IJITKM 0973-4414 Volume-9, Number-1 Jun-Dec 2015 pp. 13-18 **Impact Factor 2.8** available online at www.csjournalss.com

Time based Synchronous Global State Determination Algorithm for MDSs

Raman Kumar[1], Parveen Kumar[2]
[1]Research Scholar, Deptt. of Computer Sci. & Engg., Mewar Univ., Chittorgarh, Rajesthan, INDIA
[2]Department of Computer Science & Engineering, NIT, Hamirpur (H.P), INDIA
rmn.kmr1@gmail.com, pk223475@gmail.com

Abstract: Two checkpointing approaches i.e., coordinated checkpointing and times based checkpointing are widely used in the literature of MDSs. Coordinated checkpointing protocols uses the less checkpoints and domino free but have large coordinated message overheads as processes synchronize by exchanging coordinated message. However, time based protocol requires every process to take checkpoint during checkpointing. Some of the time based protocols defines the timeout period t_p. When this timeouts occur, processes take their checkpoint. If the checkpointing interval is too small then multiple checkpoints are transferred from MHs to MSS through wireless link and also checkpointing takes some time to save the application state. This approach increases the checkpointing overheads. On the other hand, if checkpointing interval is too large this may leads to large amount of computational loss during rollback and recovery. As a result time based approach has minimum coordinated messages overheads cost but has high checkpointing cost as it requires large number checkpoints than minimum but coordinated checkpointing approach have minimum checkpointing cost than time based approach but higher coordinated message overheads cost. In this paper, we design an efficient coordinated checkpointing protocol which uses time to indirectly coordinate to minimize the number of coordinated message transmitted through the wireless link and reduces the number of checkpoints nearest to the minimum. The algorithm is non-blocking and minimum process.

Keywords: Checkpointing, Global State, Mobile Distributed Systems, Consistent Global State, Coordinated Checkpointing, Time based Checkpoiting

Introduction

Checkpointing/rollback recovery is an attractive and popular technique which gives fault tolerance without additional efforts in DSs [3]. A checkpoint is a global state of a process stored on stable storage. In a DS, since the processes in the system do not share memory and have not any synchronized clock, a global state of the system is defined as a set of LSs, one from each process. A global state is said to be "consistent" if it contains no orphan message; i.e., a message whose receive event is recorded, but its send event is lost. To recover from failure, the system restarts its execution from a previous CGS saved on the stable storage during fault-free execution. Checkpointing algorithms for DSs have been extensively studied in the literature (e.g., [2], [8], [9], [10], [11], [12], [13], [19]). Due to the emerging challenges of the MDS as low bandwidth, mobility, lack of stable storage, frequent disconnections and limited battery life, the fault tolerance technique designed for distributed system cannot directly implemented on mobile distributed systems(MDSs) [1], [5], [14].[Figure 1]

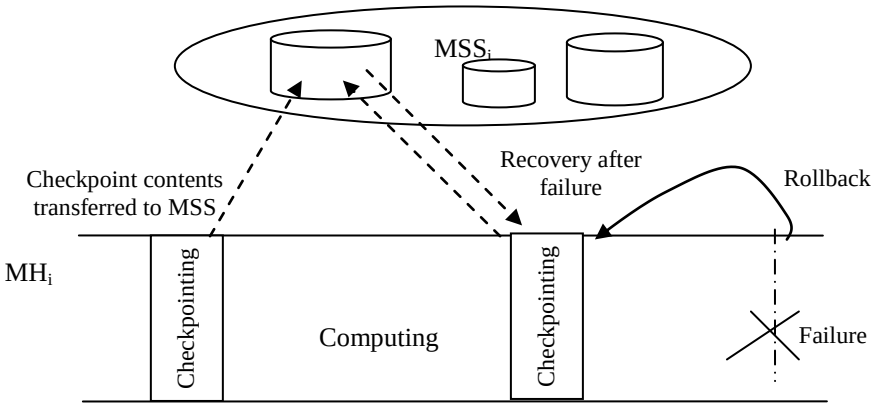

Figure 1 Checkpointing and rollback recovery in MDS

A common goal of checkpointing algorithm for MDSs is to reduce the checkpointing cost by taking minimum number of checkpoints and reducing the coordinated messages. Hence, the checkpointing algorithms having lesser number of coordinated messages and fewer checkpoints nearly to minimum are preferred for mobile environment.

Related Works and Problem Formulation

Recent research in checkpointing has considered for MDSs [1], [5], [15], [20], [21]. In [1] authors introduced a two-phase checkpointing protocol to determining global consistent checkpoints for mobile systems. In their protocol, they proposed that checkpoint be stored on stable storages at the MSS. In this approach, an MH takes a local checkpoint whenever a message receipt is preceded by the message sent at that MH. This algorithm has no control over checkpointing activity on MHs and depends totally on communication patterns. In [14] authors evaluated the performance of different state saving protocols and handoff strategies. They also suggested storing checkpoints on MSS. An adaptive checkpointing scheme which uses time to indirectly coordinate the creation of CGS for mobile systems was proposed in [11]. A hybrid checkpointing protocol which leaves an agent process on each MSS was developed in [19]. In this approach processes in SHs recover from consistent checkpoints during recovery.

Cao and Singhal [5] achieved non-intrusiveness in minimum-process algorithm by introducing the concept of mutable checkpoints to adapt to mobile environments. This algorithm is modified version of the algorithm proposed in [4]. Mutable checkpoints need not be saved on the stable storage and can be store on the main memory of the MHs and has not any transferring cost. In this algorithm, checkpoint initiator process (says P_i) sends the checkpoint request to P_j only if P_i receives m from P_j in the current checkpointing interval (CI). If P_j does not inherit the request, it simply ignores it. Otherwise, P_j takes its tentative checkpoint and propagates the request to P_k only if P_j receives m from P_k in the current CI. In this case, if P_j knows that some other process has already sent the checkpoint request to P_k and P_k is not going to inherit the current checkpoint request, then Pj does not send the checkpoint request to P_k. This process is continued till the checkpoint request reaches all the processes on which the initiator process transitively depends. Suppose, during checkpointing process, P_1 receives m from P_2. P_1 takes its mutable checkpoint before processing m only if the following conditions are met: (i) P_2 has taken some checkpoint in the current initiation before sending m (ii)P_1 has not taken any checkpoint in the current initiation. (iii)P_1 has sent at least one message since its last permanent checkpoint. If P_1 takes mutable checkpoint and is not a member of the minimum set, it discards its checkpoint on commit.

In papers [6], [16], [11], and [19] authors proposed an efficient time base checkpointing algorithm. The algorithm presented in [6] has lower coordinated message overheads but a global checkpoint consists of all the *Nth* checkpoints of every process which awoke the processes in doze mode operation. In [16], each process takes its checkpoint at predetermined time instants according to its own local clocks to make the checkpoint consistent. This problem is addressed by using extra messages for clock synchronization. In [11], authors proposed adaptive checkpointing algorithm where they used time to indirectly coordinate the creation of recoverable consistent checkpoints. It requires that checkpoints be sent back only to home agents, which results in high failure-free overhead during checkpointing [17].

Proposed Algorithm

A. System Model

The MDS can be considered as consisting of "n" Mobile Hosts (MHs) and "m" Mobile Support Stations (MHSs). All the MSSs are connected through static wired network. A cell is a small geographical area around the MSS supports a MH only within this are and there is a wireless link between a MH to MSS. A MH can communicate to another MH only through their reachable MSS. There are n spatially separated seqsuential processes denoted by P_0, P_1,.., P_{n-1}, running on MHs or MSSs, constituting a mobile distributed computing system. Each MH/MSS has one process running on it. The processes do not share memory or clock and message passing is the only way for processes to communicate with each other.

As there is no common clock and processes do not share a common memory but every MH and MSS contains a system clock, with typical clock drift rate p in the order of 10^{-5} to 10^{-6}. The system clocks of MSSs can be synchronized using the network time protocol (NTP). MHs start their execution with their own initial timer. Clock can be re-synchronizing with following two methods, to solve the initial time inaccuracies.

B. Data structure

a) Each process P_i maintains the following data structure:

Proposed algorithm consider a distributed system which has a set of n processes, $\{P_0, P_1,\ldots,P_{n-1}\}$ where each process P_i maintains the following data structure.

***c_i*:** a Boolean flag c_i that is initially set at zero. It is set 1only when process P_i sent a message during current CI, after its latest checkpoints.

***csn*:** checkpoint sequence number of the process which is incremented after taking the checkpoint.

***CLK_i*:** clock of process P_i which shows the time interval until next checkpoint.
***CS_i*:** a Boolean flag Checkpoint State CS_i which is initially set to 0 which shows that process does not takes during its current CI. If process takes checkpoint during its current CI then it will set $CS_i = 1$.
***m_i*:** computation message sent by the process P_i.
***SC_i*:** Each process P_i takes soft checkpoint SCi when its local timer expires and does not sends any during its current CI.
***$Reply_i$*:** Each process P_i sends reply message to its local MSS of taking the permanent and soft checkpoint after the expiries of its timer.

b) Each MSS maintains the following data structure:
CLK_M: clock of the MSS which show the time interval until next checkpoint.
***Minset[]*:** Each MSS maintains the set to minimum number of process which communicate through the MSS.

C. The Algorithm

When local clock of process P_i expires

```
If(CLK_i has expired)
        If((c_i=1)AND(CS=0))
        {take checkpoint;
         increment in csn;
         set c_i=0; //
         continue normal operation ;}
 else if CS=1; //
        set CS=0;
        if SC= T; //
        set SC=F; //
else
        continue normal operation;
 else
        set SC= T; // take soft checkpoint
```

When P_i receives piggybacked message from P_j when its local clock has not expired

When a process P_i sends computation message by attaching its csn to another process P_j through its local MSS, MSS piggybacked it with time interval to next checkpoint, to the destination process.

```
Receives message (P_j, csn_j, CLK_M, m_j)
        If((csn_i=csn_j) AND (CLK_i ≠CLK_M))
        reset CLK_i = CLK_M;
        receives message
else if ((csn_i < csn_j) AND (c_i=1))
        take checkpoint;
        increments csn_i;
        set CS_i=1; set c_i = 0;
        reset timer CLK_i = CLK_M;
        process message m_j;
        set SC=F;
else
        resumes normal operation
```

D. *Working of the Algorithm*

A process P_i takes its checkpoint on two ways:
a) On the expires of its local timer CLK_i : if its timer has expires, then it checks the status of c_i. if $c_i=1$, then it takes a permanent checkpoint, in another case if $c_i=0$, it takes soft checkpoint(SC).

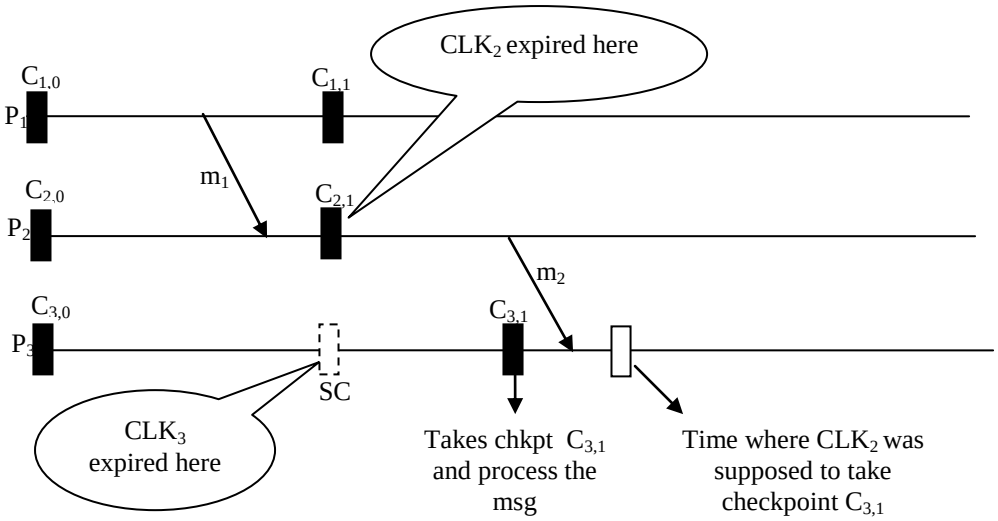

Figure 2 Working of proposed algorithm

b) On the receipt of piggybacked message in between current CI: in such case it will checks and compares the received csn with its own csn_i. There may be following two possibilities:

i) msg_csn < = own_csn_i : if it is true, process P_i receives message as a normal without taking any checkpoint. In Figure 2, csn of process P_1 received with message m_1 is equal to the csn_2. Hence, checkpoint is not taken after receiving the message. However process P_2 takes a checkpoint, after expires of its current CI it sends any message.

ii) msg_csn > own_csn_i: it is true, process P_i takes checkpoint first and then precedes the message. If process P_i taken any soft checkpoint, it will discards it and set the checkpoint state (CS) is equal to 1. Furthermore P_i does not takes checkpoint, in the current CI, after expires of its local clock CLK_i, if it does not send any message after taking the checkpoint and before expires of the clocks. In Figure 2, process P_3 takes a checkpoint before receiving the message m_2 as csn received with m_3 is greater than its own csn. Process P_3 receives the message after taking the checkpoint $C_{3,1}$ and it does not takes checkpoint after expires of its local clock. The

Performance Analysis

In this section we analyze our checkpointing algorithm by comparing with different existing algorithms in different context. We use following notations to compare our algorithm with some of the most notable algorithms in this area of research, namely [2],[5],[6] and [11]. A performance comparison is given in Table 6.2. In this Table:

C_{air}	Cost of sending a message m from any P_i to P_j;
C_{wired}	Cost of sending a message in the wired link from MSS to MSS.
C_{wl}	Cost of sending a message in the wireless link from MH to MSS.
C_{broad}	Cost of broadcasting a message to all processes;
N_{min}	Number of processes that belong to minset;
N	Total number of MHs in the system;
N_{dep}	Average no. of processes on which a process depends;
T_{ckpt}	Total time taken to store the checkpoint on stable storage

Table 6.2 compares the performance of our algorithm with the algorithms in [2], [5], [6], [11]. Compared to Naves-Fuchs[11],which is also time-based , our algorithm minimize the half coordinated message including with involving only minimum number of process, so that the total number of checkpoints transmitted onto the wired and wireless network is reduced. Fewer checkpoints and coordinated message transmitted low power consumption for MHs. Compared to [6], our approaches involves only minimum number of processes in global state. For the size of the piggybacked information and the coordination message in the wireless link, our algorithm reduces the overheads compared to Cao-Singhal [5] with O(1) to O(N).

Table 1 Analytical performance comparison

Algorithm	Blocking time	Checkpoints	Messages
Koo-Toueg[2]	$N_{min} * T_{ckpt}$	N_{min}	$3*N_{min}* N_{dep} * (C_{wired} + C_{wl})$
Cao-Singhal[5]	0	N_{min}	$2*N_{min}*(C_{wired}+C_{wl})+min (N_{min}*(C_{wired} +C_{wl}), C_{broad})$
N-Fuchs[11]	0	N	$2*N_{min}*(C_{wired} +C_{wl})$
AK[6]	0	N	$N_{min}*(C_{wired} +C_{wl})$
Proposed	0	Nearest to N_{min}	$N_{min} *(C_{wired} + C_{wl})$

Our proposed checkpointing algorithm has the following characteristics:

Blocking time:

It is clear that the blocking time of our algorithm is 0.

Power consumption:

In our proposed checkpointing algorithm power efficiently is high compared to [6] and [11], as only minimum number of process are involved in determining the consistent global state. It does not awaken the processes in doze mode operation.

No. of coordinated message on wireless link:

It has very less coordinated message compared to [2], [5], [6] and [11] as it is takes decision about their checkpoint independently and only reply message are sent through the wireless links to their local MSS.

Conclusion

In this paper we presented an efficient time based coordinated checkpointing algorithm for mobile computing environments. Our work is an improvement over two phase algorithms [2], [4], [5], [7] and time based approaches [6][11][16].The algorithm has the following good features which makes it suitable for MDSs: (a)It does not use any extra message to coordinate and synchronize the clocks as clocks are attached with the application message which reduces the coordination message overheads. (b) It takes reduced number of checkpoints because a process does not take any temporary checkpoint and a process takes checkpoint if and only if it has sent or receives any message during its current checkpoint interval which helps in the efficient use of the limited resources of the mobile computing environment. (c) It is non-blocking and takes checkpoint decision independently from the other. (d) It does not require tracking and computation dependency information.

Hence our proposed algorithm takes reduced number of checkpoints, minimum interaction (only once) between the MHs and the MSS and no there is no synchronization delay. To achieve these all objective we use very simple data structure. These all features make our algorithm more suitable for mobile computing environment.

References

[1]. Acharya A. and Badrinath B.R., "Checkpointing Distributed Application on Mobile Computers", in the Proc. of the 3rd Int'l Conf. on Parallel and Distributed Information Systems, pp. 73-80, Sept. 1994.

[2]. Koo R. and Toueg S., "Checkpointing and Roll-Back Recovery for Distributed Systems", IEEE Trans. on Software Engg., Vol.13, No.1, pp.23-31, Jan. 1987.

[3]. Candy K.M. and Lamport L., "Distributed Snapshots: Determining Global State of Distributed Systems", ACM Trans. on Computing Systems, Vol. 3, No. 1,pp. 63-75, Feb.1985.

[4]. Cao G. and Singhal M., "On Coordinated Checkpointing in Distributed Systems", IEEE Trans. on Parallel and Distributed Systems, Vol. 9, No.12, pp. 1213-1225, Dec.1998.

[5]. Cao G. and Singhal M., "Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing Systems", IEEE Trans. on Parallel and Distributed Systems, Vol. 12, No.2, pp. 157-172, Feb. 2001.

[6]. Singh A.K., "On Mobile Checkpointing using Index and Time Together", World Acdemy of Science, Engineering and Technology, Vol 32, pp. 144-151, 2007.

[7]. Kumar P., Kumar L., Chauhan R.K. and Gupta V.K., "A Non-Intrusive Minimum Process Synchronous Checkpointing Protocol for mobile Distributed Systems", in the Proc. of the IEEE ICPWC-2005, Jan. 2005.

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

[9]. Elnozahy E.N., Johson D.B. and Zwaenepoel W., "The Performance of Consistent Checkpointing", in the Proc. of the 11th Symp. on Reliable Distributed Systems, pp. 39-47, Oct. 1992.

[10].Johnson, D.B., Zwaenepoel, W., "Sender-based message logging", In the Proc. of the 17th Int'l Symp. on Fault-Tolerant Computing, pp. 14-19, 1987.

[11].Neves N. and Fuchs W. K., "Adaptive Recovery for Mobile Environments", ACM Communication, Vol. 40, No.1, pp. 68-74, January 1997.

[12].Silva L.M. and Silva J.G., "Global checkpointing for distributed programs", in the Proc. of the 11th Symp. Reliable Distributed Systems, pp. 155-62, Oct. 1992.

[13].Storm R. and Temini S., "Optimistic Recovery in distributed Systems", ACM Trans. on Computer Systems, pp. 204-226, Aug. 1985.

[14].Prakash R. and Singhal M., "Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems", IEEE Trans. on Parallel and Distributed Systems, Vol. 7, No.10, pp1035-1048, Oct. 1996.

[15].Cao G. and Singhal M., "On the Impossibility of Min-process Non-blocking Checkpointing and an Efficient Checkpointing Algorithm for Mobile Computing", in the Proc. of the Int'l Conf. on Parallel Processing, pp.37-44, Aug. 1998.

[16].Neogy S., Sinha A., and Das P., "Distributed Checkpointing using Synchronized Clocks," in the Proc. the 26th IEEE Annual Int'l Conf. Computer Software and Applications (OMPSAC'02), pp. 199-206, 2002.

[17]. J. Ahn, S. Min, and C. Hwang, "A Casual Message Logging Protocol for Mobile Nodes in Mobile Computing Systems", Future Generation Computer Systems, Vol. 20, No. 4, pp. 663-686. May 2004.

[18].Jangra Surender et.al "Low Overhead Time Coordinated Checkpointing Algorithm for Mobile Distributed Systems", Computer Networks & Communications (NetCom), Volume 131, 1Pg. 173–182, Springer, New York, 2013, ISBN 978-1-4614-6153-1, ISSN 1876-1100

[19]. Surender Kumar, et.al., "Designing and Performance Analysis of Coordinated Checkpointing Algorithms for Mobile Distributed Systems", International Journal of Distributed and Parallel Systems [IJDPS] (AIRCC France), Vol.1, No.1, pp. 61-80, Sept. 2010. ISSN 2229-3957(Print), 0976-9757(Online).

[20].Manivannan D. and Singhal M., "Quasi-Synchronous Checkpointing: Models, Characterization, and classification", IEEE Trans. Parallel and Distributed System, Vol.10, No.7, pp.703-713, July 1997.

[21]. Manivannan D. and Singhal M., "A Low overhead Recovery Techniques using Quasi Synchronous Checkpointing ", in the Proc. of the 16th int'l conf. on Distributed Computing Systems, pp100-107, May 1996