Fault Tolerant Job Scheduler with Efficient Job Execution in Grid Computing

P. Latchoumy
Research Scholar, Dept. of IT, BSA University,
Vandalur, Chennai, Tamil Nadu, India.
lak75dhana@gmail.com

P. Sheik Abdul Khader
Prof. & Head, Dept. of CA, BSA University,
Vandalur, Chennai, Tamil Nadu, India.
hodca@bsauniv.ac.in

Abstract*—Grid computing provides the ability to access, utilize and control a variety of underutilized heterogeneous resources distributed across multiple administrative domains while it is an error prone environment. The failure of resources affects job execution during runtime. We propose a new strategy named Fault Tolerant Job Scheduler with Efficient Job Execution using Improved Fault Tolerant Algorithm in Grid computing which effectively schedules grid jobs tolerating faults gracefully and executes more jobs successfully within the specified deadline. This system maintains the history of fault occurrence of resources with respect to processor, memory and bandwidth. Whenever a resource broker has jobs to schedule, the system finds the fault tolerant resources based on their failure rate. The resources with lowest failure rate will have the highest priority for scheduling. The job manager can monitor the execution of job and return the results to the user after successful completion. If failure occurs it reschedules the job with the next optimal resource using the last saved state.

Keywords- Fault-Tolerant Job Scheduler, Resource Fault Value, Failure Rate, IFT Algorithm, Job Manager, Checkpointing , Utilization Rate.

I. INTRODUCTION

The term grid computing is a way to make the computation power of idle work stations available to remote grid users for the execution of their computation hungry jobs [1].

Typically, the probability of a failure is higher in the grid computing than in a traditional parallel computing and the failure of resources affects job execution fatally.

The emergence of grid computing will further increase the importance of fault tolerance. Grid computing will impose a number of unique new conceptual and technical challenges to fault-tolerance researchers. Thus, the incorporation of fault tolerance related features in a grid job scheduling strategy should not be an optional feature, but a necessity.

In this paper, we advocate the need for a fault tolerant job scheduling mechanism for grid environment and add fault tolerant features using failure rate of resources with respect to processor, memory and bandwidth. Here, we present an improved fault tolerant algorithm to find the optimal resources to execute the jobs successfully within the specified deadline. If failure occurs it reschedules the job with the next available optimal resource using the last saved state.

Rest of the paper is organized as follows: Section II contains the description of the related work. In section III, proposed strategy is described. Section IV discusses the experiment result and finally section V concludes the paper.

II. RELATED WORK

Fault tolerance is an important property in grid computing, since the resources are geographically distributed. Moreover the probability of failure is much greater than in traditional parallel systems. Therefore fault tolerance has become a crucial area of interest. A large number of research efforts have already been devoted to fault tolerance [2]. Various aspects that have been explored include design and implementation of fault detection services as well as the development of failure prediction and recovery strategies.

The work on Grid fault tolerance can be divided into pro-active and post-active mechanisms. In pro-active mechanisms, the failure consideration for the Grid is made before the scheduling of a job, and dispatched with hopes that the job does not fail [3]. Whereas, Post-active mechanisms handles the job failures after it has occurred.

Leili Mohammad Khanli and Maryam discussed a strategy named Reliable Job Scheduler using RFOH in Grid Computing. This strategy maintains the history of fault occurrence of resources. Whenever a resource broker has jobs to schedule, it finds the optimal resources using fault occurrence and response time [4]. It does not consider the resource failure as different aspects like processor, memory and BW.

In [5], Amoon and his Co-workers addressed the problem of how to schedule user jobs in grids so that failures can be avoided in the presence of resource faults. He used job replication methodology to avoid failure of jobs. They proposed an algorithm to determine the number of job replicas according to the grid failure history and schedule those replicas. Hence replication increases memory requirement. But our proposed work supports optimal resource utilization with respect to Processor speed,

Memory size and required Bandwidth. Hence the allocation of job depends on the availability and the failure rate of the Processor, Memory and Bandwidth of a resource.

In [6], Babar Nazir , Kalim Qureshi, Paul Manuel proposed a strategy called "Adaptive Checkpointing strategy to tolerate faults in economy based grid". The system uses the different intensity of Check pointing considering that resources have different tendency towards fault. This system is not considering reduction in Checkpointing time. But we have calculated the optimal number of checkpoints using runtime conditions to reduce Checkpointing time [7].

In order to deal with the preceding limitations, the proposed model Fault-Tolerant Job Scheduler with Efficient Job Execution is based on the failure rate of the Resources with respect to processor, memory and BW. This system schedules job to resources with lowest failure rate for successful execution of jobs. After submitting the job to the selected optimal resources, job manager monitors the job execution till completion. If it is failed at unavoidable situation, the job manager will reschedule the interrupted job to the next available optimal resource with last saved state.

III. PROPOSED MODEL

This section explains the proposed model that enables the system to tolerate faults gracefully. The systematic design diagram with process flow of fault tolerant Job scheduler with efficient job execution in grid environment is shown below (Figure 1).

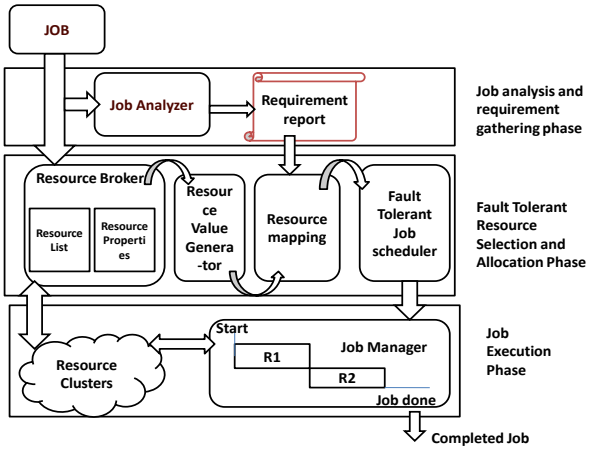

Figure 1. Fault Tolerant Job Scheduler with Efficient Job Execution

Process flow

Initially, the job is given to Job Analyzer and Resource Broker.

1. The Job Analyzer analyzes the job and generates a job requirement report. The job requirement report will be in the form of combination of 1s and 0s.
2. Mean while the Resource Broker generates the resource value report according to the availability and capability of resources on that time. The resource value report will also be in the form of 1s and 0s.
3. The Resource Mapper receives the resource value report from Resource Broker and job requirement report from the first phase.
4. The Resource Mapper maps the resources respective to the job requirement based on the value codes generated and passes it to the Fault Tolerant Scheduler.
5. The Fault tolerant Scheduler prioritizes the resources according to their failure rate using Improved Fault Tolerant (IFT) algorithm.
6. The IFT algorithm first categorizes the resources into capable old resources and capable new resources. If number of executions is greater than zero, then those resources will come under old capable resource and remaining resource will comes under new capable resources.
7. Then the IFT algorithm computes the failure rate of each old resource and sorts them according to their failure rate in ascending order.
8. Then the job will be submitted to the first available resource and in case of unavailability of lowest failure rate resource, the job will be submitted to next available new capable resource.
9. Then the Job Manager starts to monitor the execution of the job.
10. If any interruption occurred during execution, then job will be transferred to next available least failure rate resource from its last saved state by the Job Manager.
11. So, once the job is completed, it will be removed from the Job Manager and returned to the user as successfully completed job.

MODULE DESCRIPTION:

This system is divided into three phases of process. The respective three phases are

- Job Analysis and Requirement Gathering Phase
- Resource Selection and Allocation Phase
- Efficient Job Execution Phase.

Job Analysis and Requirement Gathering Phase

This phase is the first phase in this fault tolerant system. The systems used in this phase are job analyzer and requirement report generator. Here the system receives the job request and sends it to the job analyzer system. The job analyzer analysis the job and generates the report according to the job's nature and its requirements. The requirement values will be either 1 or 0. For example, if the value of speed is 1, then the job needs high speed resources. The report generated by this phase will the in the form as given below (Table1) using the descriptions for each value codes that are generated for jobs and resources (Table3).

Table 1. Job Requirement Report

Job Id	Speed	Memory	Bandwidth
J1	1	0	1
J2	1	1	0

Resource Selection and Allocation Phase

Resource broker generates the Resource capability table with respect to speed, memory and bandwidth of each resource. The values of each column will be either 1 or 0. If the resource is good in memory and processing speed and but having low bandwidth, then the values will be given as 1 for memory and processing speed and 0 for bandwidth. Resource mapping system receives the resource capability report (Table 2) and the job requirement report and maps the respective resources with the job and provides the optimal list of resources.

Table 2. Resource Capability Report

Res. Id	Speed	Memory	Bandwidth
R1	1	0	1
R2	1	1	1
R3	1	0	0

The IFT (Improved Fault Tolerant) algorithm prioritizes the resources with the help of failure rate of each resource. The IFT algorithm first categorizes the resources into already executed (No. of. executions > 0) and new resources (No. of executions=0). After that, computes the failure rate of each resource that comes under already executed category. Then it sorts the resources according to their failure rate in ascending order.

The formula to calculate the failure rate is given below.

Formulae to Calculate Failure Rates

No. of Executions = E
No. of ailures w.r.t Processor = NProc
No. of failures w.r.t Memory = NMem
No. of failures w.r.t Bandwidth = NBW
Failure Rate of Processor = FRProc
Failure Rate of Memory = FRMem
Failure Rate of Bandwidth = FRBW

FRProc = NProc * 100/(E - (NMem+NBW))
FRMem = NMem * 100/(E - (NProc+NBW))
FRBW = NBW * 100/(E - (NMem+NProc))

After computing the failure rate of each resource, the report is generated as given in the Resource Failure Rate Report (Table4) using the descriptions for each value codes that are generated for jobs and resources (Table 3). Then the resource with least failure rate is allocated to the job. In case of unavailability of already executed resources, the IFT algorithm will select the first available capable new resource.

The value codes of resources and jobs will be in either one of the below eight combinations (Table 3).

If the value of capability of resource (processor, memory and bandwidth) is greater than 80% then it considered as High otherwise Low. The binary value 1 is assigned to High and 0 is assigned to Low.

Table 3. Value Code Description

Values	Description
0 0 0	Low Speed, Low Memory, Low BW
0 0 1	Low Speed, Low Memory, High BW
0 1 0	Low Speed, High Memory, Low BW
0 1 1	Low Speed, High Memory, High BW
1 0 0	High Speed, Low Memory, Low BW
1 0 1	High Speed, Low Memory, High BW
1 1 0	High Speed, High Memory, Low BW
1 1 1	High Speed, High Memory, High BW

The failure rate of Processor, Memory and Bandwidth of five resources is given below (Table 4).

Table 4. Resource Failure Rate Report

Res Id	No. of Exec. (E)	Speed		Memory		BW	
		No.of Fail.	Fail Rate. %	No. of Fail.	Fail. Rate %	No. of Fail.	Fail. Rate %
R1	0	0	0	0	0	0	0
R2	200	3	1.5	0	0	7	3.6
R3	100	1	1.9	45	47.4	4	7.4
R4	200	1	0.6	42	21.4	3	1.9
R5	100	0	0	25	25	0	0

Efficient Job Execution Phase

After selecting the respective resource, the job will be scheduled in that selected resource by local job manager. As soon as the execution starts, the job manager calls the checkpoint manager to find the optimal number of checkpoints to reduce the Checkpointing time [7]. Checkpoint manager updates the status of the job completion with its table periodically and passes this information's to the job manager.

During execution, if the job is failed to proceed rather than data insufficiency, then the job will be allocated to the next available resource which have the highest priority from the last saved state. The states of the job will be saved using checkpoint techniques and local job manager. During interruption, the job manager takes care about the selection of next optimal resource and submits the interrupted job from its last saved state and continues to monitor the job till it completely done. On successful completion of the job execution, the job manager returns the output to the user via Resource broker. Once the job is either completed successfully or interrupted, the failure rate table of respective failures of resource is updated by the resource broker with the request initiated by the job manager.

IMPROVED FAULT TOLERANT ALGORITHM

1. Get user request (specifications)
2. Register resources to resource broker with Res id, no. of PEs, Processing Speed, Memory Size, Bandwidth
3. Submit Job to the Job Analyzer
4. Job Analyzer analyze the job & return requirement Report
5. Resource Broker maintains the resource properties and Failure Rate(History) & Assign values to each resources
   If (resource_proc. speed(i) > job_proc. speed)
   Assign resource.proc speed(i) =1 ; Else Assign 0
   If (resource_memory(i) >job_memory)
   Assign resource_memory(i) =1 ; Else Assign 0
   If (resource_baudrate(i) >job_baudrate)
   Assign resource_baudrate(i) =1 ; Else Assign 0
6. Filter the list of resources that have the resource value code >= Job Requirement value code
   a. Calculate the failure rate of each resource in terms of processor, memory & bandwidth limitations
   b. if (E !=0) // if already executed jobs
   // Create optimal list
   Sort the list of capable resources in ascending order with respect to their failure rate. (Smallest failure rate gets highest priority).
   Select the first available, lowest failure rate resource.
   Else // new resources with expected requirement
   Select the first available new resource
7. Return selected resource & Submit job to it
8. Call job manager for monitoring
   a. Start Execution &
   b. Call CheckpointRequest()
   //Calculate the Checkpointing time and update
   //Checkpoint Info.Table in Checkpoint Manager.
   c. Make a handshake with running job;
   If Status= Done
   Assign RSpeed=RMem=RBW=0;
   & Call UpdateHistory()
   ElseIf(job has been failed) // Check Status
   If(Failed due to lack of Proc.) Assign RProc=1;
   If(Failed due to lack of Mem)Assign RMem=1;
   If(Failed due to lack of BW) Assign RBW=1;
   & Call UpdateHistory()
   Return job to next available resource in optimal list;
   // Continue handshaking till it ends
   Goto End
   UpdateHistory()
   If Status=Done
   Increment No.of executions by 1
   Else
   Increment number of executions by one
   Assign NProc=NProc+RProc;
   Assign NMem= NMem+RMem
   Assign NBW=NBW+RBW;
9. Return the completed job to the user.
10. End

IV. EXPERIMENTAL RESULTS

We have implemented our proposed model using Grid Simulation toolkit GridSim 5.2 [13]. A simulation is conducted in heterogeneous environment where each resource has machines with different characteristics such as processing speed, memory size, and bandwidth. The parameters such as number of executions and number of failures are taken into account to calculate the failure rate and the utilization rate of each resource. The simulation is done successfully in order to verify that the proposed IFT algorithm is more efficient than the existing strategy. The simulation setup is given in the Table 5 for five resources with respect to their total number of executions, failure rates, and utilization rate and migration time.

Table 5. Simulation Setup

Resource Id	No. of Executions	No. of Failures	Overall Failure Rate (%)
R1	100	20	20
R2	200	10	5
R3	100	50	50
R4	20	10	50
R5	400	100	25

Simulation results are shown below.

Figure 2. Resources Vs Overall Failure Rate(%)

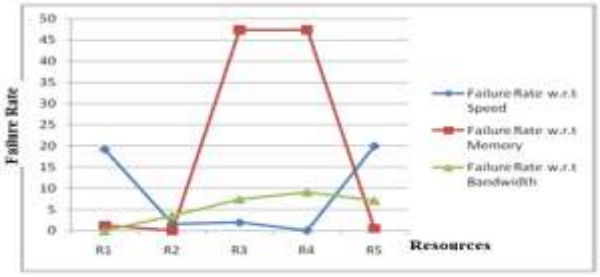

Figure 3. Resources Vs Specific Failure Rate(%)

From Figure 2 and Figure 3, it seen that the proposed strategy can increase the usage of resources while categorizing the resources according to their capability failure rate rather than overall failure rate. Thus the proposed strategy can improve the probability of resource selection depends on the need for the job requirements.

The utilization factor of each resource is calculated to check the utilization range of each resource.

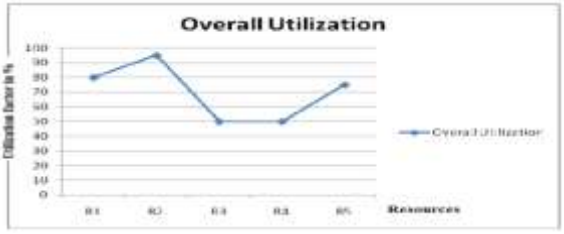

Figure 4. Resources Vs Overall Utilization

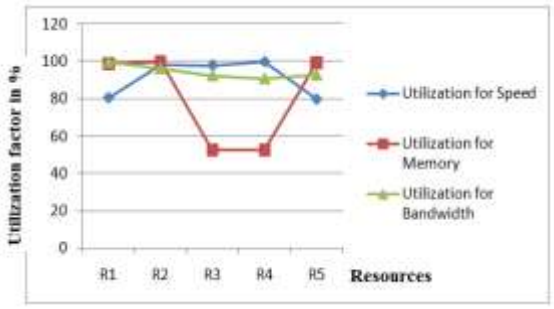

Figure 5. Resources Vs Categorized Resource Utilization

From the Figure 4 and Figure 5, it is seen that the proposed technique gives the utilization factor of each resource during resource selection phase and reduces the wastage of fault tolerant resources. The proposed method improves the success rate of the submitted job to the matched resource even in the occurrence of failure and improves the reliability.

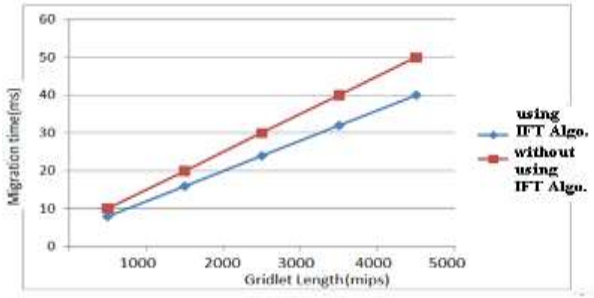

Figure 6. Gridlet Length Vs Migration Time

From the above figure (Figure 6), it is seen that the migration time of proposed IFT algorithm is much lesser than the existing resource selection strategy. Thus the proposed strategy attains the better efficiency in terms of increased resource utilization and reduced migration time.

V. CONCLUSION AND FUTURE WORK

In Grid environment, resource failures can occur for various reasons. In this strategy, a new approach called Improved Fault Tolerant Algorithm (IFTA) for fault tolerant job scheduling in Grid is addressed to assure fault tolerance during execution of failure of job execution with increased resource utilization. The IFT algorithm uses resources information which is maintained in the grid information server and generates the value for each resource according to their capability. The usage of this information causes the reduction of selecting chances of the resource which have more failure probability with respect to Processor, Memory and Bandwidth. After selection of resource, the job will be submitted to the selected resource and the submitted job will be monitored for its successful completion by the job manager. Thus we conclude that even in the presence of faults, the proposed strategy effectively schedules grid jobs tolerating faults gracefully and executes more jobs successfully within the specified deadline and allotted budget.

In future, in order to improve the throughput of this fault tolerant scheduler, the prediction of failure can be found using some more failure parameters. And also, the proposed strategy has not addressed on recovery time reduction during unavoidable failure situation. So these are the areas that can be worked upon.

VI. REFERENCES

[1] Foster, C. Kesselman, and S. Tueke (2001), "The anatomy of the grid: Enabling scalable virtual organizations," Supercomputing Applications.

[2] P. Latchoumy, P. Sheik Abdul Khader (2011), "Survey on Fault Tolerance in Grid Computing", International Journal of Computer Science & Engineering Survey(IJCSES) Vol. 2, No. 4, November 2011.

[3] Huda MT, Schmidt HW, Peake ID (2005), "An agent oriented proactive fault tolerant framework for grid computing", In: First international conference on e-science and grid computing (e-science'05).

[4] Leili Mohammad Khanli, Maryam Etminan Far, Amir Masoud Rahmani (2010), "RFOH: A New Fault Tolerant Job Scheduler in Grid Computing" june 2010.

[5]Amoon.M. dept. of comput. sci., king saud univ., riyadh, saudi arabia (2011)" design of a fault-tolerant scheduling system for grid computing". in networking and distributed computing (icndc), 2011 second international conference .

[6] Babar Nazir , Kalim Qureshi, Paul Manuel (2008), "Adaptive checkpointing strategy to tolerate faults in economy based grid" ©Springer Science+Business Media.

[7]P. Latchoumy, P. Sheik Abdul Khader (2012),"Fault Tolerant Scheduler with Reduced Checkpointing Time in Grid Computing" in the National Conference on Information Technology (NCIT'12).

[8] Dasgupta, G.; Ezenwoye, O.; Liana Fong; Kalayci, S.; Sadjadi, S.M.; Viswanathan, B.," Runtime Fault-Handling for Job-Flow Management in Grid Environments ", In International Conference on Autonomic Computing, 2008.

[9] S.Baghavathi Priya, M. Prakash, Dr. K. K. Dhwan (2007), "Fault Tolerance-Genetic Algorithm for Grid Task Scheduling using Check Point," The Sixth International Conference on Grid and Cooperative Computing (GCC).

[10] Imran, M.; Niaz, I.A.; Haider, S.; Hussain, N.; Ansari, M.A. ," Towards Optimal Fault Tolerant Scheduling in Computational Grid".In,Emerging Technologies, 2007. ICET 2007. International Conferance.

[11] Li Y, Lan Z (2006), " Exploit failure prediction for adaptive fault tolerance in cluster", In: Proceedings of the sixth IEEE international symposium on cluster computing and the grid (CCGRID'06),ISBN 0-7695-2585-7, vol1.

[12] http://www.cloudbus.org/gridsim/

[13] http://sourceforge.net/projects/gridsim/