AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

An Analysis on Checkpoint Mechanism for Grid Computing

[1]Aghila Rajagopal and [2]M.A. Maluk Mohamed

[1]Anna University, K.L.N. College of Information Technology, Information Technology, Aghila Rajagopal, Box.3030.Sivagangai, India.
[2]Anna University, M.A.M. College of Engineering, Information Technology, M.A.Maluk Mohamed, Box.3030.Tiruchirappalli. India.

ARTICLE INFO

Article history:
Received 25 December 2013
Received in revised form 22 February 2014
Accepted 26 February 2014
Available online 15 March 2014

Keywords:
Fault Recovery, Checkpoint, heterogeneous

ABSTRACT

Background: A computational grid is to flexibly handle computation management, data movement, storage management and other infrastructure that manifest to access many systems without restricting themselves to specific hardware and requirements. Fault tolerance and scheduling are expected to be vital challenges in Grid computing. It is because the dependability of individual Grid resources might not be guaranteed; also as resources are used outside of organizational boundaries, different scheduling instances for arbitrary Grid resources are supported. In existing system, rollback recovery techniques are used. But it was not resilient to all possible failure configurations. Our paper aims to provide an analysis of fault tolerant mechanism. The Primary ingredient of Grid Checkpoint Recovery service is recoverability of jobs among heterogeneous Grid resources. In essence, resources on which jobs are check pointed need not be of the same type as those on which the jobs are recovered, as long as the application code operating on the check pointing resource can be built for and run on the recovery platform. However there can be some circumstances even if the checkpoint failures there arises a question on recovery. Our paper provides a strategic solution for handling this erroneous situation of Checkpoint Failure. **Objective:** An Analysis On Checkpoint Mechanism For Grid Computing. **Results:** As discussed earlier, the factors that are to be considered while focusing on checkpoint algorithm are checkpoint overhead, control information, domino effect, orphan message, scalability, memory consumption, contention in accessing the stable storage etc. **Conclusion:** In this paper, we provided a brief introduction to the research field concerned with Checkpoint. As we observed, all the Existing Checkpoint Based Recovery Mechanism depends on the checkpoints stored. If the storage goes off, the performance of the system degrades. Here we come out with a solution. Our idea is to maintain a replica for the checkpoint. Though the solution is simple, it addresses the severe problem. Further Research is needed to consolidate the conceptual foundations of this approach.

© 2014 AENSI Publisher All rights reserved.

To Cite This Article: Aghila Rajagopal and M.A. Maluk Mohamed., An Analysis On Checkpoint Mechanism For Grid Computing. ***Aust. J. Basic & Appl. Sci.,*** *8(2): 43-48, 2014*

INTRODUCTION

Grid is a collection of computing nodes to achieve high end computational capabilities to perform a task by dividing a given task into subtasks. A subtask may run for several hours or days on a number of computing nodes. Grid computing has been used in many scientific applications. A grid system manages a large number of heterogeneous resources.

In (Baker, M., 2002) the grid is also defined as “A type of parallel and distributed system that enables the sharing, selection and aggregation of geographically distributed autonomous and heterogeneous resources dynamically at runtime depending on quality of service requirements”. It enables users to share a large number of distributed computing resources over a network. Nowadays, grid computing has been widely accepted, researched, and given attention to by researchers (Foster, I., 2002).

Resources are continuously appearing, disappearing in the grid for sharing their capabilities with others and they communicate via message passing. Hence periodic saving the state of processes is required to achieve Fault Tolerance.

In a large grid computing environment, failures occur very frequently. The failure rate is proportional to the number of processors. Hence a large grid computing systems should have some Fault Tolerance mechanism to allow reliable execution. Failures may be due to hardware or software failures. Generally, there are four major means for achieving reliable system: Fault Prevention, Fault Tolerance, Fault Removal, and Fault Forecasting (Algirdas Avizienis, 2004).

Corresponding Author: Aghila Rajagopal, Anna University, K.L.N. College of Information Technology, Information Technology, Aghila Rajagopal, Box.3030.Sivagangai, India.

Fault Prevention, prevents the occurrence or introduction of faults. Fault Tolerance avoids service failures in the presence of faults. Fault removal reduces the number and severity of faults. Fault forecasting, estimate the present number, the future incidence, and consequences of faults. Fault Tolerance which aims at delivering a correct service even in the presence of faults becomes a preferred choice for reliable Grid service.

Various research efforts have been made to provide Fault Tolerance Mechanism. Our paper major focuses on Fault Tolerance in Grid. In Particular checkpoint Mechanism for Fault Tolerance was analyzed. When checkpoint is introduced in Grid, the major problem lies in Grid heterogeneity nature. Many checkpoint Algorithms has been proposed in Grid on considering its heterogeneity nature. This Paper discusses the various Checkpoint Mechanism in Grid.

The paper is organized as follows. Section 2 discusses the various Fault Tolerant Techniques in Grid. Section 3 highlights the related work in Check pointing. Analysis of Checkpoint Mechanism in Section 4. The Paper ends in section 5 with the Conclusion.

Fault Tolerance in Grid:

Providing Fault tolerance is a challenging task. Fault may cause a huge loss of money and time. Fault Tolerance techniques enable systems to perform tasks even in the presence of faults. Performance, Scalability, Robustness, Transparency, Efficiency and Consistency etc are the important criteria in Fault Tolerance. Failure of a single node does not result in failure of the entire system. But as the number of node failure increases, a recovery mechanism is very important.

Fault Tolerance can be broadly classified as Static and Dynamic. Static Fault Tolerance mechanism is no more useful due to the highly dynamically adapting environment. One of way by which a fault tolerance technique can be made dynamic is by an adaptive programming model.

Grid systems failures can result not only from an error on a single node but also from the interaction between the nodes. Grid systems are extremely dynamic with nodes joining and leaving the system at any time.

Two approaches Push and Pull models. In push model grid nodes periodically send a heartbeat message announcing that they are alive. If the heartbeat message is not received for a time interval, it is decided that the node has failed. In pull model, the detector sends a request “are you alive” periodically to the grid nodes. Both the models result in network overhead.

Fault Tolerance can be classified as Pro active and Post active management. In Pro active mechanism, the jobs are handled with hopes that the jobs do not fail, whereas in Post active mechanism, jobs are handled after the job failure.

Failure of a node has a cascading effect on the performance of the grid. It reduces the performance of the computation which in turn the system. A cluster head is responsible to coordinate the activities. It results in a single point of failure. If a cluster head fails, a substitute has to be made.

Frequent failure handling, single point failure avoidance and fault tolerance capability are the criteria for deciding the performance of the fault tolerant techniques (Richard Koo and Sam Toueg, 1987).

Fault Tolerant Techniques:

The basic Fault Tolerant Techniques are

Retrying:

As the application fails, it is restarted from the scratch. It is very simple. Disadvantage of this system is the loss of computation time.

Replica:

A number of copies are kept in different nodes. All replicas are active and perform the same with same input parameters at different nodes.

Replication based Technique is one of the popular fault tolerance techniques (Gorender, S. and M. Raynal, 2007). The word replica means many copies. It is a process of maintaining different copies of a data item. The request from client is sent to anyone of the replica. The disadvantage is redundancy. Since the failure of some nodes will not result in failure of the system, Fault tolerance is achieved. The Static replication means that when some replica fails, it is not replaced by a new one, whereas in dynamic replication new replicas can be generated during run time.

The state of replicas is kept closely synchronized, replicas service the same requests in parallel and undergo the same state transitions. This algorithm is referred to as the active replication (Foster, I., 2002). Other replicas are kept as standby and can take over in the case of a primary failure (Algirdas Avizienis, 2004). It is called Passive replication.

Checkpoint:

It periodically saves the process state in a stable storage during the execution when failure is encountered; it restarts from one of its saved state.

Australian Journal of Basic and Applied Sciences, 8(2) February 2014, Pages: 43-48

Only if the time taken to recover from Checkpoint is less than the time taken to execute transaction from scratch, the recovery mechanism is useful. Otherwise it leads to overhead.

So the Checkpoint recovery scheme can be applied to long running jobs. Short running jobs can be resubmitted from the scratch. Some short running jobs need to be recovered when some input is from a sensor sensing a time dependent parameter.

Fusion Based Technique:

In fusion based tolerance a back-up machine is used. A back-up machine is a cross product of original computing machines. Overhead is high. It can be applied when fault rate is low.

A checkpoint is a saved local state of a process. Saved local state has to be a consistent state. On failure, when process restarts from a inconsistent checkpoint, it may cause the following two problems (Cristian, F. and F. Jahanian, 1985).

Checkpoint Mechanism in Distributed System:

Types of Checkpoint:

System Level Checkpoint:

The state of the process is written out to a file or a network socket. The information like state of processor's registers Stack and Memory. It needs to be restarted on the same type of hardware platform.

Application Level Checkpoint:

The application program is written in a way such that it stores and restores state. It can be restarted on totally different platform.

Methods of Taking Checkpoint:

Various Checkpoint methods are introduced with some changes in the existing methods.

A Full Checkpoint stores the total state of the application to a local storage occasionally. The storage required is very large.

In incremental Checkpoint, only the modified pages are stored instead of the whole process state.

In Coordinated Checkpoint, global Checkpoint is periodically sent by the Central Manager to all the nodes. After receiving the acknowledgement from all the nodes, Checkpoints are taken. Lots of time is consumed.

In Uncoordinated Checkpoint, checkpoints are generated independently at their local sites. Message passing interface is used for communication between the nodes. Checkpoints can be stored at their local sites.

The Creation of Checkpoints can be initiated by work stealing or at specific Check pointing periods. Checkpoints resulting from work stealing are called Forced Checkpoints. Periodic Checkpoints are called Local Checkpoints.

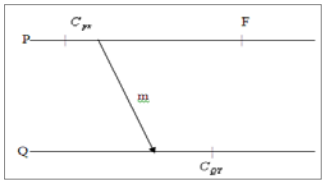
Communication induced checkpoint piggybacks control information on application messages. Each process has some basic check pointing, inspite of the additional forced check pointing.

Communication induced checkpoint algorithm is classified into two categories, Model based and index based. Model based have a deterministic behavior for each process. Index based algorithm associate each local checkpoint with a sequence number and try to maintain consistency among local checkpoints. It ensures domino free roll back. Less overhead.

Checkpoints Algorithms are broadly classified into Blocking and Non Blocking Algorithms. Blocking Algorithms blocks the underlying computation during check pointing. (Deng, Y. and E.K. Park, 1994; Kim, J.L. and T. Park, 1993; Koo, R. and S. Toueg, 1987; Bhargava, B., 1990) blocking algorithms was discussed. Blocking algorithm degrade system performance (Elnozahy, E.N., 1992; Roberto Baldoni, 1990).

Non Blocking algorithm (Roberto Baldoni, 1990) does not block the underlying computation during check pointing.

Consider Case 1:

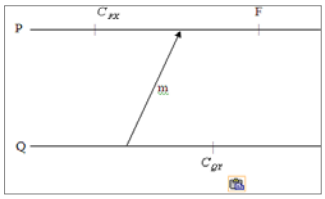


- P and Q are process
- P takes checkpoint at time X and sends a message m to Q.
- Now Q takes a Checkpoint at time Y
- P fails at F.
- P restarts from the checkpoint at X.

- P's local state shows that no message has been sent to Q.
- Q's local state shows that a message from has been received
- Thus inconsistent persist.

In order to overcome the above problem, both the process can be rolled back.

Case 2:



- Process P takes a checkpoint at time X.
- Q sends a message m to Process P and takes a checkpoint at time y.
- P receives m and fails at F
- Now P and Q rollback and restarts from X and Y respectively.
- Q is in a state which has already sent m and P is in a state which has not received any message from Q
- Thus message m is lost.

Hence the Checkpoints State should be a consistent State.

Need for Checkpoint Recovery in Grid:

High-computing scientific-research, bioinformatics and similar fields need to access and process a large amount of data. Failures of hardware or software service at processing site may degrade the performance of the systems and consume more time. If checkpoint facility is not available the subtasks have to rollback from the scratch. It is a tedious job and also consumes more time. Checkpoint stores the process state periodically in a stable storage. Check pointing in grid is important due to the following reasons,

Grid is prone to failures

Grid has to be managed

Efficiency and Resource Utilization

Related Work:

Overhead can be reduced by minimizing the number of synchronization message and the number of checkpoints and the second approach is to make the process Non Blocking. The number of forced checkpoints is reduced by means of index based check pointing algorithm which reduces the checkpoint overhead (Nitin, H. Vaidya, 1999). The numbers of forced checkpoints are mainly due to the fast increase in the sequence number. The sequence number has to be reduced. In order to reduce the sequence number, an equivalence relation is defined between the successive checkpoints of a process. A recovery line is allowed without increasing its sequence number. Equivalence between checkpoints can be detected by embedding the mechanism. Piggybacks the message N+1 integers as control information. Results shows that about 30% of checkpoint overhead is reduced.

Prakash Singhal Algorithm (Jean Michel Helary, 1999) forces only a minimum number of processes to take checkpoints, but it does not block the underlying computation. When process receives first request it takes local checkpoints and continues. These local checkpoints from a global checkpoint i.e. state information collected by each independent checkpoint is combined. Therefore the amount of computation lost during rollback is minimized. This non blocking algorithm is efficient, requires minimum number of tentative checkpoints and avoids avalanche effect.

Checkpoints can be staggered in any desired manner by using Staggered Consistent Checkpoint Algorithm (Tamir, Y. and C.H. Sequin, 1984). It assures that all processes share a single stable storage. The processes may share multiple stable storages. The process may be grouped into clusters being identical to the number of stable storage. Each cluster with a unique stable storage. Algorithm ensures that physical checkpoint taken by various process are staggered. It minimizes the contention in accessing the stable storage. Algorithm that staggers more would tend to perform poorly when degree of synchronization and message volume is large. Performance of stagger can be improved by reducing the amount of information logged.

Transient problems also have to be addressed. It is solved by using Check pointing and Rollback Recovery Algorithm. The process finds a consistent checkpoint among the set of saved checkpoints. Then it can be roll backed and restarted. It results in Domino effect. Another approach is to save only the recent checkpoints (Prakash, R. and M. Singhal, 1996). Whenever a process rollback, it notifies all other processes to rollback. Consistent state can be reached even after transient failures by using two phase commit protocols. Initiator takes tentative checkpoint and requests all processes to take tentative checkpoints. If Q learns that all process have

taken tentative checkpoints, Q decides it to make as permanent or discards. Q decision is propagated and carried out all processes. Process keeps a variable to denote its willingness to take checkpoints.

Various consistency issues like Transitlessness and strong consistency issues in distributed system was discussed by Jean Michel Helary and Robert Netzer.

As the research focuses mainly on optimizing checkpoint operation, Yawei Li and Zhiling Lan concentrated on restart Mechanism. They presented FREM a Fast restart mechanism for checkpoint restart protocol.

A checkpoint free tolerance approach was also emerged. It is an algorithm based approach in which a coded global consistent state of the application data is maintained in memory by modifying applications to operate on encoded data. Results show that their approach is able to survive process failure with a very low overhead.

Analysis of Checkpoint Mechanism:

On analyzing the various Fault Tolerance Mechanism, we can conclude that for short jobs, basic methods such as Retrying or Replication can be used and for long jobs checkpoint mechanism can be used. Though there are different mechanisms available for fault tolerance, the most preferred mechanism by everyone is the Checkpoint mechanism. By making incremental changes to the existing checkpoint mechanism, some additional methods are introduced. They differ in the way how the checkpoints are taken. The main criteria that must be taken into account while choosing the best Fault tolerance mechanism are Memory, Execution Time of the process, handling failures i.e Detection and Recovery. A detector must detect all the faults as early as possible. Recovery Time must be very less.

Several Check pointing algorithms are available in the literature, but it is very difficult to compare their performances. Some algorithms work better for some systems, while some other work fare better. In this paper, we provided a brief introduction to the research field concerned with Checkpoint.

As discussed earlier, the factors that are to be considered while focusing on checkpoint algorithm are checkpoint overhead, control information, domino effect, orphan message, scalability, memory consumption, contention in accessing the stable storage etc.

Conclusion:

Fault Tolerance must be capable of finding failures and handling failures i.e Detection and Recovery. A detector must detect all the faults as early as possible. Recovery Time must be very less.

In this paper, we provided a brief introduction to the research field concerned with Checkpoint. As we observed, all the Existing Checkpoint Based Recovery Mechanism depends on the checkpoints stored. If the storage goes off, the performance of the system degrades. Here we come out with a solution. Our idea is to maintain a replica for the checkpoint. Though the solution is simple, it addresses the severe problem. Further Research is needed to consolidate the conceptual foundations of this approach.

REFERENCES

Algirdas Avizienis, Jean Clauide Laprie, Brian Randell and Carl Landwehr, 2004. “Basic Concepts and Taxonomy of Dependable and Secure Computing”, 1(1).

Baker, M., R. Buyya and D. Laforenza, 2002. “Grid and Grid Technologies for wide area Distributed Computing in Journal of Software-Practice and Experience, 32(15): 1437-1466.

Bhargava, B., S.R. Lian and P.J. Leu, 1990. ”Experimental Evaluation of Concurrent Checkpointing and rollback Recovery Algorithms”, Proc. Int ‘I Conf Data Eng., pp: 182-189.

Cristian, F. and F. Jahanian, 1985. ”A Timestamp Based Checkpointing Protocol for Long Lived Distributed Computations,”ACM Trans. Computer Systems, Feb.

Deng, Y. and E.K. Park, 1994. “Checkpointng and Rollback Recovery algorithms in Distributed System, J.Systems and Software, pp: 59-71.

Elnozahy, E.N., D.B. Johnson and W. Zwaenepoel, 1992. ”The performance of Consistent Checkpointing”, Proc. 11th Symp. Reliable Distributed Systems, pp: 86-95.

Foster, I., 2002. “The Grid: A new Infrastructure for 21st Century Science”, Physics Today, 55(2): 42-47.

Gorender, S. and M. Raynal, 2007. “An Adaptive Programming Model for Fault Tolerant Distributed Computing” IEEE Transactions on Dependable And Secure Computing, 4(1).

Ian T. Foster, Carl Kesselman and Steven Tuecke, 2001. “The anatomy of the grid enabling scalable Virual Organisations”, CORR, cs.AR/0103025.

Jean Michel Helary and Robert H.B. Netzer, 1999. “Consistency issues in Distributed Checkpoints”, IEEE Transactions on Software Engineering, 25(2).

Kim, J.L. and T. Park, 1993. ”An Efficient Protocol for Checkpointing Recovery in Distributed System,”IEEE Trans. Parallel and Distributed Systems, 5(8): 955-960.

Koo, R. and S. Toueg, 1987. “Checkpointing and Rollback Recovery for Distributed Systems,”IEEE Trans. Software Eng., 13(1): 23-31.

Nitin, H. Vaidya, 1999. "Staggered Consistent Checkpointing", IEEE Transactiond on Parallel and Distributed Systems, 10(7).

Prakash, R. and M. Singhal, 1996. "Low Cost Checkpointing and failure Recovery in Mobile Computing Systems, IEEE Trans. Parallel and Distributed System, 7(10).

Richard Koo and Sam Toueg, 1987. "Checkpointing and Rollback Recovery for Distgributed Systems", IEEE Transactions on Software Engineering,vol SE-13, No. 1 January.

Roberto Baldoni, Francesco Quaglia and Paolo Fornara, 1990. "An Index Based Checkpointing Algorithm for autonomous Distributed Systems", IEEE Transactions on Parallel and Distribute Systems, 10(2).

Sanjay Bansal, Sanjeev Sharma, Ishita Trivedi, "A Detailed Review of Fault-Tolerance Techniques", International Journal on Internet and Distributed Computing Systems, 1(1).

Tamir, Y. and C.H. Sequin, 1984. "Error recovery in multicomputers using global checkpoints, " in Proc. 13th Int Conf. Parallel Processing.

Yawei Li and Zhiling Lan, 2011. "FREM: A Fast Restart Mechanism for General Checkpoint / Restart", IEEE Transaction on Computers, 60(5).