International Journal of Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.05,Issue.11,
May-2016,
Pages:2229-2231

www.ijsetr.com

Analysis and Design of Check Pointing Algorithm for Mobile Distributed System

HURMAT SHAHIN[1], NIDHI SHARMA[2]

[1]PG Scholar, Dept of CSE, NIMS University, Jaipur, India.

[2]Assistant Professor, Dept of CSE, NIMS University, Jaipur, India.

Abstract: A distributed system is a collection of computers that are spatially separated and do not share a common memory and global clock. The existence of mobile nodes in a distributed system introduces new issues that need proper handling while designing a check pointing algorithm for such systems. These issues are mobility, disconnections, finite power source, vulnerable to physical damage, lack of stable storage etc. The location of an MH within the network, as represented by its current local MSS, changes with time. Therefore, we propose that in the first phase, all concerned MHs will take ad hoc checkpoint only. In case of an MH, ad hoc checkpoint is stored on the memory of MH only. In this case, if some process fails to take checkpoint in the first phase, then MHs need to abort their ad hoc checkpoints only. In this way, we try to minimize the loss of check pointing effort when any process fails to take its checkpoint in coordination with others.

Keywords: Globally Consistent Checkpoints (GCC), MSS.

I. INTRODUCTION

A distributed system is a collection of independent entities that cooperate to solve a problem that cannot be individually solved. A distributed system can be characterized as a collection of mostly autonomous processors communicating over a communication network. In a distributed system, there exists no system wide common clock (global clock). In other words, the notion of global time does not exist. Due to the absence of global time and shared memory, it is difficult to reason about the temporal order of events in a distributed system. Hence, algorithms for a distributed system are more difficult to design and debug compared to algorithms for centralized systems. In addition, the absence of a global clock makes it harder to collect up to- date information on the state of the entire system. The World Wide Web is used by millions of people daily for an assortment of purposes including email, reading news, downloading music, online shopping or simply accessing information about anything. Using a customary web browser, the user can access information stored on Web servers sited wherever on the sphere.

II. IMPLEMENTATION

Khatri Y. proposed a message- induced check pointing scheme with the exception that processes will not take any basic checkpoints. A process will take a checkpoint only when it receives a message from the other process. Here apart from this message-induced checkpoint, a new kind of forced checkpoint is incorporated to ensure a small re-execution of the processes after recovery from a failure. The proposed algorithm runs periodically to find out the set of globally consistent checkpoints (GCC). In case of failure, participating processes just need to roll back to their previous checkpoints. Recovery is as simple as in case of synchronous approach and further no coordination is required among processes. Proposed the concept of Soft Checkpoint based Coordinated Checkpoint protocol. In this algorithm Suppose, during the execution of the check pointing algorithm, Pi takes its checkpoint and sends m to Pj.

III. DISCUSSION

A. Algorithm

First Phase of the Algorithm: When a process, say P_i, running on an MH, say MH_i, initiates a check pointing, it sends a checkpoint initiation request to its local MSS. The initiator MSS maintains the dependency vector of P_i. On the basis of dependency vector, the set of dependent processes of P_i form the minimum set. The initiator MSS broadcasts volatile checkpoint request to all MSSs. When an MSS receive checkpoint request along with minimum set, it checks, if any processes in minimum set are in its cell. If so, the MSS broadcast the volatile checkpoint request message to all MH. The processes which are included in the minimum set will take volatile checkpoint and sends a response to its local MSS. After receiving response messages from the processes the local MSS sends a response to the initiator MSS. In the first phase, all process takes the volatile checkpoints instead of tentative checkpoint.

Second Phase of the Algorithm: After receiving response at the initiator MSS from every MSS algorithm enters the second phase. If the initiator MSS comes to know that all relevant processes have taken their volatile checkpoints successfully, it asks them to convert their volatile checkpoints into tentative ones. Alternatively, if initiator MSS comes to know that some process has failed to take its volatile checkpoint in the first phase, it issues abort request to all MSS. In this approach the MHs need to abort only the volatile

checkpoints not the tentative ones. In this algorithm we try to reduce the cost of check pointing effort in case of abort of check pointing algorithm in first phase.

Third Phase of the Algorithm: Finally, when the initiator MSS comes to know that all processes in the minimum set have taken their tentative checkpoints successfully, it issues commit request to all MSSs. When a process in the minimum set gets the commit request, it converts its tentative checkpoint into permanent one and discards its earlier permanent checkpoint, if any.

IV. OVERVIEW OF CHECK POINTING

A. Protocol

Checkpoint is defined as a designated place in a program at which normal processing is interrupted specifically to preserve the status information necessary to allow resumption of processing at a later time. Check pointing is the process of saving the status information. A checkpoint of a process is the information about the state of a process at some instant of time. Fault tolerance through checkpoint and recovery techniques includes taking check- point of an application process periodically, and logging the checkpoint in a stable storage which is immune to failures. Checkpoint of an application process is the information about the state of the process and can be used to restart it from a state corresponding to the checkpoint. On the other hand, roll back recovery for an application is defined as the procedure for restarting the application process from a checkpoint stored in stable storage. A checkpoint can be saved on either stable storage or the volatile storage of another process, depending on the failure scenarios to be tolerated. For long-running scientific applications, check pointing and rollback-recovery can be used to minimize the total execution times in the presence of failures.

VI. HANDLING FAILURES INCOORDINATEDCHECKPOINTING

A distributed system is a collection of independent entities that cooperate to solve a problem that cannot be individually solved. A mobile computing system is a distributed system where some of processes are running on mobile hosts (MHs), whose location in the network changes with time. In minimum-process check pointing protocols, some useless checkpoints are taken or blocking of processes takes place. In this chapter, we propose a minimum-process coordinated check pointing algorithm for non-deterministic mobile distributed systems, where no useless checkpoints are taken. An effort has been made to minimize the blocking of processes and synchronization message overhead. We capture the partial transitive dependencies during the normal execution by piggybacking dependency vectors onto computation messages. Frequent aborts of check pointing procedure may happen in mobile systems due to exhausted battery, non-voluntary disconnections of MHs, or poor wireless connectivity.

VII. BACKGROUND

Most of the existing coordinated check pointing algorithms rely on the two-phase protocol and save two kinds of checkpoints on the stable storage: tentative and permanent. In the first phase, the initiator process takes a tentative checkpoint and requests all or selective processes to take their tentative checkpoints. If all processes are asked to take their checkpoints, it is called all-process coordinated check pointing. Alternatively, if selective communicating processes are required to take checkpoints, it is called minimum-process check pointing each process informs the initiator whether it succeeded in taking a tentative checkpoint. After the initiator has received positive acknowledgments from all relevant processes, the algorithm enters the second phase. Alternatively, if a process fails to take its tentative checkpoint in the first phase, the initiator process requests all or concerned processes to abort their tentative checkpoint. If the initiator learns that all concerned processes have successfully taken their tentative checkpoints, the algorithm enters in the second phase and the initiator asks the relevant processes to make their tentative checkpoints permanent. In order to record a consistent global checkpoint, when a process takes a checkpoint, it asks (by sending checkpoint requests to) all relevant processes to take checkpoints. Therefore, coordinated check pointing suffers from high overhead associated with the check pointing process. Much of the previous work in coordinated check pointing has focused on minimizing the number of synchronization messages and the number of checkpoints during the check pointing process. However, some algorithms (called blocking algorithm) force all relevant processes in the system to block their computations during the check pointing process. Check pointing includes the time to trace the dependency tree and to save the states of processes on the stable storage, which may be long. Moreover, in mobile computing systems, due to the mobility of MHs, a message may be routed several times before reaching its destination. Therefore, blocking algorithms may dramatically reduce the performance of these systems recently, non-blocking algorithms have received considerable attention. In these algorithms, processes need not block during the check pointing by using a check pointing sequence number to identify orphan messages. Moreover, these algorithms require all processes in the system to take checkpoints during check pointing, even though many of them may not be necessary.

VIII. MOBILE DISTRIBUTED SYSTEMS

A Mobile distributed system is a collection of independent entities that cooperate and co-ordinate to solve a problem with the help of mobile protocols. A message passing system consists of N fixed number of nodes that communicate each other only through messages. Some of the nodes may change their location with time. They are referred to as mobile hosts or MHs Static Nodes are referred to as mobile support stations or MSSs are connected to each other by a Static Network. An MH can be directly connected to at most one MSS at any given time and can communicate with other MHs and MSSs only through the MSS to which it is directly connected. The system does not have any shared memory or a

global clock. Hence all the communication and synchronization takes place through messages. The messages generated by underlying distributed application will be referred to as computation messages. Messages generated by the nodes to advance checkpoints, handle failures and for recovery will be referred to as system messages. Processes have access to a stable storage device that survives failures. Mobile Hosts (MHs) are increasingly becoming common in distributed systems due to their availability, cost, and mobile connectivity. An MH is a computer that may retain its connectivity with the rest of the distributed system through a wireless network while on move. An MH communicates with the other nodes of the distributed system via a special node called mobile support station (MSS). A "cell" is a geographical area around an MSS in which it can support an MH.

IX. CONCLUSION

We have made a study of the existing check pointing protocols for distributed and mobile systems. We have proposed a minimum-process check pointing protocol for non-deterministic mobile distributed systems, where no useless checkpoints are taken and an effort has been made to minimize the blocking of processes. We try to reduce the check pointing time and blocking time of processes by limiting check pointing tree which may be formed in other algorithms. We captured the transitive dependencies during the normal execution by piggybacking dependency vectors onto computation messages. The Z-dependencies are well taken care of in this protocol. We also try to reduce the loss of check pointing effort when any process fails to take its checkpoint in coordination with others. Frequent aborts of check pointing procedure may happen in mobile systems due to exhausted battery, non-voluntary disconnections of MHs, or poor wireless connectivity. Therefore, we proposed that in the first phase, all concerned MHs will take ad hoc checkpoint only. Ad hoc checkpoint is stored on the memory of MH only. In this case, if some process fails to take checkpoint in the first phase, then MHs need to abort their mutable checkpoints only. In this way, we try to minimize the loss of check pointing effort when any process fails to take its checkpoint in coordination with others.

Future Scope: We have avoided the concurrent executions of the proposed protocols. The proposed algorithms need to be modified for allowing concurrent executions of both the algorithms. There are hardly any check pointing schemes for ad hoc networks. The check pointing schemes designed for distributed systems or mobile distributed systems may not be applicable for ad hoc networks due to certain issues with them. Therefore, there is a scope for modifying existing check pointing schemes designed for distributed or mobile systems for their suitability for ad hoc networks. One should further try to minimize.

X. REFERENCES

[1]Vijaya kapoor & Parveen Kumar "A Comparative Study on Snapshot Protocols for Mobile Distributed Systems "International Journal of Computer Applications (0975 – 8887) Volume 106 – No.3, November 2014.

[2]Cao G. and Singhal M., "On coordinated checkpointing in Distributed Systems", IEEE Transactions on Parallel and Distributed Systems, vol. 9, no.12, pp. 1213-1225, Dec 1998.

[3]Cao G. and Singhal M., "On the Impossibility of Min-process Non-blocking Checkpointing and an Efficient Checkpointing Algorithm for Mobile Computing Systems," Proceedings of International Conference on Parallel Processing, pp. 37-44, August 1998.

[4] L. Kumar, M. Misra, R.C. Joshi, "Low overhead optimal checkpointing for mobile distributed systems" Proceedings. 19th IEEE International Conference on Data Engineering, pp 686 – 88, 2003.

[5] Acharya A. and Badrinath B. R., "Checkpointing Distributed Applications on Mobile Computers," Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems, pp. 73-80, September 1994.

[6] Elnozahy E.N., Alvisi L., Wang Y.M. and Johnson D.B., "A Survey of Rollback-Recovery Protocols in Message-Passing Systems," ACM Computing Surveys, vol. 34, no. 3, pp. 375-408, 2002.

[7] Cao G. and Singhal M., "Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing systems," IEEE Transaction On Parallel and Distributed Systems, vol. 12, no. 2, pp. 157-172, February 2001

[8] Kumar Parveen, Gupta Sunil Kumar, Chauhan R.K., "Backward Error Recovery Protocols in Distributed Mobile Systems: A Survey", Journal of Theoretical and Applied Information Technology, pp. 337-347, 2008

[9] Vijaya kapoor & Parveen kumar " Review of State Based Approach Recovery Schemes in Mobile Distributed Environment" 2015 Fifth International Conference on Advanced Computing & Communication Technologies, 2327-0659/15 IEEE , DOI 10.1109/ACCT.2015.19

[10]Vijaya kapoor & Parveen kumar "Recent Trends on Consistent Global Snapshot Algorithms for Distributed Mobile Environments", International Journal of Computer Applications (0975 – 8887) Innovations in Computing and Information Technology (Cognition 2015)

[11]S. kalaiselvi & v Rajaraman " A survey of checkpointing algorithms for parallel & distributed computers " 2000 sadhana vol 25.

[12]Acharya A., "Structuring distributed algorithms and services for networks with mobile hosts", Ph.D. Thesis, Rutgers University, 1995.

[13]Kumar, P.," A Low-cost hybrid coordinated checkpointing protocol for mobile distributed systems", Mobile Information Systems pp 13-32, Vol. 4, No. 1. ,2007

Author's Profile:

Miss. Hurmat Shahin is working towards a B.Tech & M.Tech in E C E at prestigious NIMS University, Jaipur, India. His Research interests are in the areas of Embedded systems, which are the emerging field in current scenario.

Mrs.Nidhi Sharma is presently working as an Assistant Professor of Computer & Science Engineering with NIMS University Jaipur.