Improved Fault Tolerant Job Scheduler for Optimal Resource Utilization in Computational Grid

P. Latchoumy
Research Scholar, BSA University,
Vandalur, Chennai, Tamil Nadu, India.

P. Sheik Abdul Khader
Professor & Head, BSA University,
Vandalur, Chennai, Tamil Nadu, India.

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 Improved Fault Tolerant Job Scheduler (IFTJS) for Optimal Resource Utilization in Computational Grid 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. The usage of this information causes the reduction of selecting chances of the resources which have more failure probability and hence improves the resource utilization. Also, the system guarantees the efficient job execution using Reduced Recovery Time (RRT) strategy. Whenever the scheduler has jobs to schedule, the Improved Fault Tolerant (IFT) algorithm finds the optimal resources based on their failure rate. The resources with lowest failure rate will have 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 re-executes the job with the same resource using the last saved state when the Failure Rate of the resource is lesser than the optimal value or with the backup resources when it exceeds an optimal value with the last saved state using RRT strategy. Otherwise it reschedules the failed job with the next available optimal resource using the last saved state. Hence the recovery time is getting reduced. Approach is effective in the sense that the resource manager detects the occurrence of resource failures and the job manager guarantees that the submitted jobs executed with optimal resources with the specified deadline.

Keywords
Improved Fault-Tolerant Job Scheduler (IFTJS), Failure Rate, Checkpointing Time, Reduced Recovery Time (RRT), Optimal Resources, Job Manager, Resource Manager, Utilization Rate.

1. INTRODUCTION
The term grid computing is a way to make the computational 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 re-executes the job with the same resource using the last saved state when the Failure Rate of the resource is lesser than the optimal value or with the backup resources when it exceeds an optimal value using the last saved state. Otherwise it reschedules the failed job with the next available optimal resource using the last saved state.

Rest of the paper is organized as follows: Section 2 contains the description of the related work. In section 3, proposed strategy is described. Section 4 discusses the experimental results and finally section 5 concludes the paper.

2. 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. Our proposed work use both pro-active and post-active mechanisms for optimal resource utilization and efficient job execution.

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 our work we consider the different aspects of resource failure and hence it leads to optimal resource utilization.

In [5], Amoon and his Co-workers addressed the problem of how to schedule the 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. But replication increases memory requirement and hence it leads to

lesser resource utilization. In our proposed work we find the availability of resources with respect to speed, memory and bandwidth before allocating the job for optimal utilization of resources.

In [6], Babar Nazir , Kalim Qureshi, Paul Manuel proposed a strategy called "Adaptive Checkpointing strategy to tolerate faults in economy based grid". He used only failure rate of resource for calculating the number of checkpoints. But, we have calculated the optimal number of checkpoints using failure rate of resource and runtime conditions of the job and hence it reduces the total job completion time is discussed in the paper [7].

In [7], P. Latchoumy and P. Sheik Abdul Khader proposed a strategy called "Fault-Tolerant Scheduler with Reduced Checkpointing Time in Grid Computing". We proposed an algorithm to reduce the Checkpointing Time, but not considered about the reduction in recovery time. Our proposed algorithm Improved Fault Tolerant Job Scheduler is also considered in reduction of recovery time of failed job.

In order to deal with the preceding limitations, the proposed model Improved Fault Tolerant Job Scheduler (IFTJS) for Optimal Resource Utilization 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 it gets successfully completed. If it is failed at unavoidable situation, re-executes the job with the same resource using the last saved state when the Failure Rate of the resource is lesser than the optimal value or with the backup resources when it exceeds an optimal value using the last saved state. Otherwise it reschedules the failed job with the next available optimal resource using the last saved state.

3. PROPOSED MODEL

This section explains the proposed model that enables the system to tolerate faults gracefully. The overall architecture of Improved Fault Tolerant Job Scheduler (IFTJS) for Optimal Resource Utilization with RRT strategy is shown in Figure 1.

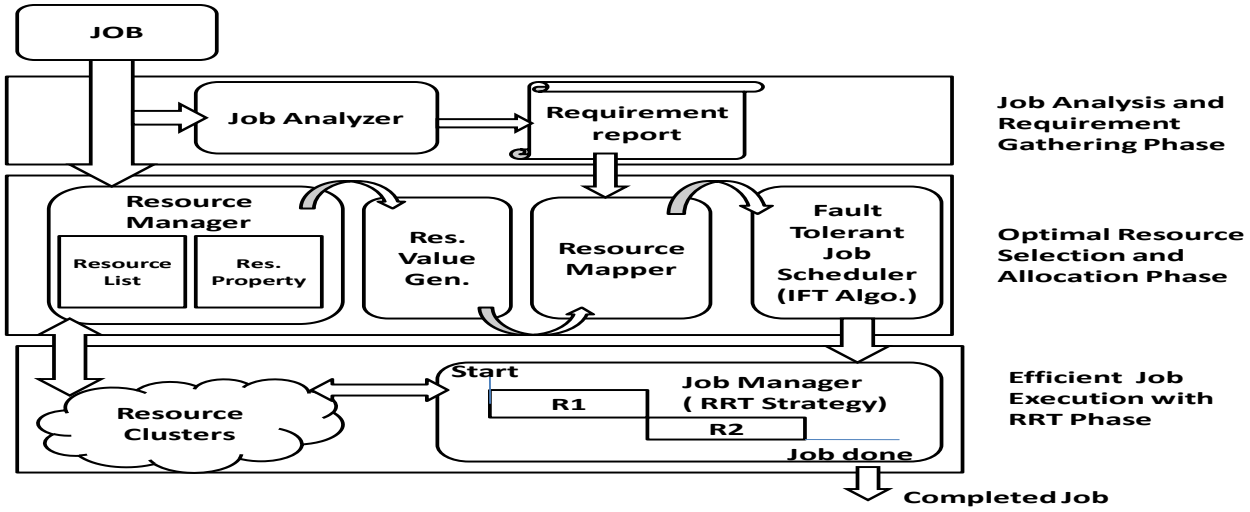

Figure 1. Improved Fault Tolerant Job Scheduler (IFTJS) for Optimal Resource Utilization with RRT Strategy

3.1. Process Flow

Initially, the job is given to Job Analyzer and Resource Manager. The Job Analyzer analyzes the job and generates a job requirement report. Mean while the Resource Manager generates the resource value report according to the availability and capability of resources on that time. The Resource Mapper receives the resource value report from Resource Manager and job requirement report from the first phase. 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. The Fault tolerant Scheduler prioritizes the resources according to their failure rate using Improved Fault Tolerant (IFT) algorithm. 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. Then the IFT algorithm computes the failure rate of each old resource and sorts them according to their failure rate in ascending order. 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. Then the Job Manager with RRT strategy starts to monitor the execution of the job. If any interruption occurred during execution, re-executes the job with the same resource using the last saved state when the Failure Rate of the resource is lesser than the optimal value or with the backup resources when it exceeds an optimal value using the last saved state. Otherwise it reschedules the failed job with the next available optimal resource using the last saved state. So, once the job is completed, it will be removed from the Job Manager and returned to the user as successfully completed job.

The main components of the proposed model are the Fault Tolerant Job Scheduler and Job Manager with reduced recovery time.

The Scheduler allocates the job to the optimal resources using Improved Fault Tolerant (IFT) algorithm. The following diagram (Figure 2) shows the detailed design of the fault tolerant job scheduler with clear visualization of processing steps.

Fault Tolerant Job Scheduler using IFT Algorithm:

1. Gets the list of capable resources and job requirement report
2. Categorizes the new resources and already executed resources
3. Calculates the failure rate of already executed resources
4. Prioritizing the resources based on their failure rate.
   Note: Lowest failure rate gets highest priority and New
   Resources will get least priority
5. Gets the list of Optimal (Fault Tolerant) Resources.
6. Select the first priority Fault tolerant resource and
7. Submit the job to the selected resources

After allocating the job with the optimal resource for the execution, the Job Manager monitor the job till it gets successfully completed in an efficient way using reduced recovery time strategy.

Job Manager with Reduced Recovery Time (RRT) Strategy

If failure occurs the Job Manager re-executes the job with

1. The same resource using the last saved state when the Failure Rate of the resource is lesser than the optimal value, or with
2. The backup resources when it exceeds an optimal value with the last saved state using Reduced Recovery Time (RRT) strategy.
   Where Optimal Value=90% Failure Rate.
3. Otherwise it reschedules the failed job with next available optimal resource using the last saved state. Hence the recovery time is getting reduced.

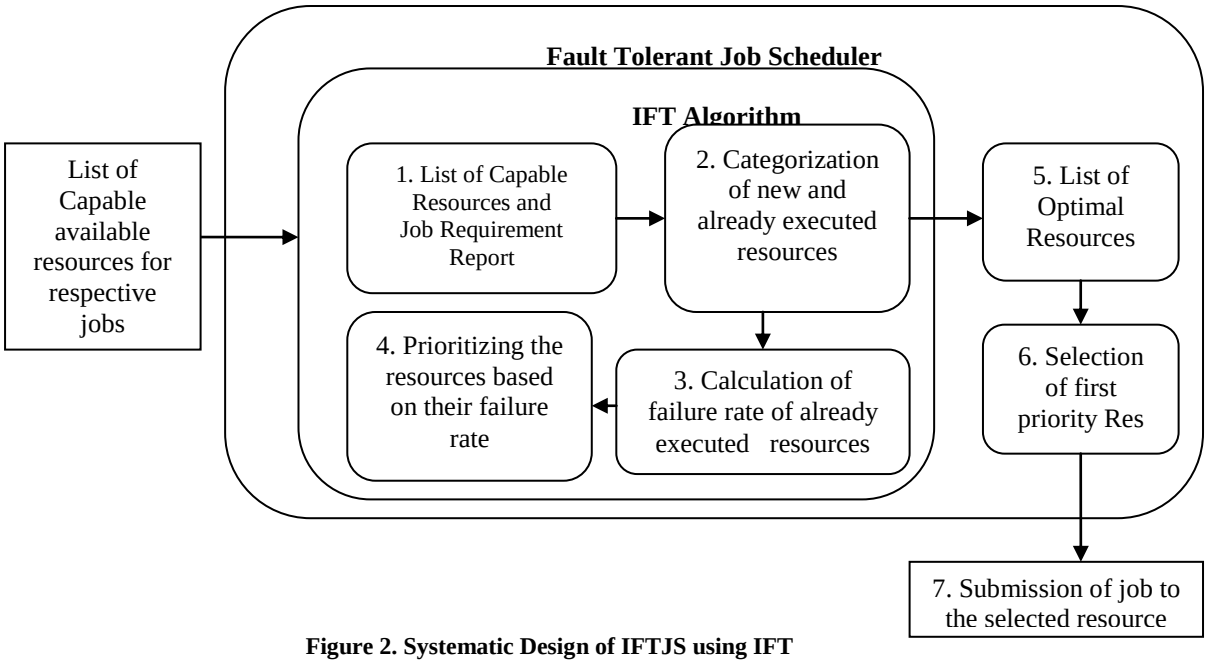

Figure 2. Systematic Design of IFTJS using IFT

3.2. Module Description

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

- Job Analysis and Requirement Gathering Phase
- Optimal Resource Selection and Allocation Phase
- Efficient Job Execution with Reduced Recovery Time (RRT) Strategy Phase

processor. 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

3.2.1 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

3.2.2 Optimal Resource Selection and Allocation Phase

Resource Manager generates the Resource capability value report 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 value report (see Table 2) and the job requirement report and maps the job with the respective resources and provides the optimal list of resources.

Table 2. Resource Capability Value 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 system first categorizes the resources into already executed (No. of. executions > 0) and new resources (No. of executions=0). After that, it 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.

3.2.2.1 Formula to Calculate Failure Rates

No. of Executions = E

No. of failures 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))

The value codes of resources and jobs will be in either one of the eight combinations as shown in Table 3. If the value of capability of processor speed, memory and bandwidth is greater than 80% then it considered as High otherwise considered as Low. The binary value 1 is assigned to High and 0 is assigned to Low.

Table 3. Value Codes Description

Value Codes	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

Using the descriptions for each value codes that are generated for jobs and resources (see Table 3) the system computes the failure rate of each resource and the report is generated as given in the Resource Failure Rate Report (see Table 4). Then the resource with least failure rate is assigned to the job.

Table 4. Resource Failure Rate Report

Res Id	No. of Exec. (E)	Speed		Memory		Bandwidth	
		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

The process of selecting the optimal resources according to their failure rate is represented in the Figure 3. In case of unavailability of already executed resources, the IFT algorithm will select the first available capable new resource.

3.2.2.2 Example for Resource Selection

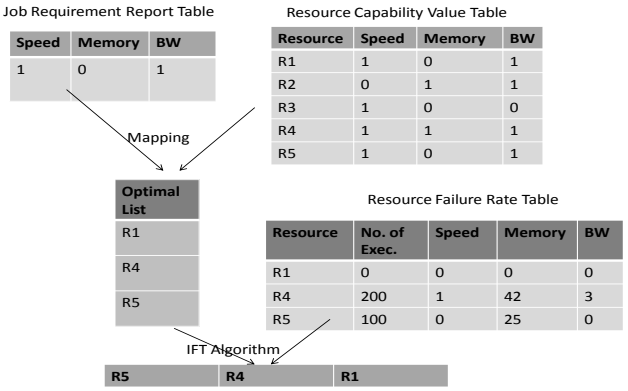

Figure 3. Resource Selection Process

3.2.3 Efficient Job Execution with Reduced Recovery Time (RRT) Strategy Phase

After selecting the respective resource, the job will be scheduled in that selected resource by the 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. Checkpoint manager updates the status of the job completion with its table periodically and passes this information's to the job manager. During failure, the job manager calls the recovery manager to select an appropriate method for recovery of the job using RRT strategy.

3.2.3.1. Checkpoint Manager

An inappropriate check pointing interval leads to delay in the job execution, and reduces the throughput. Hence checkpointing frequency is calculated based on current status and history of failure information of the resource [7]. Checkpoint Manager in the job manager receives the partially executed result of a task from a grid resource in the intervals specified by the job manager based on the checkpointing frequency. It maintains grid tasks and their checkpoint table which contains information of partially executed tasks by the grid resources. It also receives and responds to the task completion and task failure message from grid resources. It updates its table periodically and passes that information's to job manager.

3.2.3.2. Recovery Manager

During interruption, the recovery manager in the job manager selects an appropriate method for recovery of the job from the failure during an execution using RRT Strategy.

Method 1: Recovery using the Same Resource.

If failure occurs Job Manager re-executes the job with the same resource using the last saved state when the Failure Rate of the resource is lesser than the optimal value.

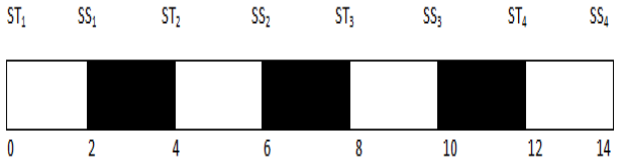

Figure 4. Timeline of Job Execution within Same Resource

ST_i → i^{th} Start Time
SS_i → i^{th} Saved State

Reduced Recovery Time (RRT) = $\sum_{i=1}^{n} (ST_{(i+1)} - SS_{(i)}) + (SS_{(n)})$

Where, ***n*** *is the* number of Saved States &
Checkpointing State is called as Saved State.

Where Optimal Value= 90% Failure Rate.

Method 2: Recovery using the Backup Resource.

If failure occurs Job Manager re-executes the job with the backup resource when the Failure Rate of the resource is exceeds the optimal value with the last saved state.

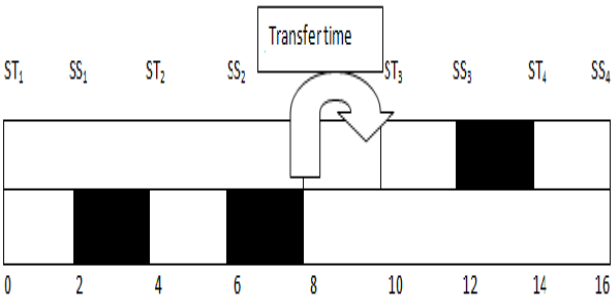

Figure 5. Timeline of Job Execution with Backup Resource

Reduced Recovery Time (RRT) = $\sum_{i=1}^{n} (ST_{(i+1)} - SS_{(i)}) + (SS_{(n)})$ + Transfer Time

Otherwise the job manager reschedules the failed job with the next available optimal (Fault Tolerant) resource using the last saved state and continues to monitor the job till it completely done (see Figure 4).

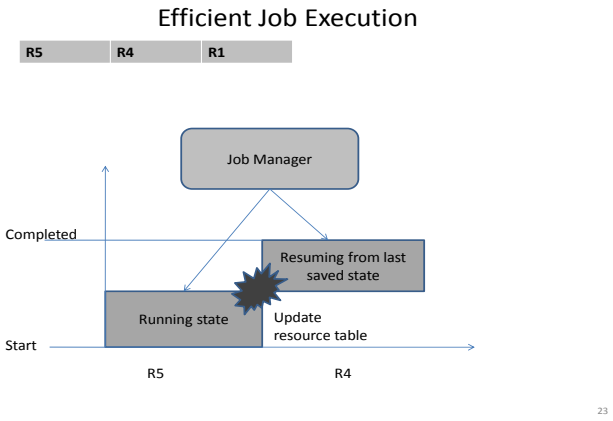

Figure 6. Job Execution Process

On successful completion of the job execution, the job manager returns the output to the user via Resource manager. Once the job is either completed successfully or interrupted, the failure rate table of respective failures of resource is updated by the resource manager with the request initiated by the job manager.

3.3 Improved Fault Tolerant (IFT) Algorithm for Optimal Resource Utilization

1. Get user request (Specifications)
2. Register resources to Resource Manager with Resource id, No. of PEs, Processing Speed, Memory Size, Bandwidth
3. Submit Job to the Job Analyzer
4. Job Analyzer analyzes the job & return job requirement Report
5. Resource Manager maintains the resource properties and Failure Rate (History) & Assign values to each resources
   For each job Ji from a queue
   { For each selected resource Rj
   { If (resource_proc. speed(j) > job_proc. Speed(i))
   Assign resource.proc speed(j) =1 ; Else Assign 0
   If (resource_memory(j) >job_memory(i))
   Assign resource_memory(j) =1 ; Else Assign 0
   If (resource_bandwidth(j) >job_bandwidth(i))
   Assign resource_bandwidth(j) =1 ; Else Assign 0 } }
6. Filter the list of resources that have the resource value code >= Job Requirement value code
   { // FT Scheduling using IFT Algo.
   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 &
   Call CheckpointRequest()
   //Calculate the Checkpointing time and update
   //Checkpoint Info.Table in Checkpoint Manager.
   b. 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 **SelectionRequestforRecovery()**
{ //using Reduced Recovery Time (RRT) Strategy

A. If Failure Rate<=Optimal Value, allow to recover the failed job with the **Same Resource** using the last Saved State.

Calculate RRT=$\sum_{i=1}^{n}(ST_{(i+1)} - SS_{(i)}) + (SS_{(n)})$

B. Else, allow to recover the failed job with the **backup resource** using the last Saved State.

Calculate RRT=$\sum_{i=1}^{n}(ST_{(i+1)} - SS_{(i)}) + (SS_{(n)})$ + Transfer Time
Where Optimal Value= 90% Failure Rate.

C. Calculate Total Job Completion Time
(Where, TJCT<Deadline of the Job)
&
Update Checkpoint Information Table in the Checkpoint Manager.
& Call UpdateHistory()

}
} Else
Return the failed job to next available resource in the optimal list; // continue handshaking till it ends
c. Goto 10 \\ End
UpdateHistory()
If Status=Done
Increment No.of Executions by 1
Else
Increment number of Failures by 1
Assign NProc=NProc+RProc; NMem= NMem+RMem & NBW=NBW+RBW

9. Return the completed job to the user.
10. End

4. EXPERIMENTAL RESULTS

We have implemented our proposed model using Grid Simulation toolkit GridSim 5.2. 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 range of each resource. Recovery time is also calculated and checked with number of failures.

The simulation is done successfully in order to verify that the proposed IFT algorithm is more efficient than the existing strategy. The simulation setup (see Table 5) for five resources with respect to their total number of executions, number of failures and failure rates.

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

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 capability of resources which match the need for the job requirements. The utilization rate of each resource is calculated with respect to processor, memory and bandwidth rather than the overall utilization rate of each resource.

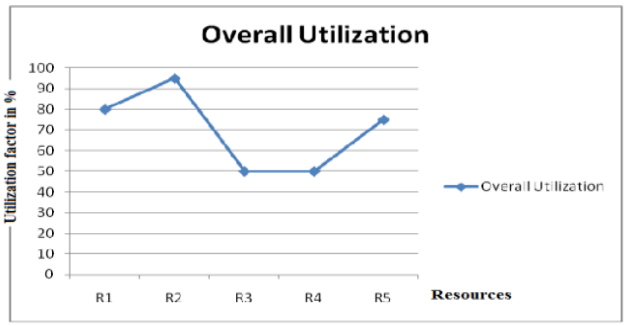

Figure 7. Resources Vs Overall Utilization

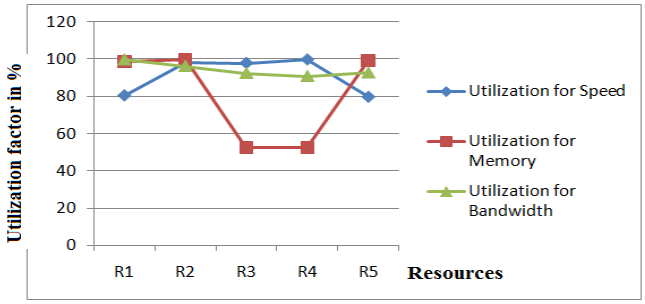

Figure 8. Resources Vs Categorized Resource Utilization

From the Figure 7 and Figure 8, it is seen that the proposed technique gives the utilization of each resource during resource selection phase and reduces the wastage of fault tolerant resources. This leads the optimal utilization of resources.

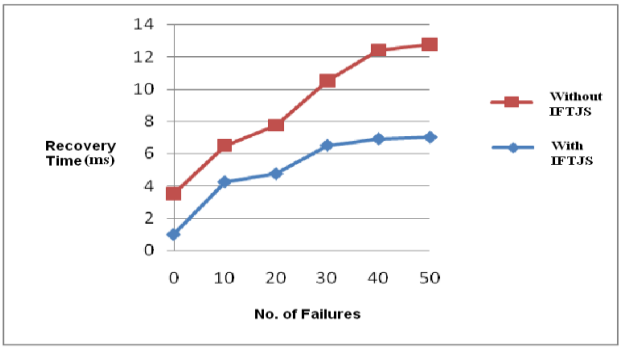

Figure 9. No. of Failures Vs. Recovery Time (ms)

The system reduces the recovery time during the failure of the execution of the job (see Figure 9). The jobs are continued with the appropriate resources that satisfy the condition that is the total job completion time should be lesser than the given deadline. For the collection of 50 jobs, the recovery time is calculated with respect to number of failures occurred.

Thus the proposed strategy attains the better efficiency in terms of increased resource utilization and reduced recovery time.

5. CONCLUSION AND FUTURE WORK

In Grid environment, resource failures can occur for various reasons. In this strategy, a new approach called Improved Fault Tolerant Job Scheduler (IFTJS) for Optimal Resource Utilization in Computational Grid is addressed to assure fault tolerance during job execution with increased utilization of optimal resources. This system maintains the history of fault occurrence of resources with respect to Processor, Memory and Bandwidth. The Improved Fault Tolerant (IFT) algorithm uses this information from the resource manager to generate the value for each resource according to their capability and find the optimal resource for job execution. The usage of this information causes the reduction of selecting chance of the resources which have more failure probability. 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.

The proposed model recovers the failed job during an execution with the same resource if the Failure Rate of resource is lesser than the optimal value or with backup resources if the Failure Rate exceeds an optimal value using last saved state using Reduced Recovery Time (RRT) strategy. Otherwise it reschedules the failed job with the next available optimal resource using the last saved state. Hence the adaptation of recovery method decreases the total job completion time and increases the job throughput, and thus makes the grid environment more reliable.

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

6. 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.

[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.

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

[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) 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 National Conference on Information Technology-NCIT.

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

[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. 2007 Towards Optimal Fault Tolerant Scheduling in Computational Grid In. Emerging Technologies (ICET).

[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] Malarvizhi Nandagopal and Rhymend Uthariaraj 2011 Performance Analysis of Resource Selection Algorithms in Grid Computing Environment Journal of Computer Science 79(4):493-498.