IJCSMS International Journal of Computer Science and Management Studies, Vol. 11, Issue 03, Oct 2011
ISSN (Online): 2231-5268
www.ijcsms.com

GRID COMPUTING AND FAULT TOLERANCE APPROACH

Pankaj Gupta,
Vaish College of Engineering, Rohtak, India
Pankajgupta.vce@gmail.com

Abstract

Grid computing is a means of allocating the computational power of a large number of computers to complex difficult computation or problem. Grid computing is a distributed computing paradigm that differs from traditional distributed computing in that it is aimed toward large scale systems that even span organizational boundaries. This paper proposes a method to achieve maximum fault tolerance in the Grid environment system by using Reliability consideration by using Replication approach and Check-point approach. Fault tolerance is an important property for large scale computational grid systems, where geographically distributed nodes co-operate to execute a task. In order to achieve high level of reliability and availability, the grid infrastructure should be a foolproof fault tolerant. Since the failure of resources affects job execution fatally, fault tolerance service is essential to satisfy QOS requirement in grid computing. Commonly utilized techniques for providing fault tolerance are job check pointing and replication. Both techniques mitigate the amount of work lost due to changing system availability but can introduce significant runtime overhead. The latter largely depends on the length of check pointing interval and the chosen number of replicas, respectively. In case of complex scientific workflows where tasks can execute in well defined order reliability is another biggest challenge because of the unreliable nature of the grid resources.

Keyword: *Grid Computing, Checkpoint, Replication, Fault-tolerance*

1. Introduction

Grid computing enables aggregation and sharing of geographically distributed resources and data into a single virtual machine for solving the large-scale problems, which require more computational power. Grid can be used for many applications like medical imaging and diagnosis, biometrics, satellite image processing. Grid has many design issues like scheduling, data management, resource management, security, load balancing and fault tolerance. Grid jobs are very large and our grid environment is more likely to have failure, so fault management becomes more important to give a Quality of Service (QoS) like deadline and budget, to grid user. Due to large size of jobs, it may also be the case that the cost and difficulty of finding and recovering from faults in Grid applications is higher than normal applications. The term fault tolerance means to continue the work correctly in the presence of fault. Computational grids have become a popular approach to handle vast amounts of available information and to manage computational resources. Examples of areas where grids have been successfully used for solving problems include biology, nuclear physics and engineering. A Grid enables the sharing, selection, and aggregation of a wide variety of geographically distributed resources. Stand-alone system is prone to crash. These individual components in a distributed computing system may fail without stopping the entire computing system. Computational grid consists of large sets of diverse, geographically distributed resources that are grouped into virtual computers for executing specific applications. As the number of grid system components increases, the probability of failures in the grid computing environment becomes higher than that in a traditional parallel computing scenario. Compute intensive grid applications often require much longer execution time in order to solve a single problem. Thus, the huge computing potential of grids, usually, remains unexploited due to their susceptibility to failures like, process failures, machine crashes, and network failures etc this may lead to job failures, violating timing deadlines and service level agreements, denials of service, degraded user expected quality of service. Thus fault management is a very important and challenging for grid application developers. It has been observed that interaction, timing, and omission faults are more prevalent in grid.

2. Need for Fault-tolerance

Fault tolerance is an important property in Grid computing as grid resources are geographically distributed in different administrative domains worldwide. Also in large-scale grids, the probability of a failure is much greater than in traditional parallel systems. Since grid applications run in a very heterogeneous computing environment, fault tolerance is important in order to ensure their correct behavior. The development of correct grid applications is difficult with traditional software development methods. Hence, formal methods can be beneficial in order to ensure their correctness and structure their development from specification to implementation. Fault tolerance is the ability of a system to perform its function correctly even in the presence of faults. The fault tolerance makes the system more dependable. A

complementary but separate approach to increase dependability is fault prevention. This consists of techniques, such as inspection, whose intent is to eliminate the circumstances by which faults arise. A failure occurs when an actual running system deviates from this specified behavior. The cause of a failure is called an error. An error represents an invalid system state that does not comply the system specification. The error itself is the result of a defect in the system or fault. In other words, a fault is the root cause of a failure. However, a fault may not necessarily result in an error; nevertheless, the same fault may result in multiple errors. Similarly, a single error may lead to multiple failures.

Research on fault tolerance in the grid environment can divide its mechanism into two main types: pro-active and post-active. Proactive fault tolerance mechanism takes into account the failure of grid resource before scheduling jobs on grid resources, like replication approach. On the other hand, the post-active mechanism takes appropriate action after the job failure, like check pointing approach. So mainly there are two approaches, Replication base approach and Check-point base approach. Fault Tolerance approach requires three main steps, failure detection, failure notification and failure recovery. Failure detection is the phase between the failure occurrence and the time in which the failure is discovered. Failure notification is the phase between the failure detection and the instant in which nodes responsible for recovering the failure are notified. Failure recovery is the phase between failure notification and the time in which the pre-failure working conditions are recovered.

3. Issues

In a distributed real time system or in general, fault tolerance is the technique to give the required services in the presence of fault or error within the system. The aim is to avoid failures in the presence of faults and provide services as per requirement. In fault tolerance the fault is detected first and recovers them without participation of any external agents. The main issue in fault tolerance is how, where, and which technique is using to tolerate fault in distributed system. In any real time distributed system there are three main issues.

1. *Feasibility*- this means that a task running should be finished on its deadline even though there is a fault in the system. Dead line in real time system is the major issue because there is no meaning of such a task which is not finishing before its deadline. So the question is that which method is to be applied by which the task can finish on deadline in the presence of fault.
2. *Reliability*- in real time distributed system reliability means availability of end to end services and the ability to experience failures or systematic attacks, without impacting customers or operations.
3. *Scalability* –it is about the ability to handle growing amount of work, and the capability of a system to increase total throughput under an increased load when resources are added.

Now the question arise how these faults can be detected and removed or tolerated from different environment. A task running in distributed environment should be finished on its deadline. It may be hard or soft depend on task requirement. In hard deadline a task should be finished by its deadline sharply but in soft deadline task can finished nearby its deadline. Many types of fault and failure arise in a system, so there should be an appropriate method which can tolerate such problems. In this paper various technique for tolerating different fault are discussed.

4. Fault tolerance approaches

Grid computing has many fault tolerance approaches some of which are as follows.

4.1 Replication

It is a technique based on an inherent assumption that any single resource is much susceptible to failure as compared to simultaneous failure of multiple resources. Unlike check pointing the replication avoids task re-computation by executing several copies of the same task on more than one compute stations. The job replication and determination of the optimal number of replicas involves many technical considerations. The task replication in grids has been studied in. A number of approaches have been used to implement replication in grid computing environment. Replication, is the process of sharing information by which it can ensure consistency between redundant resources (i.e. software or hardware components), to improve reliability, fault-tolerance, or accessibility. Job replication is the method of replicating job on multiple servers such as in grid computing service is capable of receiving jobs, executing them, performing checksum operations on them, and sending the result back to the client. Fig.1 shows a distributed network in which every server S can communicate to each other and each server is connect to multiple clients C. A job can be distributed to servers for operation and result is back to the client. Data Replication is also commonly used by fault tolerance mechanisms to enhance availability in Grid like environments where failures are more likely to occur.

In this data is stored on multiple storage devices as a replica. Data replication may be synchronous or asynchronous depend upon data consistency. Components are replicated on different machines, and if any component or machine fail, then that application can be transferred and run on another machine

having the required components. Data Replication is also commonly used by fault tolerance mechanisms to enhance availability in Grid like environments where failures are more prone.

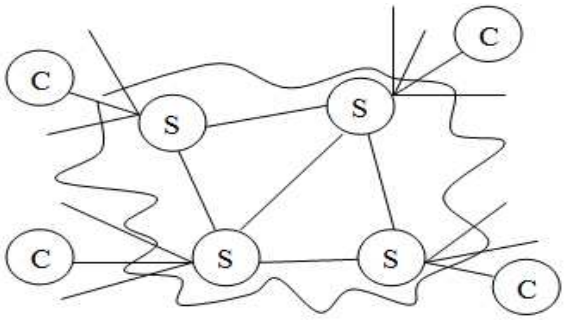

Fig. 1 Distributed System with multiple clients and servers.

4.2 Checkpoint

The checkpoint is one of the most popular techniques to provide fault-tolerance on unreliable systems. It is a record of the snapshot of the entire system state in order to restart the application after the occurrence of some failure. The checkpoint can be stored on temporary as well as stable storage. However, the efficiency of the mechanism is strongly dependent on the length of the checkpointing interval. Frequent checkpointing may enhance the overhead, while lazy checkpointing may lead to loss of significant computation. It is the process to saving from complete execution of a task. It checks the acceptance test, if fail then go to previous checkpoint instead of beginning. Hence, the decision about the size of the checkpointing interval and the checkpointing technique is a complicated task and should be based upon the knowledge about the application as well as the system. Therefore,various types of checkpointing optimization have been considered by the researchers, e.g., Full checkpointing, Incremental checkpointing, Unconditional checkpointing, Dynamic checkpointing, Synchronous and asynchronous checkpointing etc. A check point may be system level, application level, or mixed level depends on its characteristics. Check-pointing is also categorized on the basis of In-transit or orphan message. These are Uncoordinated Checkpointing, Coordinated Check-pointing, and Communication-induced Check-pointing. Check-pointing also can be classified is based on who instruments the application that do the actual capturing and re-establishing of the application execution state. These are Manual code insertion, Pre-compiler check pointing, Post-compiler check-pointing. A check point may be local or global on the basis of their scope. Check-point for separate process is local checkpoint and a check-point applied for set of processes is called global check-point. Check-pointing have some demerits such as Check-pointing causes execution time overhead even if there are no crashes. The cost of writing check-point data to stable storage whenever a check-point is taken is called the check-pointing cost.

5. Conclusion

In the light of above survey, fault tolerance plays an important role in order to achieve availability and reliability of a grid system. Replication and Check pointing are the major techniques used in any fault-tolerant grid management system. Replication provides better reliability and improved response time. In this paper it is tried to achieve the maximum fault tolerance by using Reliability consideration. Reliability defects generally are failures that might occur in the future inside a system that has been working well so far. The ability to checkpoint a running application and restart it later can provide many useful benefits like fault recovery, advanced resources sharing, dynamic load balancing and improved service availability. In case of dependent task grid (workflow grid) fault tolerance can be handled at two levels i.e. task level and workflow level.

It is very different to detect a fault in distributed system as compare with uniprocessor. Fault tolerance techniques are also depending upon its occurrence. In this paper various reliable fault detection and fault tolerance methods are explored. There are three things in real time distributed system which should be kept in mind when fault tolerance is applying. These are reliable, scalable and feasible. In real time distributed system feasibility of a task is much important because there is a dead line defined for every task and the task should be finished on or before its deadline even there is a fault in the system.

6. References

[1] Oren Laadan Jason Nieh, Transparent Checkpoint-Restart of Multiple Processes on Commodity Operating Systems, 2007 USENIX Annual Technical Conference

[2] J. H. Abawajy, Fault-Tolerant Scheduling Policy for Grid Computing Systems, 2004 IEEE

[3] Babar Nazir, Taimoor Khan, Fault Tolerant Job Scheduling in Computational Grid, 2006 IEEE, pp 708-713

[4]Mattern, F., Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation, Journal of Parallel and Distributed Computing, pp. 423-434, 1993

[5]D. Ryu, Quality, product quality, and market share increase, Int J Reliab Appl 2 (2001) (3), p. 163 174,182.

[6] Paul Stelling, Ian Foster, Carl Kesselman, Craig Lee, Gregor von Laszewski , A Fault Detection Service for Wide Area Distributed Computations

[7]Chandy, K. M., Lamport, L., Distributed Snapshots: Determining Global States of Distributed Systems, ACM
Transactions on Computer Systems, pp. 63-75, February 1985.

[8] I. Foster, C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann Publishers, Los Altos, CA, 1998.

[9] F. Berman, G. Fox, and T. Hey, "Grid Computing: Making the Global Infrastructure a Reality. Chichester", John Wiley & Sons, 2003

[10] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid", In Proceedings of 4th International Conference on High Performance Computing in Asia-Pacific Region, Beijing, China, 2000.

[11] K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, "Grid Information Services for Distributed Resource Sharing", In Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing, 2001 .

[12] I. Foster and C. Kesselman, S.T., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", Intl. J. Supercomputer Applications, 2001.

[13] E. Beguelin, Seligman and P. Stephan, "Application Level Fault Tolerance in Heterogeneous Networks of Workstations", Journal of Parallel and Distributed Computing on Workstation Clusters and Networked-based Computing, Vol. 43(2), 1997, pp. 147–155.

[14] K Limaye, B. Leangsuksun, Z. Greenwood, S. L. Scott, C. Engelmann, R. Libby and K. Chanchio,
"Job-Site Level Fault Tolerance for Cluster and Grid environments" In Proceedings of the IEEE international conference on cluster computing, 2005, pp. 1-9

[15] P. Stelling, I. Foster, C. Kesselman, C. Lee, G. von Laszewski, "A fault detection service for wide area distributed computations", In: Proceedings of 7th IEEE Symposium on High Performance Distributed Computing, 1998.

[16]. Wei Luo, Xiao Qin, Member, IEEE, Xian-Chun Tan, Ke Qin, and Adam Manzanares "Exploiting Redundancies to enhance Schedulability in Fault-Tolerant and Real-Time Distributed Systems" IEEE Transactions on system man and cybernetics – part A : systems and humans, VOL. 39, NO. 3, May 2009.

[17]. Louca, neophytou. Lachanas. And evripidou "MPI-FT: portable fault tolerance for MPI" In parallel processing latters Vol 10,no. 4 (2000) pg.371-382.

[18]. Alain Girault, Hamoudi Kalla, and Yves Sorel "An active replication scheme that tolerates failure in distributed embedded real time system"

[19]. Nitin B. Gorde, Sanjeev K. Aggarwal "A Fault Tolerance Scheme for Hierarchical Dynamic Schedulers in Grids" IEEE international Conference on Parallel Processing – Workshops -2008.