IJCSMS International Journal of Computer Science & Management Studies, Vol. 11, Issue 01, May 2011 64
ISSN (Online): 2231 –5268
www.ijcsms.com

GRID COMPUTING AND CHECKPOINT APPROACH

Pankaj gupta
Vaish College of Engineering, M.D.U, 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. In this paper we investigate the different techniques of fault tolerance which are used in many real time distributed systems. The main focus is on types of fault occurring in the system, fault detection techniques and the recovery techniques used. A fault can occur due to link failure, resource failure or by any other reason is to be tolerated for working the system smoothly and accurately. These faults can be detected and recovered by many techniques used accordingly. An appropriate fault detector can avoid loss due to system crash and reliable fault tolerance technique can save from system failure. This paper provides how these methods are applied to detect and tolerate faults from various Real Time Distributed Systems. The advantages of utilizing the check pointing functionality are obvious; however so far the Grid community has not developed a widely accepted standard that would allow the Grid environment to consciously utilize low level check pointing packages. Therefore, such a standard named Grid Check pointing Architecture is being designed. The fault tolerance mechanism used here sets the job checkpoints based on the resource failure rate. If resource failure occurs, the job is restarted from its last successful state using a checkpoint file from another grid resource. A critical aspect for an automatic recovery is the availability of checkpoint files. A strategy to increase the availability of checkpoints is replication. Grid is a form distributed computing mainly to virtualizes and utilize geographically distributed idle resources. A grid is a distributed computational and storage environment often composed of heterogeneous autonomously managed subsystems. As a result varying resource availability becomes common place, often resulting in loss and delay of executing jobs. To ensure good performance fault tolerance should be taken into account. Here we address the fault tolerance in terms of resource failure. Commonly utilized techniques to achieve fault tolerance is periodic check pointing, which periodically saves the jobs state. But an inappropriate check pointing interval leads to delay in the job execution, and reduces the throughput. Hence in the proposed work, the strategy used to achieve fault tolerance is by dynamically adapting the checkpoints based on current status and history of failure information of the resource, which is maintained in the Information server. The Last failure time and Mean failure time based algorithm dynamically modifies the frequency of checkpoint interval, hence increases the throughput by reducing the unnecessary checkpoint overhead. In case of resource failure, the proposed Fault Index Based Rescheduling (FIBR) algorithm reschedules the job from the failed resource to some other available resource with the least Fault-index value and executes the job from the last saved checkpoint. This ensures the job to be executed within the deadline with increased throughput and helps in making the grid environment trust worthy.

Keywords: *Grid Computing, Fault Tolerance, Checkpoint, Replication, Gridisim.*

1. INTRODUCTION

Grid computing is a term referring to the combination of computer resources from multiple administrative domains to reach a common goal. The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of files. What distinguishes grid computing from conventional high performance computing systems such as cluster computing is that grids tend to be more loosely coupled, heterogeneous, and geographically dispersed. Although a grid can be dedicated to a specialized application, it is more common that a single grid will be used for a variety of different purposes. Grids are often constructed with the aid of general-purpose grid software libraries known as middleware. Grid computing has emerged as the next-generation parallel and distributed computing methodology that aggregates dispersed heterogeneous resources for solving various kinds of large-scale parallel applications in science, engineering and commerce. A Grid enables sharing, selection, and aggregation of a wide variety of geographically distributed resources including supercomputers, storage systems, data sources and specialized devices owned by different organizations. Management of these resources is an important infrastructure in the grid computing environment. It becomes complex as the resources are geographically distributed, heterogeneous in nature, owned by different individual or organizations with their own policies, have different access models, and have dynamically varying loads and availability. Grid computing or the use of a computational grid, is applying the resources of many computers in a network to a single problem at the same time - usually to a scientific or technical problem that requires a great number of computer processing cycles or access to large

amounts of data. Compared to other distributed environments, such as clusters, complexity of grid mainly originates from decentralized management and resource heterogeneity. These characteristics often lead to strong variations in availability, which in particular depends on resource and network failure rates, administrative policies, and fluctuations in system load. Apparently, runtime changes in system availability can significantly affect job execution. Since for a large group of time-critical or time consuming jobs delay and loss are not acceptable, fault tolerance should be taken into account. Providing fault tolerance in a distributed environment, while optimizing resource utilization and job execution times, is a challenging task. To accomplish it, we use dynamic adaptation of checkpoints technique, based on current status of the job and failure history of the resource, which overcomes the checkpoint overhead that is caused by unnecessary checkpoints in case of periodic check pointing And hence achieves fault tolerance with increased throughput.

2. RELATED WORK

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.

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. 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 latter are often implemented through job check pointing in combination with migration and job replication. Although both methods aim to improve system performance in the presence of failure, their effectiveness largely depends on tuning runtime parameters such as the check pointing interval and the number of replicas. Determining optimal values for these parameters is far from trivial, for it requires good knowledge of the application and the distributed system at hand. 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. Whereas, Post-active mechanisms handles the job failures after it has occurred. Of those that look into these issues, many works are primarily post-active in nature and deal with failures through Grid monitoring.

3. TYPES OF FAULT

There are different types of fault which can occur in Real-Time Distributed System. These faults can be classified on several factors such as:

Network fault: A Fault occur in a network due to network partition, Packet Loss, Packet corruption, destination failure, link failure, etc.

Physical faults: This Fault can occur in hardware like fault in CPUs, Fault in memory, Fault in storage, etc.

Media faults: Fault occurs due to media head crashes.

Processor faults: fault occurs in processor due to operating system crashes, etc.

Process faults: A fault which occurs due to shortage of resource, software bugs, etc.

Service expiry fault: The service time of a resource may expire while application is using it.

A fault can be categorized on the basis of computing resources and time. A failure occurs during computation on system resources can be classified as: omission failure, timing failure, response failure, and crash failure.

Permanent: These failures occur by accidentally cutting a wire, power breakdowns and so on. It is easy to reproduce these failures. These failures can cause major disruptions and some part of the system may not be functioning as desired.

Intermittent: These are the failures appears occasionally. Mostly these failures are ignored while testing the system and only appear when the system goes into operation. Therefore, it is hard to predict the extent of damage these failures can bring to the system.

Transient: These failures are caused by some inherent fault in the system.

However, these failures are corrected by retrying roll back the system to previous state such as restarting software or resending a message. These failures are very common in computer systems.

4. ISSUES

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.

Grid computing operates on these technology principles.

1. Standardization-IT departments have enjoyed greater interoperability and reduced their systems management overhead by standardizing on operating systems, servers, storage hardware, middleware components, and network components. Standardizing also helps reduce operational complexity in the data center by simplifying application deployment, configuration and integration.

2. Virtualization-Virtualizing IT resources means that applications are not tied to specific server, storage, or network components and can use any virtualized IT resource. Virtualization occurs through a sophisticated software layer that hides the underlying complexity of IT resources and presents a simplified, coherent interface used by applications and other IT resources.

3. Automation-Because of the potentially large number of components—both virtual and physical—grid computing demands large-scale automation of IT operations. Each component requires configuration management, on-demand provisioning, top-down monitoring, and other management tasks. A grid management solution must ensure that infrastructure cost savings do not evaporate as a result of hiring additional staff to manage the grid. IT administrators need a top-down view from the end-user or application level so they can effectively measure service levels and proactively resolve problems. Combining these capabilities into a single, automated, integrated solution for managing grids gives organizations a maximum return on their grid investment.

5. PROBLEM FORMULATION

Grid jobs are executed by the computational grid as follows:

(i) Grid users submit their jobs to the grid scheduler by specifying their QoS requirements, i.e., deadline in which users want their jobs to be executed, the number of processors and type of operating system.
(ii) Grid scheduler schedules user jobs on the best available resource by optimizing time.
(iii) Result of the job is submitted to user upon successful completion of the job.

Such a computational grid environment has two major draw backs:
1. If a fault occurs at a grid resource, the job is rescheduled on another resource which eventually results in failing to satisfy the user's QoS requirement i.e. deadline. The reason is simple. As the job is re executed, it consumes more time.
2. In the computational based grid environments, there are resources that fulfill the criterion of deadline constraint, but they have a tendency toward faults. In such a scenario, the grid scheduler goes ahead to select the same resource for the mere reason that the grid resource promises to meet user's requirements of the grid jobs. This eventually results in compromising the user's QoS parameters in order to complete the job.
In this paper, in order to address the first problem, a job check pointing strategy is used to tolerate faults gracefully, as it is able to restore the partially completed job from the last checkpoint. In order to address the second problem, make the check pointing strategy adaptive by maintaining a fault index. This fault index is maintained by taking into consideration the fault occurrence history information of the grid resource. In this way, the checkpoint is introduced mostly when it is necessary. The application can only be restarted from the last known state, if the checkpoint is available. To increase the availability of checkpoint CRS is used. Using CRS, the current application state can be taken at any time. Simulation experiments reveal that the proposed strategy is able to tolerate faults by taking appropriate measures according to resource vulnerability toward faults.

6. CHECKPOINT APPROACH

The economy base grid is a user centric, resource management and job scheduling approach. It offers incentive and profits to resource owners as award of contributing their resources. On the other hand, it also provides user flexible environment to maximize their goal within their budget by relaxing QoS like deadline and budget. Fault tolerance in such environment is critical to consider because it effects the profit of both the

parties, but it become more important because the possibility of fault in grid environment is much higher than a traditional distributed system due to lack of centralized environment, predominant execution of long jobs, highly dynamic resource availability, diverse geographical distribution of resources, and heterogeneous nature of grid resources. Adaptive check-pointing fault tolerance approach is used in this scenario to overcome above mentioned drawbacks. In this approach, fault occurrence information is maintained for every resource. When a fault occurs, the fault occurrence information of that resource is updated. This fault occurrence information is used during decision making of allocating the resources to the job. This is implemented as fault index. Fault index indicates the resource vulnerability to fault i.e. higher the fault index, higher the failure rate. The fault index of a resource is increased every time when the resource fails to complete the assigned task within deadline and budget. Similarly the fault index is decreased every time when resource successfully completes the assigned job within deadline and budget.

The check pointing is one of the most popular technique 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 check pointing interval. Frequent check pointing may enhance the overhead, while lazy check pointing may lead to loss of significant computation. Hence, the decision about the size of the check pointing interval and the check pointing technique is a complicated task and should be based upon the knowledge about the application as well as the system. Therefore, various types of check pointing optimization have been considered by the researchers, e.g., (i) Full check pointing or Incremental check pointing (ii) Unconditional periodic check pointing or Optimal (Dynamic) check pointing (iii) Synchronous (Coordinated) or asysnchronous (Uncoordinated) check-pointing and (iv) Kernel, Application or User level check-pointing. 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. 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 Check-pointing, 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.

7. PERFORMANCE EVALUATION

The system models of these approaches are design and tested in GridSim Toolkit-4.0. The gridsim libraries are added to the platform of Eclipse, which is an integrated development environment (IDE) for java. The gridsim libraries are available freely as java runtime environment (jre), and they are linked to eclipse platform as external jre. Numbers of resources with different characteristics like cost, cpu rating, are used to design grid infrastructure for simulation purpose, as mention in World Wide Grid (WWG testbed). Different numbers of Gridlets are created to evaluate these approaches. Gridlet is define in term of length (in Million Instruction), input file size (in byte), and output file size. To simulate real world hetrogenous scenarios, the size of gridlets are varied from 3,000 to 20,000 MI at random and input file and output size are varied from 500 to 700 bytes at random.

In this experiment, 200 gridlets are submitted and available budget varies from 5000 to 17000. Total number of gridlet successfully completed is plotted in fig 1. To measure the performance, gridlets are assigned to grid in which no fault tolerance approach is used and to grid in which adaptive check-point fault tolerance approach is used. In both the scenario, first aim is to fulfill the budget parameter. In the scenario where no fault tolerance approach is used, all resources which satisfy budget parameter are treated equally and jobs are assign to any one of them. In adaptive checkpoint approach the resource who has minimum fault index is selected among the all the resources which satisfy the budget parameter. Fault occurrence history of resource in maintains at fault manager and updated every time when a gridlet return. Thus, the number of gridlet completed in adaptive check point base approach is better than the scenario where no fault tolerance approach is used.

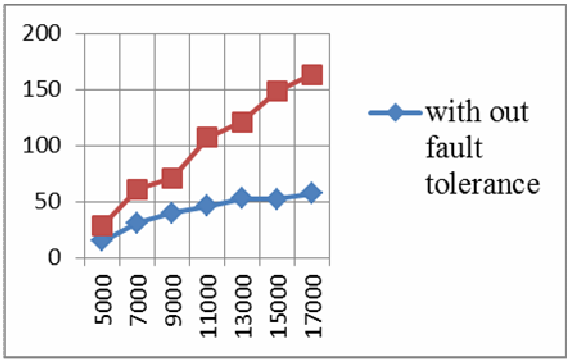

Fig1: Number of Gridlet Completed for different budget.

Numbers of failed jobs are also less in adaptive check-point approach. The sum of success job and failed job is not equal to total number of submitted jobs because there are some jobs for which no resource can satisfy the budget parameter, these jobs are consider as cancel jobs.

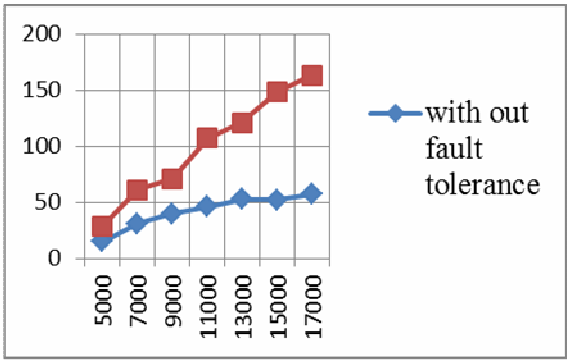

Fig2: Number of Gridlet Completed for different budget.

8. CONCLUSION

Fault tolerance forms an important problem in all distributed environments. Here we address the problem of fault tolerance in terms of resource failure. Thus the proposed work achieves fault tolerance by dynamically adapting the checkpoint frequency, based on history of failure information and job execution time, which reduces checkpoint overhead, and increases the throughput In this study, the problem of scheduling with fault tolerance strategy is proposed. Proposed scheme maintains history of the fault occurrence of resource in fault index manager. Before scheduling a job to a resource, checkpoint manager uses the resource fault occurrence history information and depending on this information, it sets different intensity of check pointing. The availability of checkpoint files is increased through replication. The performance of the proposed strategy is evaluated using GRIDSIM Toolkit . The experimental results demonstrate that proposed strategy effectively schedule the grid jobs in fault tolerant way in spite of highly dynamic nature of grid.

9. FUTURE WORK

In the future, it is planned to explore the potential of these scheduling strategies by embedding them into real world grid computing environments. Also it is planned to improve the checkpoint replication service by optimizing the recovery of checkpoint replica i.e. getting the replica in faster manner.

REFERENCE

[1]. Chtepen, M.; Claeys, F.H.A.; Dhoedt, B.; De Turck, F.; Demeester, P.; Vanrolleghem, P.A. Adaptive Task CHECKPOINTING and Replication: Toward Efficient Fault-Tolerant Grids Parallel and Distributed Systems, IEEE Transactions on Volume 20, Issue 2, Feb. 2009 Page(s):180 – 190 Digital Object Identifier 10.1109/TPDS.2008.93
[2]. Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, Eucalyptus: A Technical Report on an Elastic Utility Computing architecture Linking Your Programs To Useful Systems.UCSB computer science technical report number 2008-2010
[3] Favarim, F.; da Silva Fraga, J.; Lung Lau Cheuk; Correia, M..GRIDTS: A New Approach for Fault- Tolerant Scheduling in Grid Computing Network Computing and Applications, 2007. NCA 2007. Sixth IEEE International Symposium on Volume ,Issue,12-14 July 2007 Page(s):187–194 Digital ObjectIdentifier 10.1109/NCA.2007.27
[4]. Fangpeng Dong and Selim G. Akl January 2006 Scheduling Algorithms for Grid Computing:State of the Art and Open Problems. Technical Report No. 2006-504 School of Computing, Queen's University Kingston, Ontario
[5] Foster,I.; Yong Zhao; Raicu,I.; Lu,S; Grid computing and Grid computing 360-degree compared. Grid computing environments workshop,2008.GCE'08 12-16 Nov.2008 pages:1-10.
[6]Lars-Olof Burchard, C´esar A. F. De Rose, Hans Ulrich Heiss, Barry Linnert and J¨org Schneider. VRM: A Failure-Aware Grid Resource Management System. Proc. of the 17th

Intl: Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'05). IEEE. 2005
[7]Mohammad Tanvir Huda, Heinz W. Schmidt and Ian D. Peake. An Agent Oriented Proactive Fault tolerant Framework for Grid Computing. First International Conference on e-Science and Grid Computing (e-Science'05).IEEE. 2005
[8]R. Medeiros, W. Cirne, F. Brasileiro and J. Sauve, .Faults in Grids: Why are they so bad and What can be done abut it? in the proceedings of the Fourth Intl: Workshop on Grid Computing (GRID'03), 2003.
[9] Nazir, B.; Khan, T.Fault Tolerant Job Scheduling in Computational Grid. Emerging Technologies, 2006. ICET apos;06. International Conference on Volume , Issue, 13-14 Nov.2006 Page(s):708–713 Digital Object Identifier 10.1109/ICET.2006.335930
[10] D. Feitelson, Parallel Workloads Archive, http://www.cs.huji.ac.il/labs/parallel/workload/, 2008
[11] Jang-uk In,Paul Avery, Richard Cavanaugh. SPHINIX:A fault tolerant system for scheduling in dynamic environments,proceedings of the 19th IEEE international parallel and distributed processing symposim.
[12]www.gridsim.org
[13]gridsimulator.http://www.buyya.com/gridbus/gridsim/,relea sed on Apr 08, 2009
[14] S. Agarwal, R. Garg, M. Gupta, and J. Moreira, "Adaptive Incremental Checkpointing for Massively Parallel Systems," Proc.18th Ann. Int'l Conf. Supercomputing (SC '04), Nov. 2004.
[15] A. Subbiah and D. Blough, "Distributed Diagnosis in Dynamic Fault Environments," Parallel and Distributed Systems, vol. 15, no. 5,pp. 453-467, 2004.