that resources may fail at any movement of time so different event file is taken for each resource. The simulations are conducted to compare and analyze our proposed algorithm with non fault tolerant Batch mode scheduling algorithm Max-Min [13].

The workload for each task is randomly generated between 18850 to 131950 million instructions. The speed of resource varied from 100 to 377 MIPS. The detailed simulation parameters are shown in table 1, 2, 3 and 4 as follows:

TABLE 1. SIMULATION PARAMETERS OF SCENARIO 1

Parameters	**Value**
Number of Resources	3
Number of tasks	5
Task Size (MI)	18850-113100
Resource Speed (MIPS)	100 – 300

TABLE 2. SIMULATION PARAMETERS OF SCENARIO 2

Parameters	Value
Number of resources	7
Number of tasks	10
Task Size (MI)	37700-131950
Resource Speed (MIPS)	150 – 377

TABLE 3. SIMULATION PARAMETERS OF SCENARIO3

Parameters	Value
Number of resources	7
Number of tasks	21
Task Size (MI)	18850-131950
Resource Speed (MIPS)	100 - 377

TABLE 4. SIMULATION PARAMETERS OF SCENARIO 4

Parameters	Value
Number of resources	10
Number of tasks	26
Task Size (MI)	18850-131950
Resource Speed (MIPS)	150 - 475

The performance is compared in terms of throughput, average turnaround time and success rate as shown in Figures (1), (2) and (3).

Throughput is the total number of tasks completed per unit Time. Figure 1 depicts throughput comparison of proposed strategy proactive fault tolerant Max-Min with Max-Min [13] under four scenarios shown in table 1, 2, 3 and 4. Proposed strategy shows approximately 48% improvement when compared with Max-Min.This is due to fact because in our proposed strategy the number of task completed per unit time increases as we prefer fast resources with history of lesser delay time.

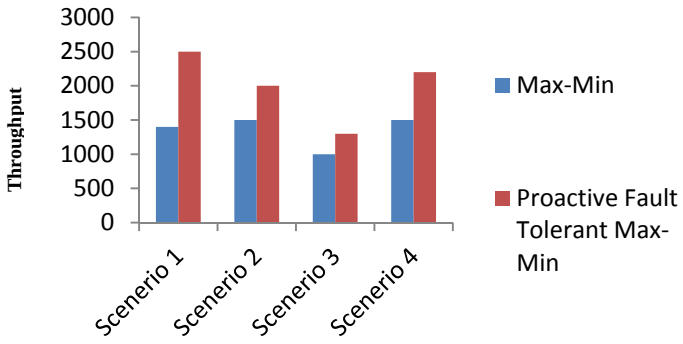

Fig. 1. Comparison of Throughput

Turnaround time is another important parameter to determine the performance of any Fault tolerant system. Turnaround time is the amount of time taken to fulfill a request. In Desktop Grid it is the total time taken between the submission of a task for execution and the return of the complete output to the user. Figure 2 compares our proposed Proactive Fault Tolerant Max-Min with Max-Min [13] with respect to average turnaround time which indicates that our proposed strategy outperforms up to a factor 30% .It is due to the fact because proposed strategy prefer better resource in terms of speed and reliability so outperforms in terms of average turnaround time.

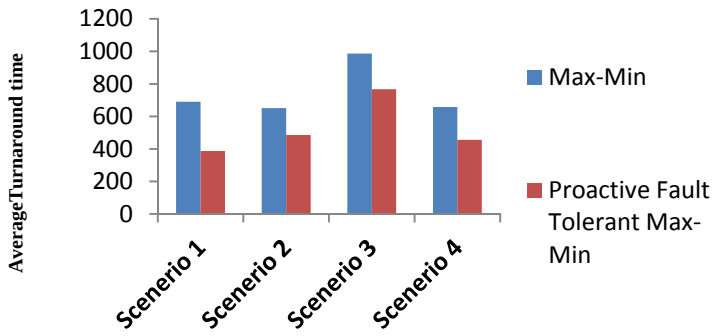

Fig. 2. Comparison of AverageTurnaround time

Another important parameter to measure the performance of desktop grid is Success Rate That is ratio of tasks successfully executed to total number of tasks. Figure 3 demonstrates the comparison of proposed strategy with Max-Min [13] with respect to Success Rate which indicates approximately 14 % improvement. This is because of reality in light of the fact that in proposed strategy most reliable resource with least delay time is selected for execution of task.

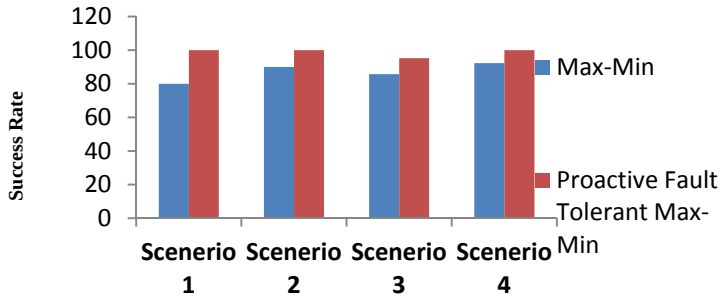

Fig. 3. Comparison of Success Rate

V. CONCLUSION

Due to dynamic, Scalable and fault prone nature of desktop grid, traditional scheduling algorithm does not give good performance. So fault tolerant Strategy is proposed that also considers the unwanted delays in association with speed and load of resources while allocating the tasks to resources. The idea is to increase the throughput and success rate and decrease the turnaround time by giving least preference to most faulty resource. The experimental results depicts that proposed strategy outperforms up to a factor of 48% of throughput, 30% of turnaround time and 15% of Success Rate in spite of unpredictable nature of grid.

REFERENCES

[1] F. Berman, G. C. Fox, et al. Grid Computing Making the Global Infrastructure a Reality, Wiley, 2003.
[2] D. Kondo, M. Taufer, et al. Characterizing and Evaluating Desktop Grids: An Empirical Study, International Parallel and Distributed Processing Symposium 2004 (IPDPS 2004), IEEE CS Press, pp. 26-35, Apr. 2004.
[3] Q. Zheng, B. Veeravalli, et al. On the design of fault-tolerant scheduling strategies using primary-backup approach for computational grids with low replication costs, IEEE Trans , 58(3), pp. 380-393, 2009.
[4] P. Keerthika and N. Kasthuri, An Efficient Fault Tolerant Scheduling Approach for Computational Grid, American Journal of Applied Sciences, vol. 9(12), pp. 2046-2051, 2013.
[5] P. Keerthika and N. Kasthuri, An Efficient Grid Scheduling Algorithm with Fault Tolerance and User Satisfaction, Mathematical Problems in Engineering, vol. 2013, Article ID 340294, http://dx.doi.org/10.1155/2013/340294, 2013
[6] M. Huda, H. Schmidt, et al. An agent oriented proactive fault-tolerant framework for grid computing, in: Proceedings of international conference on e-science and grid computing, Melbourne, Australia, pp. 304-11, December 5-8; 2005.
[7] B.T. Benjamin Khoo, B. Veeravalli, Pro-active failure handling mechanisms for scheduling in grid computing Environments , J. Parallel Distrib. Comput., pp. 189-200, 2010.
[8] M. Amoon, A fault-tolerant scheduling system for computational grids, Elsevier Ltd.,38(2),pp.399-412, March 2012.
[9] B. Nazir, K. Qureshi, et al. Adaptive checkpointing strategy to tolerate faults in economy based grid, J Supercomput, pp.1-18, 2009.
[10] Jiang C, Xu X, Wan J. Replication based job scheduling in grids with security assurance. In: Proceedings of 3rd international symposium on electronic commerce and security workshops, Guangzhou, China, July 29–31; 2010.
[11] Nandagopal M, Uthariaraj, VR. Fault tolerant scheduling strategy for computational grid environment. Int J Eng Sci Technol ,2(9),pp. 4361–72, 2010.
[12] R. Buyya, C. Yeo, et al. Visual modeler and simulation toolkit, in: International Conference on Computational Science, pp. 1123–1132, 2003.
[13] George Amalarethinam. D.I et al. Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems, International Journal of Computer Science and Information Technologies,3 (2) ,pp. 3659-3663, 2012.