Comparative Analysis of Fault Tolerance Techniques in Grid Environment

R.K.Bawa
Associate Professor
Punjabi University Patiala

Ramandeep Singh
Assistant Professor
Lovely Professional University
Jalandhar

ABSTRACT
Grid being a collection of heterogeneous resources connected through network, to execute complex jobs with high processing power requirements, is more vulnerable to faults. Faults may affect the performance and QoS of Grid. Faults are dealt with either avoiding them or recovering them by either re-execution or by resuming the execution from the point of failure by using the checkpoints. The various fault tolerance techniques use resource management, job scheduling services combined with checkpointing scheme. Different techniques targets different kind of faults and have their respective advantages and limitations. In this paper we have analyzed various faults, fault tolerance approaches and techniques. Finally different techniques have been evaluated based on resource utilization, redundancy, execution time and checkpointing overhead.

General Terms
Fault Tolerance, Resource management, Job Scheduling, Checkpointing, Replication.

1. INTRODUCTION
Grid is a collection of heterogeneous resources in the form of hardware e.g. CPU, memory, devices and software e.g. some specific software applications owned by specific organizations. All these resources are scattered geographically and are connected via different types of networks. These resources are used to execute those jobs in a distributed environment which need a huge amount of computational power. So Grid is actually an implementation of the supercomputing concept in a distributed environment. The various areas of application of Grid Computing are complex scientific experiments, advanced modeling scenarios, astronomical research, wide variety of simulations and complex scientific & business modeling scenarios. Because a large amount of resources are available on Grid so we assume that the average response time will decrease and the overall performance of the system will increase.

2. GRID FAULTS
"Grid is vulnerable to faults"
Being the collection of different types of resources, heterogeneous platforms and different types of networks chances of encountering a fault are very high as compared to any other parallel execution [5] environment. Another reason for the occurrence of faults in Grid environment is that the jobs executed on Grid are very large means huge amount of processing power and resources are required to execute these jobs, few jobs e.g. scientific experiments may even take weeks to execute. So more the time a job consumes while executing, more are the chances of encountering a fault during its execution. The lack of [2] centralized control in Grid is also a reason for the faults. Broadly the faults in Grid environment can be divided in the following categories [2]:

2.1 *Hardware Faults*
Hardware failures take place due to faulty hardware components such as CPU, memory and faulty storage devices. Hardware faults are very difficult to recover because the solution is either the troubleshooting or the replacement of the faulty part.

2.2 *Application and OS faults:*
Application and operating system faults are faced due to exceptions or errors on the node which may be a result of any DoS (Denial of Service attacks), viruses etc.

2.3 *Network or Configuration Faults*
The network faults may lead to a node failure due to loss of connection, packet loss or the corruption of the data during the transmission from one node to the other and it may lead to the faulty or inconsistent results so it can affect the quality of service requirement of the user.

2.4 *Middleware Faults*
Middleware is the interface between the user and the resources on Grid and performs all the major tasks like resource management, resource allocation, job scheduling and fault tolerance etc. Any exceptions in the working of the middleware may lead to faults in Grid.

2.5 *Transient Faults*
Transient faults are the faults which disappear itself eventually without any significant intervention. Transient faults are less severe but hard to diagnose and handle. It is caused by temporary malfunction of some system component. Some environmental interference also causes transient fault or faults.

2.6 *Timing and Interaction Faults*
The jobs which require high amount of interaction between different nodes while executing are affected by these kinds of faults. The delay in messages may cause timeout situations hence interrupting the execution.

According to the survey conducted among Grid users that what kind and extent of faults they are encountering in the usage of Grid [1]. The results are shown in figure 1. Any of these faults may affect the execution of a job in either delaying the completion of the job, totally failing the execution of the job or not meeting the required QOS (quality of service) requirements of the user. These kinds of faults make Grid less reliable for executing the jobs with high QOS requirements. Dealing with [1] these complex failure scenarios is challenging. Detecting that something is wrong is not so difficult (in general, symptoms are quickly identified), but difficulties arise to identify the root cause of the problem, i.e., to diagnose a failure in a very complex and heterogeneous

environment such as a computational Grid. In case of any failure there are chances that the transparency provided by the middleware will be compromised and the user will have to dig deep into the middleware, operating system of the network this is not an easy task for a human being.

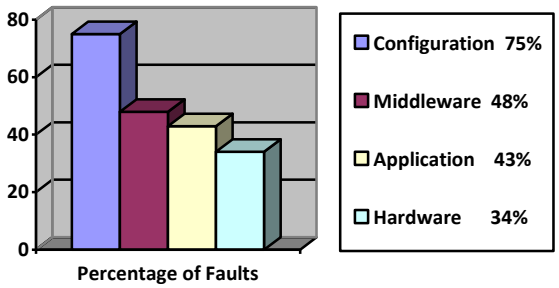

Figure 1.1: Types and Frequency of Faults in Grid

This is a major limitation of Grid. I.e. why fault tolerance is the major area of research in the field of Grid computing now days.

3. GRID FAULT TOLERANCE TECHNIQUES

Fault tolerance techniques deal with faults which are encountered in Grid. A particular fault tolerance technique will have a particular area of application based on its design and implementation. Because [6] Grid is a complex structure and that is what makes it vulnerable to different kinds of faults. There can be a situation in which more than one faults can take place at the same time or one fault may lead to a series of faults. Handling such a situation requires a systematic approach and improved algorithms. The various fault tolerance techniques used in Grid follow two basic approaches:

(a) *Prevention and Resistance*

"*Prevention is better than cure"*
These techniques prevent the faults and resist any situation in which it may have to face faults. These techniques use protective measures while scheduling and executing the job to avoid faults. The performance of Grid may be affected by the overhead being caused by preventive measures.

(b) *Detection and Recovery*

These techniques are based on the concept of detecting faults and recovering from the situation. The recovery may be in the form of re-starting the job or resuming the execution of the job from the point of failure.
R. Buyya [2] has divided the fault tolerance techniques in two sub-parts i.e. Task Level and Work-Flow Level.

(I) Task Level (TL)
Task-level [2] techniques mask the effects of the execution failure of tasks in the workflow. TL fault tolerance techniques have the following sub-types:

(a) Retry technique is the simplest failure recovery technique, as it simply tries to execute the same task on the same resource again after failure.

(b)Alternate resource technique [2] submits failed task to another resource which is available.

(c)Checkpoint/Restart technique moves failed tasks transparently to other resources, so that the task can continue its execution from the point of failure.

(d) Replication technique runs the same task simultaneously on different Grid resources to ensure task execution provided that at least one of the replicas does not fail to execute the job.

(II) Work Flow Level (WFL) or Service level
Workflow-level [2] techniques manipulate the workflow structure such as execution flow to deal with erroneous conditions. Workflow level fault tolerance techniques have the following sub-types:
(a)Alternate task technique executes another implementation of a certain task if the previous one failed.

(b) Redundancy technique executes multiple alternative tasks simultaneously.

(c) User-defined exception handling allows the users to specify a special treatment for a certain failure of a task in workflow.

(d) Rescue workflow technique developed in Condor DAGMan system ignores the failed tasks and continues to execute the remainder of the workflow until no more forward progress can be made.

The FT techniques which are designed and implemented in a Grid environment can be based on any of the approaches discussed in TL and WFL. To improve the efficiency and robustness of the system more than one approach can be combined. The other issues which need to be considered while designing any FT technique are as follows:

(a) Scalability is important because the resource structure in GE (Grid Environment) is dynamic means resources are removed and added into Grid from time to time.

(b) Performance is the basic motivation. The fault tolerance technique should perform in both the conditions i.e. faulty and fault free. The prevention and detection measures should not affect the performance beyond an acceptable loss.

(c) Robustness is the capability to withstand the faults and perform even in a condition with more than one type of faults or a series of faults.

(d) Transparency and Compatibility is necessary because different users should be able to use it irrespective of their location and the resources or configuration. (As for as the minimum requirements for use are satisfied)

(e) *Consistency and Integrity* should be maintained while not compromising with the performance and efficiency of Grid.

According to the survey conducted by Raissa [1] among all the FT techniques used 57% of it are Application Dependent including the monitoring systems such as GMA [11] and ReGS [12](12), 29% are based on checkpointing based and 14% based on some other techniques.

4. FT TECHNIQUES –ANALYSIS & DESIGN

The FT techniques mainly use two basic approaches which are replication and checkpointing. Other than using these two basic approaches individually, these may also be combined with any other technique based on the QoS service requirements.

4.1 Resource Management for Fault Tolerance

The resource management is process of keeping track of the information about the resources so that it may be used for supporting the other services e.g. job scheduling and fault tolerance. In a GE the no. of available resources may vary from time to time due any reasons like any fault or may be the resource is has been withdrawn from Grid by the owner. The information about a resource is stored in the form of various parameters which describe a resource and may vary from one resource management technique to another e.g.

- Type of Resource
- Performance
- Cost for Using the Resource
- Reliability

The FT techniques which are based on the resource management are mainly concerned about the reliability factor of the resource. The reliability of a resource is defined by its capability to resist faults and completing the job within the time and also providing the required QoS. So based in the number of successful and unsuccessful execution performed by a resource in past the reliability of a resource can be calculated and also the probability of the occurrence of a fault. This information is very important because it can be used while deciding that which resources will be used for executing the job. The user may select the resources with more reliability and better performance based on the QoS requirement. The more reliable resources can be used for executing the jobs in real time. But the reliability and performance will come at a price which would obviously be higher than the resources with less reliability and performance. The GIS (Grid Information Services) has access to information about the resources. Many techniques have been designed for fault tolerance using resource management. One of these techniques is proposed by [3] which is based on RFOH i.e. Resource Fault Occurrence History. This technique keeps a track of the faults which have been encountered by a particular resource and the total number of jobs submitted that resource. This information is stored in the form of a table FOHT (Fault Occurrence History Table).

The one limitation of this technique is that it is not designed to deal with the situation when more than one user will try to select the same resource for the execution of their jobs. This approach can be refined by embedding load balancing to prevent more reliable resources from getting overloaded. A technique has been suggested by [9] which uses the GIS to support the user with the decision making process of selecting the resources for the execution of the job. This information service can provide the result of the complex queries submitted by the user from different perspectives.

4.2 Job Scheduling for Fault Tolerance

The job scheduling is a process of selecting the resources for the execution of the job based on the user requirement and then to start the execution of the job. The FT techniques based on the scheduling uses different type of schedulers which can meet the required QoS requirements of the user. The basic FT approaches based on scheduling use the following methods.

- Replication or Redundancy
- Over Provisioning of Resources.

The replication is a process of running more than one replicas of the same job. The process of replication assumes [10] that any resource may face any type of faults so more than required number of resources should be used for executing the job. So that in case if few resources are not able to execute their job we would still be able to get the results from the other nodes or resources. Hence the job may not have to face any delay for its completion. But there is one disadvantage of this technique that it leads [13] to low resource utilization because the resources are being used for performing a redundant work and the cost for the execution of the job will also increase. But the jobs which require a high QoS or real time applications may use this approach.

Replication is very effective in executing those kinds of jobs which require a high amount of interaction and message passing between nodes executing the job. Because if these nodes are not able to interact with each other due to any fault then the execution cannot be completed and it may be delayed or rescheduled due to timeout and it can lead to a chain reaction as shown in the figure 2 below.

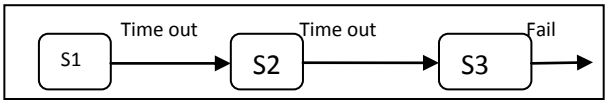

Figure 2: Time out Faults

Replication based FT technique has been suggested in [4]. Grid resources which are compatible to their FT service, register themselves with a UDDI repository. The co-ordination services contact one or more UDDI registries in order to determine the location and number of compatible resources available. The fault tolerance service on nodes is capable of receiving jobs, executing them, performing checksum operations on them, and sending the jobs back. The job is submitted and the FT service at each node generates the checksum of the calculated result and broadcast the checksum to the coordination service(s). The correct result is the result sent by more than half of the nodes. After this a node with correct checksum is selected and is requested for the complete result. Other than low resource utilization the limitation of this technique is that it uses the polling for deciding the correct result and for poling it have to wait for certain number of outputs form different nodes which adds to the total execution time of the job. The other limitation is the network traffic which is caused by the events like registry, job submission by multicast, polling. The replication approach is more effective if sufficient numbers of reliable resources are available to perform replication.

Over provisioning of resources is another method which is used for fault tolerance. More than the required number of resources are assigned to a job before starting the execution of the although not all the resources are used but kept as a reserve in case if any of the nodes executing the job encounter any fault. In case of the fault any resource from the allocated resources will be used to reschedule the failed part of the job. The advantage of this approach is that it provide better availability of the resources but on the other hand other jobs may have to wait for starting the execution due to lack of available resources so over provisioning of resources also leads to low resource utilization.

4.3 Combined Approach of Job Scheduling and Checkpointing for Fault Tolerance

Combined approaches use all the advantages of individual techniques and provide a better environment for dealing with the faults. The problem with the scheduling FT techniques is the absence of the recovery and resuming mechanisms which can resume the execution of the job from the failed stage. So the time and resources used to execute the job before the point of failure will not be useless. The execution of the job can be resumed using the checkpoints which are computed and stored while the job is being executed. The checkpointing approach is very effective while executing the jobs with a long execution time e.g. scientific experiments and simulations which may last for days, weeks or even months. So to recover the processed part of such jobs is very beneficial from the resource utilization and execution time point of view. The two most widely used checkpointing techniques in Grid are:

4.3.1 Application Level Checkpointing

In this technique the checkpointing functionality is inserted in the application code so that application should be able to store and manage checkpoints itself. CPPC (Com Piler for Portable Checkpointing) [4] is an application level checkpointing tool focused on inserting checkpointing code into long running message passing applications. It consists of a runtime library containing checkpoint support routines, together with a compiler that automates the use of the library. CPPC provides all the features which are key issues for fault tolerance support on large scale heterogeneous systems such as Grid. It uses portable code and protocols, and generates portable checkpoint files while avoiding traditional solutions which have some scalability overhead.

4.3.2 System Level Checkpointing

This checkpointing is done at the level of the system executing the job. The system level checkpoints save the entire state of the system as well as the job). It requires more data for saving the state than application checkpoints; this means system can checkpoint any application at an arbitrary point in its execution and allows programmers to be more productive. System initiated checkpointing is better to save the kernel level information and runtime policy whereas application initiated provides more efficient and portable checkpointing. Together [5] checkpointing are job replication techniques can be used to achieve a more fault tolerant environment than any of these individually. Replication will run duplicate jobs and checkpointing will be used to store the intermediate results. It will be an overhead to run duplicate jobs and to manage their checkpoints but this cost can be bear in case of high QoS requirements of the user. Yulan and Yanhang [5] have implemented the same technique and found out that it provides better results.

Checkpoint algorithms [8] may be classified into three broad categories: *(a) synchronous, (b) asynchronous and (c) quasi-synchronous*. In asynchronous check pointing each process takes checkpoints independently. If all the processes take checkpoints at the same time instant, the set of checkpoints would be consistent. Since globally synchronized clocks are very difficult to implement, processes may take checkpoints within an interval. In synchronous check pointing process synchronize through message passing interface before taking checkpoints.

The efficiency and performance of any checkpointing FT technique will be based on the following factors:

- *Frequency* of checkpoints effect the performance by either increasing or decreasing the checkpointing overhead because resources and time is consumed to calculate the checkpoints and to store on the storage media.

- *Availability* of the checkpoints effect the resuming time of a failed process because it takes time to locate and transfer the checkpoint data from the storage (which can be a server or any node) to the node which is going to execute the job after rescheduling the job. Another issue with the storage of the checkpoints is that whether to keep all the checkpoint data on a single node or more than one as suggested by [5]. Considering the possibility of the failure of the single node on which checkpoint data is residing the checkpoints can be replicated on more than one node to increase the availability and to avoid any kind faults or errors which may be caused while transmitting the data from one node to another.

- *Checkpoint Size* if a central site is being used to store the checkpoints then the size of a single checkpoint can be a problem because the storage space requirement will also increase with that [8]. To deal with this issue the checkpoint data can be compressed before storing or transferring over the network and the older checkpoints should be removed from the system when a latest checkpoint has been generated.

To frequency and number of replicas of checkpoints can be decided [5], according to QoS requirements or according to the reliability of the resources which are going to be used for the execution of the job, at the scheduling time of the job. But the frequency of checkpoints is difficult to decide at the scheduling level because different jobs may have different stages of checkpoints so we cannot decide a fix frequency and time of checkpoints for all the jobs. Every FT technique which is designed or existing at present have a particular area of application but there exists no technique which have answer to all type of faults and problems of Grid.

These are the few existing fault tolerance techniques which are being used at present to deal with the faults:

RFOH: A New Fault Tolerant Job Scheduler in Grid Computing [3]

This technique is based on managing the resources to keep a track of their failures. The resources with high rate of failure are avoided while selecting the resources for executing the job. They have also used the checkpointing to recover a job after failure.

Fault Tolerance within a Grid Environment [4]

This technique is based on the replication. It assumes that Grid is a collection of huge number of resources so to achieve the required QoS more than required number of resources can be used. They have proposed the method of voting to select the correct output out of at least more than half outputs and reduce the execution time of the job.

Achieving Fault Tolerance on Grids with the CPPC Framework and GridWay Metascheduler [7]
This paper provides a complete and totally transparent solution for the execution of fault-tolerant sequential and parallel applications on Grids, CPPC are used together with GridWay. The resulting architecture, called CPPC-GW, will be in charge of submitting, monitoring, and automatically restarting the execution in case of failure as well as of generating, managing, and replicating checkpoint files.

A Fault Tolerance Optimal Neighbor Load Balancing Algorithm for Grid Environment [8]
This technique is based on improving the fault tolerance of Grid by using load balancing strategies along with the checkpointing to migrate a task from one node to another and resuming its execution from the point of failure. The goal of this technique is to provide FT but also better resource utilization.

5. COMPARISON
The above mentioned techniques have been compared on the basis of resource utilization, checkpointing overhead caused by the particular technique, processing cost which will be based on the number and type of resources being used, and load balancing functionality, replication and the total execution time. The result is shown in table 1. The existence or value of one parameter will affect the other dependent parameter values e.g. replication may decrease the execution time but it will increase the execution cost and decrease the resource utilization due to the redundancy.

6. CONCLUSION
This paper provides the study of Grid faults, approaches and techniques which are used for fault tolerance. The techniques have been analyzed and compared based on different factors. The fault tolerance services in Grid make Grid more reliable and provide a better execution environment in terms of execution time and QoS. The checkpointing approach can be used to resume the execution of a failed job. The frequency and availability of checkpoints plays a major role in the efficiency of a fault tolerance technique. Replication techniques can be used to execute the jobs with high QoS requirements but it decreases the resource utilization but the total execution time is reduced. Another important functionality which all these techniques except one [8] are missing is the capability to deal with high priority task means preemption of the jobs and load balancing because this will help to use Grid resources in more efficient way.

Table 1: Comparison of Fault Tolerance Techniques

Techniques Parameters	RFOH [3]	FT in Grid [4]	FT with CPPC [7]	FT with LB [8]
Resource Utilization	HIGH	LOW	HIGH	HIGH
Replication	NA	HIGH	NA	NA
Checkpointing Overhead	LOW	NA	HIGH	AVG
Execution Time	AVG	HIGH	AVG	LOW
Load Balancing	NA	NA	NA	YES
Processing Cost	AVG	HIGH	AVG	LOW

7. REFERENCES
[1] Raissa Medeiros, Walfredo Cirne, Faults in Grids: Why are they so bad and What can be done about it?", Proceedings of the Fourth International Workshop on Grid Computing 2003.

[2] Jia Yu , Rajkumar Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing ". Department of CS and SE university of Melbourne, Australia.

[3] Leili Mohammad Khanli, Maryam Etminan Far, Amir Masoud Rahmani, "RFOH: A New Fault Tolerant Job Scheduler in Grid Computing" IEEE Second International Conference on Computer Engineering and Applications P 422-425 2010.

[4] Paul Townend, Jie Xu, "Fault Tolerance within a Grid Environment" IEEE Second International Conference on Computer Engineering and Applications 2009. Department of Computer Science University of Durham.

[5] Yulan Yin, Yanhong Zhao, Fengna Dai, "Fault Tolerance Scheduling in Economic Grids". IEEE P 2252-2256 2011.

[6] Jesus Montes, Alberto Sanchez, Maria S. Perez "Improving Grid fault tolerance by means of global behavior modeling", Ninth International Symposium on Parallel and Distributed Computing P 101- 108 2010.

[7] Ivan Cores, Gabriel Rodrıguez, Maria J.Mart ın and Patricia Gonzalez, "Achieving Fault Tolerance on Grids with the CPPC Framework and GridWay Metascheduler".22nd International Symposium on Computer Architecture and High Performance Computing P 119 -126 2010.

[8] Jasma Balasangameshwara, Nedunchezhian Raju, "A Fault Tolerance Optimal Neighbor Load Balancing Algorithm for Grid Environment", International IEEE Conference on Computational Intelligence and Communication Systems P 428 – 433 2010.

[9] Francisco Brasileiro, Lauro Beltrao Costa, Alisson Andrade, Walfredo Cirne "A large scale fault-tolerant Grid information service" MGC 06 November 27, 2006 Melbourne, Australia.

[10] Yongjian Wang, Zhongzhi Luan, Depei, DDGrid: A Grid Computing Environment with Massive Concurrency and Fault-tolerance Support. Proceedings of the IEEE Seventh International Conference on Grid and Cooperative Computing P 5-14 2008.

[11] B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany. A Grid Monitoring Architecture. Working Document, January 2002.

[12] Y. Aridor, D. Lorenz, B. Rochwerger, B. Horn, and H. Salem, "Reporting Grid Services (ReGS) Specification.", IBM Haifa Research Lab, January 2003.

[13] Congfeng Jiang, Cheng Wang, Xiaohu Liu, "Adaptive Replication Based Security Aware and Fault Tolerant Job Scheduling for Grids" Eighth ACIS International conference on Artificial Intelligence and Parallel distributed Computing IEEE 2009.