Analysis of Various Non-blocking Coordinated Checkpointing Algorithms for Handling Failures during Checkpointing

Amit Chaturvedi, Ph.D
Head, MCA Deptt,
Govt. Engg. College,
Ajmer Rajasthan, India,

Syed Sajad Hussain
M.Phil Computer Science
Bhagwant University Ajmer,
Indergam, J & K, India,

ABSTRACT
A checkpointing algorithm needs to handle many issues for mobile computing such as mobility, low bandwidth of wireless channels, lack of stable storage on mobile hosts, disconnections, limited battery power and high failure rate of mobile hosts. These issues make traditional checkpointing techniques unsuitable for such environments. Minimum-process coordinated checkpointing is a very enchanting approach to introduce fault tolerance in mobile distributed systems transparently. This approach is domino-free, requires at most two checkpoints of a process on stable storage, and forces only a minimum number of processes to take checkpoint. In this paper, various Non-blocking Coordinated Checkpointing Algorithms are analyzed for handling failures during checkpointing, which requires only a minimum number of processes to take permanent checkpoints. In case of a failure, after recovery a consistent global state is found from the existing checkpoints and the system restarts from there.

Keywords: Checkpoint, Coordinated Checkpoint, Mobile Computing

Abbreviations': MH- Mobile Host, MSS- Mobile Support Station, CGS-Consistent Global State

1. INTRODUCTION
A mobile computing system nowadays considered as an extension of distributed systems, and many algorithms and protocols for supporting distributed services are now extended to continue the service in the mobile environment. However, the straight forward extension of the existing algorithms and protocols are not well adapted to the new environment, since the mobile system introduces a new challenge that is the handling of mobile hosts. When the mobile host (MH) moves from a cell to another cell, the mobility of mobile host must be properly handled. The bandwidth between the mobile host and the MSS is very flimsy. The low battery capacity of mobile host can also be a problem, and sometimes they perform the disconnected operations to save the power consumption. Another property of mobile host (MH) is that it usually carries the small memory and disk spaces [1]. A checkpointing algorithm for mobile computing systems needs to handle many new issues like: mobility, low bandwidth of wireless channels, and lack of stable storage on mobile host, disconnections, limited battery power and high failure rate of mobile host. An optimization technique, which significantly reduces the number of useless checkpoints at the cost of minor increase in the message complexity. In coordinated checkpointing, if a single process fails to take its temporary checkpoint; all the checkpoint effort is aborted. We try to reduce this effort by taking soft checkpoints in the first phase at Mobile Hosts [2].

2. SYSTEM MODEL
In mobile computing system processes do not share memory and they communicate via messages sent through the channels. All channels are fault-prone, that is they can lose messages. However, the order of messages is preserved by some end-to-end transmission protocol.

3. CHECKPOINTING
Checkpointing can be defined as a designated place in a program at which normal process is interrupted specifically to preserve the status information necessary to allow resumption of processing at a later time. A checkpoint is a snapshot of the local state of a process, saved on local non-volatile storage to survive process failures [8]. Checkpointing, a process periodically provides the information necessary to move it from one processor to another processor. Checkpoints store the specific states of processes in stable storage**.**
The main motive of using Checkpointing is:

- To recover from failures**.**
- Checkpointing is also used in debugging distributed programs and migrating processes in multiprocessor system.
- To balance the load the heavily loaded processors are moved from lightly loaded ones in distributed systems.
- With checkpointing, an arbitrary temporal section of a program's runtime can be extracted for exhaustive analysis without the need to restart the program from beginning [19].

4. COORDINATED CHECKPOINTING
Coordinated checkpointing or Synchronous checkpointing is a commonly used technique to prevent complete loss/failure of computation upon a failure [3]. In the system every process is periodically saved on the stable storage, which is called a checkpoint of the process. To recover from a failure, the system restarts its execution from a previous consistent global checkpoint saved on the stable storage and is not susceptible to the domino effect [2][3][4]. In order to record a consistent global checkpoint, processes must coordinate their checkpointing activities [2][3].We can also say that , when a process takes a checkpoint, it asks all same processes to take checkpoints. Therefore, coordinate checkpointing suffers from high overhead associated with the checkpointing process [5].Mostly it follows two-phase commit structure. In the first phase, processes take temporary checkpoints and in the second phase, these are made permanent/fixed. The main advantage is that only one permanent checkpoint and at most one temporary checkpoint is required to be saved. In case of a fault, processes rollback to last checkpointed state. A permanent checkpoint cannot be undone / renewed. It

guarantees that the computation needed to reach the checkpointed state will not be repeated. A temporary checkpoint, however, can be undone or changed to be a permanent checkpoint [2][3].

The Synchronous checkpointing can be divided into two types: blocking and non-blocking. In blocking algorithms, blocking of some processes takes place during checkpointing. In non-blocking algorithms, no blocking of processes is required for checkpointing. The coordinated checkpointing algorithms can also be classified into following two categories: minimum-process and all process algorithms. In all-process coordinated checkpointing algorithms, every process is required to take its checkpoint in beginning. In minimum-process algorithms, minimum interacting processes are required to take their checkpoints in an beginning [2][3]. In the first phase, all concerned MHs will take soft checkpoint only. Soft checkpoint is similar to mutable checkpoint, which 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 soft checkpoints only. The effort of taking a soft checkpoint is negligibly small as compared to the tentative one [23].

5. PREVIOUS COORDINATED CHECKPOINTING ALGO-RITHM or RELATED WORK

Research in the area of coordinated checkpointing concentrates mostly on non-blocking approaches. In [5] and [9] the [C.Guohong, S. Mukesh and P. kumar, L. kumar etal] have proposed non-blocking coordinated checkpointing algorithms that require only aminimum number of processes to take checkpointsat any instant of time. In coordinated or synchronous checkpointing, processes coordinate their local checkpointing actions such that the set of all recent checkpoints in the system is guaranteed to be consistent [6]. Always every process restarts from its most recent checkpoint. Also, coordinated checkpointing requires each process to maintain only one permanent checkpoint on stable storage, reducing storage overhead and reducing the need for garbage collection [7].

At any instant of time one process act as a checkpoint coordinator called the initiator or root process. Each process maintain one permanent checkpoint, belongs to the most recent consistent checkpoint. During each run of the protocol, each process takes a tentative checkpoint, which replaces the permanent one only if the protocol terminates successfully [6]. In this algorithm if any process is busy with other high priority job, it takes the checkpoint after the job ends. Otherwise it takes a checkpoint immediately. Each process stores one permanent checkpoint. In addition each process can have one temporary checkpoint, and are either discarded or made permanent after some time. Each process maintains a checkpoint integer number (cin), and it is incremented by one in every checkpoint session. Here we use the word checkpoint for tentative checkpoint [21].

In the first phase, processes take temporary checkpoints, and in the second phase, these are made permanent. The main advantage is that only one permanent checkpoint and at most one temporary checkpoint is required to be stored. [8].In coordinated checkpointing approach; all processes synchronize through control messages before taking checkpoints. These synchronization messages contribute to extra overhead but make the system free from domino effect. Coordinated check pointing algorithms are of two types: (a) blocking and (b) non-blocking and .Blocking algorithms force all relevant processes in the system to block their computation during check pointing latency and hence degrade system performance from the viewpoint of larger execution time of application programs. In non- blocking algorithms application processes are not blocked when checkpoints are being taken [9].The first coordinated checkpointing protocol assumes that all communications are atomic, which is too restrictive. Koo-Toeg proposed a minimum-process coordinated checkpointing protocol which relaxes the assumption that all communications are atomic. The number of coordinated messages and number of checkpoints are reduced [3][4]. However, these algorithms (called blocking algorithms) force all same processes in the system to block their computations during the checkpointing process. Checkpointing 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 final position. Therefore, they may further degrade the performance of mobile computing systems [16],

In coordinated checkpointing protocols, we may require piggy backing of integer csn (checkpoint sequence number) along with normal messages. L. Kumar et al. proposed an all process are non-intrusive checkpointing protocol for distributed systems, where just one bit is piggy backed along with normal messages. This is done by incurring extra over head of vector transfers during checkpointing.[3] Prakash-Singhal algorithm was the first algorithm to combine the two i.e., min-processes and non-blocking [15], it forces only a minimum number of processes to take checkpoints and does not block the underlying computation during checkpointing [9]. However, it was proved that their algorithm may result in an inconsistency. Cao and Singhal [17] achieved non-intrusiveness in the minimum-process algorithm by introducing the concept of mutable checkpoints. The number of useless checkpoints in [11] may be exceedingly high in some situations [13]. Kumar et. al [12],[13] reduced the height of the checkpointing tree and the number of useless checkpoints by keeping non- intrusiveness intact, at the extra cost of maintaining and collecting dependency vectors, computing the minimum set and broadcasting the same on the static network along with the checkpoint request. Some minimum-process blocking algorithms are also proposed in literature [5], [14], [15],[20]. In coordinated checkpointing algorithm presented in [5], each process needs to store only one permanent checkpoint on the stable storage and at most two checkpoints: a permanent and a tentative (or mutable) checkpoint only for the duration of the checkpointing. Generally speaking, uncoordinated checkpointing approaches suffer from the complexities of finding a consistent recovery line after the failure, the susceptibility to the domino effect, the high stable storage overhead of saving multiple checkpoints of each process, and the overhead of garbage collection. Thus, our coordinated checkpointing algorithm has many advantages over uncoordinated checkpointing algorithms [5]. All the coordinated checkpointing algorithms have focused on minimizing the number of synchronization messages and the number of checkpoints during checkpointing. However, these algorithms force all relevant processes in the system to block their computations during the checkpointing process [4].

A non-blocking coordinated checkpointing algorithm where, in the first phase an initiator sends checkpointing request to all other processes in a system. In the second phase, dependent processes take temporary checkpoints; however all processes send their responses to the initiator regarding if they have

taken temporary checkpoints or not. In the third phase, the initiator sends the commit message to all other processes if it gets replies from all the processes within a specified time interval and takes its own checkpoint; otherwise, it sends an abort message. This work does not guarantee that only minimum number of processes will take checkpoints [18].

6. PROBLEM STUDY

In the non-blocking checkpointing algorithms discussed [18] above, first all the processes are expected to take temporary checkpoints when they receive checkpoint request and then these checkpoints are converted to permanent checkpoints. However, if a process is busy with some high priority procedure when a checkpointing request arrives at it, then it will not take a checkpoint. In such a situation, every process that has already taken a temporary checkpoint must discard it, and continue normal execution. Later the algorithm has to be restarted again. It thus wastes time and delays further the execution of the application program. The same problem may also arise when a process receives a request to convert its temporary checkpoint to a permanent one while it is busy executing a high priority procedure. In this case also the checkpointing algorithm has to be abruptly terminated and it will restart later. Thus, such situations definitely affect the execution time of the application programs adversely [18].

7. Conclusion

In this paper, we have presented that a coordinated checkpointing algorithm:

- does not take any temporary or mutable checkpoint unlike in some other important related works. Hence, the need for converting such checkpoints to permanent ones is eliminated. It makes the algorithm faster.
- absence of temporary or mutable checkpoints means that a participating process faces less number of interrupts. This also contributes to the execution speed.
- to reduce the loss of checkpointing effort when any process fails to take its checkpoint in coordination with others.

Hence non-blocking coordinated checkpointing algorithm handles the failures during checkpointing by avoiding the use of temporary or mutable checkpoints.

8. ACKNOWLEDGMENTS

We feel grateful to the anonymous referees for their comments and for their valuable suggestions that have helped immensively in preparing the revised manuscript.

9. REFERENCES

[1] An Asynchronous Recovery Scheme based on Optimistic Message Logging for the Mobile Computing Systems Taesoon Park , Heon Y. Yeom.

[2] Rachit G., Praveen K., "A Nonblocking Coordinated Checkpointing Algorithm for Mobile Computing Systems",(IJCSI) International Journal of Computer Science Issues, Vol. 7, Issue 3, No 3, May 2010.

[3] Rachit G.,Praveen K.,"A Review of Fault Tolerant Checkpointing Protocols for Mobile Computing Systems", *International Journal of Computer Applications (0975 – 8887),Vol. 3, No.2, June 2010.*

[4] Guohong C., Mukesh S.,"*Checkpointing with mutable checkpoints", Theoretical Computer Science,* 290 (2003) 1127–1148.

[5] C. Guohong and S. Mukesh, "Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing Systems", IEEE Trancactions on Parallel and Distributed Systems , Vol. 12, No. 2, February 2001.

[6] Parveen K., Poonam G.," A Low-Overhead Minimum Process Coordinated Checkpointing Algorithm for Mobile Distributed System " *Global Journal of Computer Science and Technology (GJCST),* Vol.10, Issue 6, Ver. 1.0, July 2010.

[7] Ajay K., Praveen K., "An Analysis of Check-pointing Algorithms for Distributed Mobile Systems*",(IJCSE) International Journal on Computer Science and Engineering*, Vol. 02, No. 04, 2010, 1314-1326.

[8] Parveen K., Rachit G.," Soft-Check-pointing Based Coordinated Checkpointing Protocol for Mobile Distributed Systems*",(IJCSI) International Journal of Computer Science Issues*, Vol. 7, Issue 3, No 5, May 2010.

[9] Parveen Kumar, Lalit Kumar, R K Chauhan, V K Gupta.," A Non-Intrusive Minimum Process Synchronous Checkpointing Protocol for Mobile Distributed Systems‖ Proceedings*", IEEE ICPWC*-2005, pp 491-95, January 2005.

[10] Cao G. and Singhal M., "On the Impossibility of Min-process Non-blocking Checkpointing and an Efficient Checkpointing Algorithm for Mobile".

[11] Elnozahy E.N., Johnson D.B. and Zwaenepoel W., "The Performance of Consistent Checkpointing, *Proceedings of the 11th Symposium on Reliable Distributed Systems*", pp. 39-47, October 1992.

[12] Higaki H. and Takizawa M., "Checkpoint-recovery Protocol for Reliable Mobile Systems, *Trans. of Information processing Japan*", Vol. 40, No.1, pp. 236-244, Jan. 1999.

[13] Ssu K.F., Yao B., Fuchs W.K. and Neves N. F., "Adaptive Checkpointing with Storage Management for Mobile Environments", *IEEE Transactions on Reliability*, Vol. 48, No. 4, pp. 315- 324, December 1999.

[14] Silva, L.M. and J.G. Silva, "Global checkpointing for distributed programs", *Proc. 11th symp. Reliable Distributed Systems*, pp. 155-62, Oct. 1992.

[15] Parveen Kumar, "A Low-Cost Hybrid Coordinated Checkpointing Protocol for mobile distributed systems", Mobile Information *Systems*, pp 13-32, Vol. 4, No. 1, 2007.

[16] R. Prakash, M. Singhal, "Low-cost checkpointing and failure recovery in mobile computing systems", *IEEE Trans. Parallel Distributed System* 7 (10) (1996) 1035–1048.

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

[18] S. Neogy, A. Sinha, P.K. Das, "CCUML: a check pointing protocol for distributed system processes,"

Tencon 2004,IEEE Region 10 Conference Vol. B, No.2, pp. 553 – 556, November 2004, Thailand.

[19] G. Poonam, K. Parveen ," Review of Some Check-pointing Algorithms in Distributed and mobile systems", *International Journal of Engineering Science and Technology*, Vol. 2(6), 2010, 1594-1602.

[20] Parveen,Praveen K., "Analysis of Some Minimum-process Coordinated Checkpointing Algorithms for Mobile Computing Systems" *TECHNIA International Journal of Computing Science and Communication Technologies*, VOL. 4, NO. 1, July 2011. (ISSN 0974-3375).

[21]Surender K., Parveen K. & R.K.Chauhan,"Hierarchical Non-blocking Coordinated Checkpointing Algorithms for Mobile Distributed Computing", *International Journal of Computer Science and Security (IJCSS)* , Vol. 3, Issue 6,518

Dr. Amit Chaturvedi, Head, MCA Deptt., Govt. Engineering College, Ajmer have completed Ph.D. (Computer Sc.) in Mar, 2012. He has more than 12 years of teaching experience. He have published around 26 research papers in national and international Journals in the area of Mobile computing system, Spectrum requirement estimation for 4G, and Cryptography and always ready to teach the subjects to his students, which he does with great finess.

Syed Sajad Hussain received the Bachlors degree in Computer Application (BCA) from HNBG University Srinagar, Uttrakhand, India. He received the MCA degree from Uttrakhand Technical University; Dehradun, India in July 2009.He also received his B.Ed degree from University of Kashmir in December 2011. He is presently the faculty member at Baramulla Public School, Kashmir India and a research scholar. His research interests include mobile computing, cryptography, software engineering.