IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. III (Mar-Apr. 2014), PP 120-122
www.iosrjournals.org

A Survey on Fault Tolerance Mechanisms for job scheduling in Grid computing

S.Supriya[1], S.Dinesh Babu[2]
PG scholar[1,2], SNS college of engineering, Coimbatore

Abstract: *Grid computing is defined as a hardware and software infrastructure that enables sharing of coordinated resources in a dynamic environment. In grid computing, the probability of a failure is much greater than parallel computing. Therefore, the fault tolerance is an important issue in order to achieve reliability, availability of resources. When scheduling a job, the resource uses both average failure time and failure rate of grid resources combined with resources response time to generate scheduling. There are several reasons for failure in execution such as network failure, resource overloading, or non-availability of required software components. Thus, fault-tolerant systems should be able to identify and rectify the failures and support reliable execution in the presence of failures.In this paper, a survey made on various fault tolerance techniques and mechanism and job management in grid computing.*

Keywords:*Fault-tolerance, Grid scheduling, Checkpointing Recovery.*

I. Introduction

Grid is a system that coordinates resources that are not subject to centralized control, it is a shared heterogeneous computing and data resources distributed across network boundaries. Management of these resources becomes complex as the resources are geographically distributed, heterogeneous in nature, owned by different individual or organization have their own policies with varying loads and availability.The most important issue in scheduling a job to grid resource is it must obtain the Quality of Service (QoS) in existence of resource faults. Grid computing is the federation of computer resources from multiple administrative domains to reach a common goal. A grid is a collection of resorces sometimes referred to as nodes, machines, donors, hosts, and many other terms.

In computational grid, fault tolerance is preserving the delivery of required services despite the presence of fault-caused errors within the system, it enables the grid to continue its work in spite of one or more resources fail. Thus fault-tolerant service must be able to detect errors and rectify them efficiently.

II. Problem Statement

When grid users submit their jobs to the scheduler by specifying their QoS requirements in which users want their jobs to be executed. Grid scheduler schedules user jobs on the best available resource by optimizing time, as a result of the job is submitted to user upon successful completion of the job.

Such a computational grid environment has major draw backs:

If a fault occurs at a grid resource, the job is rescheduled on another resource which results in failing to satisfy the user's QoS requirement since job is rescheduled, it consumes more time.

In this paper, in order to solve this problem a job checkpointing strategy is used to tolerate faults as it is able to restore the partially completed job from the last checkpoint by maintain fault history information of the grid resource, hence checkpoint is used effectively.

III. Fault Tolerance Strategies

The Two main strategies of fault tolerance are: *Fault masking* is the process of preventing faults in a resource. It is used to hide the occurrence of faults and prevent faults from resulting in errors.*Reconfiguration* is the process of eliminating faulty component from a resource and restoring the resource to some operational state. Reconfiguration process are:

- Fault detection is the process of detecting whether fault has occurred that is required before any recovery procedure is initiated.
- Fault location is the process of determining where a fault has occurred so that an appropriate recovery can be applied.
- Fault containment is the process preventing the fault from propagating throughout the entire system.
- Fault recovery is the process of regaining the resource from the failure

3.1 Types of checkpointing

- *Incremental Checkpoint*- An incremental checkpoint is a checkpoint mechanism which occasionally stores the total state of the application to a local storage. The drawback of this checkpoint is that it consumes more time and also required very large storage to save.
- *Uncoordinated or Coordinated Checkpointing*- In coordinated checkpointing, processes have consistent checkpoints to ensure their saved states are consistent with each other. In contrast, in uncoordinated checkpointing, processes schedule checkpoints independently at different times.
- *Low Level Checkpointing*- In low levelcheckpointing procedures are included in the kernel, checkpointing is transparent to the user and no changes are required to the programs to make them checkpointable. When the system restarts after failure kernel will be responsible for managing the recovery operation. Integrating low level checkpoint packages with the grid.
- *User Level Checkponting*- In user level consists of library application programs are linked to this library for checkpointing, it does not require any changes in the application code, however explicit linking is required with user level library for recovery from failure.
- *Application Level Checkpointing*– In application level, it is responsible for carrying out all the checkpointingoperations. Code for checkpointing and recovery from failure is written into the application. It is expensive to implement.

IV. Checkpointing Algorithms

Silva L et al. [4] proposed Global checkpointing for all process in distributed systems. It is achieved by piggybacking monotonically increasing checkpoint number along with computational message. When a process receives a computational message with the high checkpoint number, it consider that checkpoint before processing the message. If each process allowed to initiate the checkpoint operation, the network may be flooded with control messages and unnecessary checkpoints. In order to avoid this, the proposed event allows one process to initiate checkpointing. The checkpoint event changes periodically by a local timer mechanism. When this timer expires, the initiator process broadcast checkpoint message to all other process.

Chandy et al.[7] proposed a global snapshot algorithm for distributed systems. It is observed that every checkpointing algorithm proposed for message passing system.The global state is constructed by coordinating all the resources and logging the channel state at the time of checkpointing. Special messages called markers are used for coordination and for identifying the messages originating at different checkpointing intervals.

Prakash-Singhal et al.[6] proposed low-level checkpointing algorithm was to combine two approaches. More specifically, it forces only a minimum number of processes to take checkpoints and does not block the underlying computation during checkpointing. It forces only part of processes to take checkpoints, the carrier sense of some processes may be out-of-date, and may not be able to avoid inconsistencies. Therefore this algorithm attempts to solve this problem by having each process maintains an array to save the problem.

Koo-Toueg's et al.[3] proposed a minimum process blocking checkpointing algorithm for distributed systems. The algorithm consists of two phases. During the first phase, the checkpoint initiator identifies all resources with which it has communicated since the last checkpoint and sends them a request. Upon receiving the request, each resource in turn identifies all resource it has communicated with since the last checkpoint and sends them a request, and so on, until no more resource can be identified. During the second phase, all resource identified in the first phase take a checkpoint.

J.L. Kim et al.[5]developed a new efficient synchronized checkpointing protocol which exploits the dependencies between processes in distributed systems. In this protocol, a process takes a checkpoint where all other processes agrees for same checkpooint and hence the process need not always wait for the decision made by the checkpointing coordinator as they are conventional synchronized protocols.

Cao et al. [9] proposed the concept of Mutable checkpoint which isneither a tentative checkpoint or a permanent checkpoint, to design efficient checkpointing algorithms for mobile computing systems. Mutable checkpoints can be saved anywhere, e.g., the main memory or local disk of MHs. In this way, taking a mutable checkpoint avoids the overhead of transferring large amounts of data to the stable storage, this minimize the number of mutable checkpoints.

D.V. SubbaRao et al. [8]proposed checkpointing algorithm combined with selective sender based message logging. This algorithm is free from problem of lost messages. This algorithm tolerates permanent faults in the presence of other processors. In their absence it tolerates only transient failures. The term selective implies that messages are logged only within a specified interval known as active interval, thereby reducing message logging overhead. This algorithm minimizes different overheads like checkpointing overhead, message logging, recovery and blocking overhead.

V. Conclusion

In computational grid, the fault detection and fault recovery is a very important task that to be addressed. The need for fault tolerance increases as the number of processors and the duration of computation increases. Checkpointing mechanism is an approach to reduce the failure recovery time. A survey on fault tolerancecheckpointing algorithms for Computational grid shows that a large number of papers have been published. Some of the scheduling algorithms with fault-tolerance may result in minimized makespan.

References

[1] DheerajBhardwaj. Grid Computing Concepts, Applications, anTechnologies Department of Computer Science and Engineering Indian Institute of Technology, Delhi

[2] Paul Townend and JeiXu, “Fault Tolerance within a Grid Environment” Proceedings of AHM2003

[3] Koo. R. and S.Toueg..Checkpointing and Rollback-Recovery for Distributed Systems. .IEEE Transactions on Software Engineering, SE- 13(1):23-31, January 1987.

[4] Silva L, Silva J 1992 Global checkpointing for distributed programs. Proc. IEEE 11th Symp.On Reliable Distributed Syst. pp 155-162.

[5] J.L. Kim and T. Park. "An efficient protocol for checkpointing recovery in Distributed Systems" IEEE Transaction On Parallel and Distributed Systems, 4(8):pp.955-960, Aug 1993.

[6] Prakash R. and SinghalM.Low-Cost Checkpointingand Failure Recovery in Mobile Computing Systems. ,IEEE Transaction On Parallel and Distributed Systems, vol. 7,no. 10, pp. 1035- 1048, October1996.

[7] Chandy K. M. and Lamport L., "Distributed Snapshots: Determining Global State of Distributed Systems," ACM Transaction on Computing Systems, vol. 3, No. 1, pp. 63-75, February 1985.

[8] D.V. SubbaRao and MM Naidu: A new, efficient corrdinatedcheckpointing protocol combined with selective sender based message logging, IEEE, 2008, Page(s): 444 – 447.

[9] G.Cao and M. Singhal. "On impossibility of Min- Process and Non-Blocking Checkpointing and An Efficient Checkpointing algorithm for mobile computing Systems". OSU Technical Report #0SU-CISRC-9/97-TR44, 1997, pp 37-44