Grid Computing with different Fault Tolerant Mechanisms
Mr. Gaurav Prajapati1 Mr. Ramesh Prajapati2 Dr. Samrat Khanna3
1Department of Computer Engineering 2Department of Computer Engineering 3Department of Information Technology
2Rai University, Center for Research & Development Saroda, Dholka, India 3Istar, Sardar Patel center for science & Tech. V.Vnagar, India

Abstract— Grid is beyond simple communication between computers but it aims ultimately to turn the global network of computers into one vast computational resource. Grid is a service of sharing computer power and data storage capacity over the Internet and Intranet. The goal of Grid computing is to create the illusion of a simple but large and powerful self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. Grid computing is defined as a hardware and software infrastructure that enables coordinated. Resource sharing within dynamic organizations. In grid computing, the probability of a failure is much greater Therefore, the fault tolerance is an important property in order to achieve reliability, availability and QOS. Fault tolerance is an important property for large scale computational grid systems, where geographically distributed nodes co-operate to execute a task. In this paper, we give a Review on various fault tolerance mechanisms, types of faults in system and management the various fault in different systems and related issues and how to develop fault tolerant system and recover the faults in the system.

Keywords: Grid computing, Grid Architecture, Fault Tolerance.

I. INTRODUCTION

Grid is a system that coordinates resources that are not subject to centralized control using standard, open, general-purpose interfaces and protocols to deliver non-trivial qualities of service. A grid is a type of parallel and distributed system that enables the allocation, selection and aggregation of resources distributed across multiple administrative domains based on their (resources) availability, capacity, performance, cost and quality of service requirements. A grid is a collection of machine sometimes referred to as nodes, resources, donors, members, clients, hosts, engines and many other such terms [2]. Grid environments are extremely heterogeneous and dynamic, with components joining and leaving the system all the time, more faults are likely to occur in grid environments. Also, the likelihood of errors occurrence is exacerbated by the fact that many grid applications will perform long tasks that may require several days of computation. This will lead to a number of new conceptual and technical challenges to fault tolerance researchers. The most important one is the scheduling of user jobs to grid resources with meeting the user’s Quality of Service (QoS) in existence of resource faults. With the increased popularity of internet and availability of high performance computers and high speed networks as low cost commodity, it has become possible to use networks of computers as a single unified computing resource. So that in the grid Computing [1], main element for concept on Resource sharing, Coordinated problem solving, and Dynamic, multi-institutional virtual organizations. Grid resources are also heterogeneous in nature so resources may enter and leave the grid at any time, in many cases outside of the applications control. So therefore interaction faults may be likely to occur between disparate grid nodes. Also resource may be outside of the organization so that is not guaranteed that a resource being used is not malicious. So here challenging issue is that faults and failure in grid. So fault tolerance is a crucial aspect for grid computing. With the growth of grid technologies, more and more companies are moving from large scale, centralized databases to databases that reside on grid-based systems. Grid technology allows organizations to use numerous computers to solve problems by sharing computing resources.

Grid computing is enabled by relatively high-performance computers, robust computer networks, grid management software, and the divisibility of difficult scientific problems. In Grid computing, individual users can access computers and data, transparently, without having to consider location, operating system, account administration, and other details. Grids tend to be more loosely coupled, heterogeneous, and geographically distributed. In Grid computing details are abstracted, and the resources are virtualized [3].

Section II describes the Literature Survey. In Section III, we talk about Fault Tolerant Mechanisms. Problems for Recovering from a Failure are presented in Section IV. In section V, we present the Research Gaps. We present the conclusion in section VI.

II. LITERATURE SURVEY

As a result of the complex nature of heterogeneous networks, fault tolerance [5] is a major concern for the network administrators, and there are various ways that detection of such occurrences can be accomplished.

When a fault occurs, it is important to: rapidly determine exactly where the fault is, Isolate the rest of the network from the failure so that it can continue to function without interference, Reconfigure or modify the network in such a way as to minimize the impact of operation without the failed component or components, and repair or replace the failed components to restore the network to its initial state.

A. Function of Fault Tolerance

The fault tolerance is “to preserve the delivery of expected services despite the presence of fault-caused errors [6] within the system itself. Errors are detected and corrected, and permanent faults are located and removed while the system continues to deliver acceptable service.”
From a user's point of view, a distributed application should continue despite failures. The fault tolerance has become the main topic of research. Till now there is no single system that can be called as the complete system that will handle all the faults in grids. Grid is a dynamic system and the nodes can join and leave voluntarily. For making fault tolerance system a success, we must consider:

- How new nodes join the system,
- How computing resources are shared,
- How the resources are managed and distributed

B. Fault Tree Analysis

Because of computational Grid heterogeneity, scale and complexity, faults become likely. Therefore, Grid infrastructure must have mechanisms to deal with faults while also providing efficient and reliable services to its end users. The fault tree analysis [18] classifies faults that may take place in Grid Computing. In the figure various kinds of faults that can occur have been shown. There are mainly six classes of faults as discussed below:

![Fig. 1: Types of Faults](image)

**TABLE 1: DETAILED TYPES OF FAULTS**

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Main Type</th>
<th>Detailed Types of faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hardware Relate Faults</td>
<td>CPU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Storage</td>
</tr>
<tr>
<td>2</td>
<td>Application &amp; Operating System Oriented Faults</td>
<td>Operating System Specific</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Memory Leaks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- OS Faults</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Application Specific</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Node fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Network oriented</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Packet corrupted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Packet Losses</td>
</tr>
<tr>
<td>3</td>
<td>Network Related Faults</td>
<td>Unhandled Exception</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unexpected Input</td>
</tr>
<tr>
<td>4</td>
<td>Software Related Faults</td>
<td>Timeout Run Exception</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Timeout Job Exception</td>
</tr>
<tr>
<td>5</td>
<td>Timeout Faults</td>
<td>Value</td>
</tr>
<tr>
<td>6</td>
<td>Response Relate Faults</td>
<td>Byzantine</td>
</tr>
</tbody>
</table>

III. FAULT TOLERANCE MECHANISM

In addition to ad-hoc mechanisms - based on users complaints and log files analysis - grid users have used automatic ways to deal with failures in their Grid Environment. To achieve the automatic ways to deal with failures, various fault tolerance mechanisms [11] are there. Some of these fault tolerance mechanisms are:

1) Application Dependent
2) Monitoring Systems
3) Check pointing-recovery
4) Fault Tolerant Scheduling

![Fig. 2: Fault Tolerance Mechanisms](image)

A. Application-dependent

Grids are increasingly used for applications requiring high levels of performance and reliability, the ability to tolerate failures while effectively exploiting the resources in scalable and transparent manner must be integral part of grid computing resource management systems [13].

Support for the development of fault-tolerant applications has been identified as one of the major technical challenges to address for the successful deployment of computational grids. To date, there has been limited support for application-level fault tolerance in computational grids. Support has consisted mainly of failure detection services or fault-tolerance capabilities in specialized grid toolkits. Neither solution is satisfactory in the long run. The former places the burden of incorporating fault-tolerance techniques into the hands of application programmers, while the latter only works for specialized applications. Even in cases where fault-tolerance techniques have been integrated into programming tools, these solutions have generally been point solutions, i.e., tool developers have started from scratch in implementing their solution and have not shared, nor reused, any fault-tolerance code. A better way is to use the compositional approach in which fault-tolerance experts write algorithms and encapsulate them into reusable code artifacts, or modules.
B. Monitoring Systems

In this a fault monitoring unit is attached with the grid. The base technique which most of the monitoring units follow is heart beating technique. The heart beating technique is further classified into 3 types [15]:

1) Centralized Heart beating - Sending heartbeats to a central member creates a hot spot, an instance of high asymptotic complexity.
2) Ring Based Heart beating - along a virtual ring suffers from unpredictable failure detection times when there are multiple failures, an instance of the perturbation effect.
3) All-to-all heart beating - sending heartbeats to all members, causes the message load in the network to grow quadratically with group size, again an instance of high asymptotic complexity

C. Check pointing-recovery

Check pointing and rollback recovery provides an effective technique for tolerating transient resource failures, and for avoiding total loss of results. Check pointing [16] involves saving enough state information of an executing program on a stable storage so that, if required, the program can be re-executed starting from the state recorded in the checkpoints. Check pointing distributed applications is more complicated than Check pointing the ones which are not distributed. When an application is distributed, the Check pointing algorithm not only has to capture the state of all individual processes, but it also has to capture the state of all the communication channels effectively. Check pointing is basically divided into 2 types:

1) Uncoordinated Checkpoint: In this approach, each of the processes that are part of the system determines their local checkpoints individually. During restart, these checkpoints have to be searched in order to construct a consistent global checkpoint.
2) Coordinated Checkpoint: In this approach, the Check pointing is orchestrated such that the set of individual checkpoints always results in a consistent global checkpoint. This minimizes the storage overhead, since only a single global checkpoint needs to be maintained on stable storage. Algorithms used in this approach are blocking and non-blocking.

D. Fault Tolerant Scheduling

With the momentum gaining for the grid computing systems, the issue of deploying support for integrated scheduling and fault-tolerant approaches becomes a paramount importance. For this most of the fault tolerant scheduling algorithms [7] are using the coupling of scheduling policies with the job replication schemes such that jobs are efficiently and reliably executed. Scheduling policies are further classified on basis of time sharing and space sharing.

Application dependent mechanisms use the reuse concept to integrate fault tolerance within the application code. Most of the present middleware uses heartbeat based monitoring system to monitor the status of grid. In case of Check pointing-recovery and fault-tolerant scheduling, they are only able to deal with crash failure semantics for both hardware and software components. Software faults with more malign failure semantics - such as timing or omission ones, which are even more difficult to deal with - are not covered by them.

IV. PROBLEMS FOR RECOVERING FROM A FAILURE

The greatest problem is to diagnose the failure, i.e. to identify its root cause [8]. The difficulty to implement the application-dependent failure recovery behavior is next main concern (the user does not know what to do to recover from a failure). To gain authorization to correct the faulty component is another important concern. Other problems such as ensuring that failures do not result in orphaned jobs on remote systems (i.e. they get cleaned up in a reasonable time), cleaning up corrupted cache files without losing lots of work in progress, and getting access to preserved state when Check pointing/recovery is used (e.g. checkpoint files may be inaccessible or totally lost) were also highlighted. Diagnosing the Failure Difficulty to implement the failure recovery behavior Gain authorization to correct the faulty components.

V. RESEARCH GAPS

For the future works, there are still many issues that need to be explored. Conventional resource management schemes are based on relatively static model and a centralized controller that manages jobs and resources accordingly. However, this fails to work if the centralized controller fails due to many reasons discussed in previous topic. Secondly there are issues like what will happen if some job hangs down or remain unfinished in between its execution? To overcome these issues, we need an adaptive infrastructure which is robust, flexible and adaptive in nature. This part of this paper will discuss fault tolerance technique that focuses on dealing with such kind of failures. The proposed framework can manage the central manager failure and also tries to keep a check on the correct completion of the job in Alchemy based computational grids.

VI. CONCLUSION

The sharing of computational resources is a main motivation for constructing Computational Grids in which multiple computers are connected by a communication network. Due to the instability of grids, the fault detection and fault recovery is a very critical task that must be addressed. The need for fault tolerance increases as the number of processors and the duration of computation increases. Main objective of the grid environment is to achieve high performance. In future work we investigated the issues and challenges involved in dealing with faults in computational grids, proposed the framework for dealing with various kinds of faults we found in Alchemy based computational grids and evaluated the efficiency of our proposed system under various conditions. To deal with failure of the grid we will propose backup manager concept. Backup manager uses the heart beating based fault tolerant technique to monitor the central manager. In case of failure of the central manager. Backup manager will take its control and avoids the grid to fail.
REFERENCES


[15] Ya-Fing Zhung, Jizhou Sun, Jian-Bo Ma, “Self-Management Model Based on multivalent and Worm Techniques” CCECE 2004


[20] Raissa Medeiros, Walfredo Brasileiro, Francisco Sauvé, “Faults in Grids: Why are they so bad and what can be done about it?” GRID’03