A Study of Mutable Checkpointing Approach to Reduce the Overheads Associated with Coordinated Checkpointing

Dr. Amit Chaturvedi*, Syed Sajad Hussain** & Vikas Kumar***

*Head of MCA Department, Government Engineering College, Ajmer, Rajasthan, INDIA. E-Mail: amit0581@gmail.com
**Research Scholar, Bhagwant University, Ajmer, Rajasthan, INDIA. E-Mail: sajadsyed82@gmail.com
***Head of Computer Science Department, Bhagwant University, Ajmer, Rajasthan, INDIA. E-Mail: vikasmca51@gmail.com

***Abstract*—**As because of the new issues in mobile computing such as: lack of stable storage, low bandwidth of wireless channels, high mobility and limited battery life. So, coordinated check-pointing is a technique used for fault tolerant as it is domino free. It will deal with transparently fault tolerance to distributed applications. In this review paper, we have taken two objectives: (1) to minimize the number of synchronization messages and the number of checkpoints, (2) to make the checkpointing process non-blocking. We will propose the possible techniques to minimize the number of checkpoints that avoids the overhead of transferring large amount of data to the stable storage at Mobile Support Station (MSS).

***Keywords*—**Checkpointing; Coordinated Checkpoint; Message Logging; Mobile Computing; Mutable Checkpoint.

***Abbreviations*—**Consistent Global State (CGS); First- In First- Out (FIFO); Globally Consistent Checkpoints (GCCs); Mobile Host (MH); Mobile Service Stations (MSS).

I. INTRODUCTION

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. The following characteristics distinguish between distributed system and mobile computing systems: Limited Bandwidth, Limited and vulnerable MH local storage, frequent disconnection/connection, Limited power, Cost to locate MHs. A distributed system consists of several processes that execute on geographically dispersed computers and collaborate via message-passing with each other to achieve a common goal. In a traditional distributed system all hosts are stationary [Parveen & Poonam, 2010; Ajay & Praveen, 2010; Suparna & Sarmistha, 2010; Poonam & Parveen, 2010]. Where some of the processes run on mobile hosts moving over the network and a few fixed hosts Mobile Service Stations (MSS) act as access points to communicate with MHs. A distributed system is a collection of computers that are spatially separated and do not share a common memory. The processes executing on these computers communicate with one another by exchanging messages over communication channels [Suparna & Sarmistha, 2010; Poonam & Parveen, 2010]. Mobile hosts are increasingly becoming common in distributed systems due to their availability, cost, and mobile connectivity. A mobile host is a computer that may retain its connectivity with the rest of the distributed system through a wireless network while on move. A mobile host communicates with the other nodes of the distributed system via a special node called mobile service stations. MSS has both wired and wireless links and it acts as an interface between the static network and a part of the mobile network. Static nodes are connected by a high speed wired network. Mobile Computing technology allows transmission of data via a computer without having to be connected to a fixed physical link. Mobile Computing addresses those applications and technical issues that arise when persons move within a specific region or travel between countries and continents. This proves to be the best solution to the biggest problem of business people on the move [Suparna & N. Sarmistha, 2010; Parveen & Rachit, 2010].

II. CHECKPOINTING

Checkpointing is a well-established technique to deal with process failures and increase the system reliability and fault-tolerance in distributed systems. Checkpointing is also used

in debugging distributed programs and migrating processes in a multiprocessor system. In debugging distributed programs state changes of a process during execution are monitored at various time instances. Checkpoints assist in such monitoring. Checkpointing is the process of saving the status information. 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 [Parveen & Poonam, 2010; Ajay & Praveen, 2010].

A checkpoint is a snapshot of the local state of a process, saved on local nonvolatile storage to survive process failures [Bidyut et al., 2006]. Checkpointing, a process periodically provides the information necessary to move it from one processor to another. In checkpointing, the state of each process in the system is periodically saved on stable storage, which is called a checkpoint of a process. To recover from a failure, the system restarts its execution from a previous error- free, consistent global state. In a distributed system, since the processes in the system do not share memory, a global state of the system is defined as a set of local states, one from each process. The state of channels corresponding to a global state is the set of messages sent but not yet received. 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. A mobile system is a distributed system where some of processes are running on mobile hosts. The term "mobile" means able to move while retaining its network connection. A host that can move while retaining its network connection is an MH. An MH communicates with other nodes of system via special nodes called mobile support station [Acharya & Badrinath, 1994; Cao & Singhal, 1998; Weigang Ni et al., 2003; Parveen & Poonam, 2010; Ajay & Praveen, 2010; Poonam & Parveen, 2010].

Checkpoint may be local or global depending on taking the Checkpoint. Local checkpoint is an event that records the state of a process at processor at a given instance. A global checkpoint of an n-process distributed system consists of n checkpoints (local) such that each of these n checkpoints corresponds uniquely to one of the n processes. A global checkpoint M is defined as a Consistent Global State (CGS) if no message is sent after a checkpoint of M and received before another checkpoint of M [Bidyut et al., 2006; Parveen & Poonam, 2010]. The checkpoints belonging to a consistent global checkpoint are called Globally Consistent Checkpoints (GCCs) [Parveen & Poonam, 2010; Poonam & Parveen, 2010]. To recover from a failure, the system restarts its execution from the previous consistent global state saved on the stable storage during fault-free execution. This saves all the computation done up to the last check-pointed state and only the computation done there after needs to be redone [Parveen & Rachit, 2010]. The main motive of using Checkpointing is: (1)-To recover from failures**.** (2)-Check-pointing is also used in debugging distributed programs and migrating processes in multiprocessor system. (3)-To balance the load of processors in the distributed system, processes are moved from heavily loaded processors to lightly loaded ones. (4)-With check-pointing, an arbitrary temporal section of a program's runtime can be extracted for exhaustive analysis without the need to restart the program from beginning [Poonam & Parveen, 2010].

III. SYSTEM MODEL

A mobile computing system consists of a large number of MHs and relatively fewer MSSs. The distributed computation we consider consists of n spatially separated sequential processes denoted by P_0, P_1,... P_{n-1}, running on fail-stop MHs or on MSSs. Each MH or MSS has one process running on it. The processes do no share common memory or common clock. Message passing is the only way for the processes to communicate with each other. Each process progresses at its own speed and messages are exchanged through reliable channels, whose transmission delays are finite but arbitrary. The messages generated by the underlying computation are referred to as computation messages or simply messages, and are denoted by mi or m. We assume the processes to be non-deterministic [Prakash & Singhal, 1996; Guohong & Mukesh, 2001].

IV. MESSAGE LOGGING

Message-logging is very popular for building systems that can tolerate process crash failures. Message logging and checkpointing can be used to provide fault tolerance in distributed systems in which all inter-process communication is through messages. Each message received by a process is saved in message log on stable storage. No coordination is required between the check-pointing of different processes or between message logging and check-pointing. The execution of each process is assumed to be deterministic between received messages, and all processes are assumed to execute on fail stop processes. When a process crashes, a new process is created in its place. The new process is given the appropriate recorded local state, and then the logged messages are replayed in the order the process originally received them. All message-logging protocols require that once a crashed process recovers, its state needs to be consistent with the states of the other processes. This consistency requirement is usually expressed in terms of orphan processes, which are surviving processes whose states are inconsistent with the recovered states of crashed processes. Thus, message-logging protocols guarantee that upon recovery, no process is an orphan. This requirement can be enforced either by avoiding the creation of orphans during an execution, as pessimistic protocols do, or by taking appropriate actions during recovery to eliminate all orphans as optimistic protocols do [Alvisi et al., 1993; Alvisi & Marzullo, 1995].

A mobile support station, MSS_p, also maintains the message log in its volatile storage for the MH_s residing in the cell. Since a message heading for the MH_i should be routed

through the corresponding MSS_p, logging of messages into the volatile memory space incurs little overhead. Let M_i^a be the a-th message delivered to MH_i. Then, (i, a) is used as the identifier of M_i^a. The messages delivered to the MH_s in the cell are logged into the volatile storage of MSS_p, in the order that the message was sent from the MSS_p. MSS_p also logs the messages related to the mobility of MH_s, such as the join, leave, disconnect and reconnect messages received from the MHs. For each of these messages, MH_i attaches the value of $m_i^{rev\text{-}seq}$, which is logged with the message [Rao & Vin, 1998].

V. REVIEW OF TRADITIONAL CHECKPOINTING ALGORITHM

Parveen & Poonam (2010) have proposed a minimum process check-pointing protocol, where no useless checkpoints are taken. Also they tried to minimize the blocking of processes and to reduce the loss of check-pointing effort when any process fails to take its checkpoint. Their main concentration is to reduce check-pointing time and blocking time of processes. According to Chandy & Lamport (1985) algorithm, they have obtained by relaxing many of the assumptions made by them, a comparison of the salient features of various snapshot: the higher the level of abstraction provided by a communication model, the simpler the snapshot algorithm. The requirement of global snapshots finds a large number of applications like: detection of stable properties, checkpointing, monitoring, debugging, analyses of distributed computation, discarding of obsolete information, etc [Nigamanth Sridhar & Paolo A.G. Siviloti, 2002].

According to Poonam & Parveen (2010), they have proposed that time taken by checkpointing algorithms should be minimum during failure free run. Resources requirement for checkpointing should be minimum. Recovery should be fast in event of failure. Availability of consistent global state in stable storage expedite recovery. Parveen & Rachit (2010) tried to reduce the number of useless checkpoints and blocking of processes. Thus, the proposed protocol is simultaneously able to reduce the useless checkpoints and blocking of processes at very less cost of maintaining and collecting dependencies and piggybacking checkpoint sequence numbers onto normal messages. According to Guohong & Mukesh (2003), there does not exist a non-blocking algorithm which forces only a minimum number of processes to take their checkpoints, we proposed the concept of "mutable checkpoints" in implementing the non-blocking algorithm. Mutable checkpoints can be saved anywhere; e.g., the main memory or local disk. In this way, taking a mutable checkpoint avoids the overhead of transferring large amount of data to the stable storage at the file server across the network. Based on mutable checkpoints, our non-blocking algorithm avoids the avalanche effect and forces only a minimum number of processes to take their checkpoints on the stable storage.

Surender et al., (2010), have designed a minimum process non-blocking coordinating checkpointing protocols which are suitable for mobile distributed environment. The main feature of algorithm are: (1) The number of processes that take checkpoints is minimized to avoid awakening of MHs.(2) No useless checkpoint are taken. (3) If algorithm is non-blocking and not suspends their underlying computation during checkpointing. (5) Save limited battery life of MHs and low bandwidth of wireless channels. Bidyut et al., (2006), have presented a single phase non-blocking coordinated checkpointing approach suitable for mobile computing environment. Main features of the algorithm are: (1) it is free from the avalanche effect and minimum number of processes takes checkpoints; (2) it does not take any temporary, tentative, or mutable checkpoint unlike in some other important related works.

VI. REVIEW OF COORDINATED CHECKPOINTING APPROACH

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 [Parveen & Poonam, 2010]. Since every process always 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 eliminating the need for garbage collection [Ajay & Praveen, 2010]. In the first phase, processes take tentative checkpoints, and in the second phase, these are made permanent. The main advantage is that only one permanent checkpoint and at most one tentative checkpoint is required to be stored [Parveen & Rachit, 2010]. 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 [Koo & Toueg, 1987] and (b) non-blocking [Elnozahy et al., 1992; Guohong & Mukesh, 2001; Neogy et al., 2004]. 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 [Bidyut et al., 2006].

Prakash-Singhal algorithm (1996) was the first algorithm to combine min-processes and non-blocking, it forces only a minimum number of processes to take checkpoints and does not block the underlying computation during checkpointing [Parveen Kumar et al., 2005]. However, it was proved that their algorithm may result in an inconsistency. Cao & Singhal (1998) achieved non- intrusiveness in the minimum-process algorithm by introducing the concept of mutable checkpoints. The number of useless checkpoints in [Cao & Singhal, 1998]

may be exceedingly high in some situations [Ssu et al., 1999]. Higaki & Takizawa, (1999) and Ssu et al., (1999) 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 [Silva & Silva, 1992; Elnozahy et al., 1992; Guohong & Mukesh, 2001; Parveen Kumar, 2007].

VII. RELATED WORK

Cao & Singhal (1998) presents a non-blocking coordinated checkpointing algorithm with the concept of "Mutable Checkpoint" which is neither temporary nor permanent and can be converted to temporary checkpoint or discarded later and can be saved anywhere. In the scheme MHs save a disconnection checkpoint before any type of disconnection. This checkpoint is converted to permanent checkpoint or discarded later. In this scheme only dependent processes are forced to take checkpoints [Suparna & Sarmistha, 2010]. Pradhan et al., (1996) presented two un-coordinated protocol, first when a process receives a message, protocol creates checkpoint every time. The second protocol creates checkpoints periodically and logs all messages received. In communication induced checkpointing approach, a global checkpoint is similar to the approach of coordinated checkpointing while rollback propagation can be avoided by forcing additional un-coordinated local checkpoint in processes [Parveen Kumar, 2008].

Chandy-Lamport algorithm (1985) is the earliest non-blocking algorithm for static nodes. In this algorithm a markers are sent along all channels in the network and requires First In First Out (FIFO) channels. In coordinated algorithm, we may require piggybacking of integer checkpoint sequence number on normal messages. The first coordinated checkpoint protocol proposed that all communications are atomic, which is too restricted [Barigazzi & Strigni, 1983]. A single phase non-blocking coordinated checkpointing approach suitable for mobile computing environment. The main features of the algorithm are: (1) it is free from the avalanche effect and minimum number of processes takes checkpoints, (2) it does not take any temporary, tentative, or mutable checkpoint [Guohong & Mukesh, 2001].

VIII. CONCLUSION AND FUTURE SCOPE

As mobile computing faces many new challenges such as low wireless bandwidth, frequent disconnections and lack of stable storage at mobile hosts. These issues make traditional checkpointing techniques unsuitable to checkpoint mobile distributed systems. Minimum process Coordinated checkpointing is widely used technique in mobile distributed system as it requires less storage, bandwidth and have the characteristic of domino-free. To take a checkpoint, an MH has to transfer a large amount of checkpoint data to its local MSS over the wireless network. Since the wireless network has low bandwidth and MHs have low computation power, all-process checkpointing will waste the scarce resources of the mobile system on every checkpoint.

There are two issues that have been reviewed in this paper. To minimize the number of synchronous messages and the number of checkpoints for that the new concept introduced in the paper [Guohong & Mukesh, 2001] is "mutable checkpoint", which is neither a tentative checkpoint nor a permanent checkpoint, but it can be turned into a tentative checkpoint. 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 a large amount of data to the stable storage at MSSs over the wireless network. To make the checkpointing process non-blocking following steps may be taken : (1) the number of processes that take checkpoints is minimized to avoid awakening of MHs, (2) no useless checkpoint are taken (temporary, tentative, or mutable checkpoint), absence of these checkpoints means that much fewer number of control messages are needed, (3) If algorithm is non-blocking and not suspends their underlying computation during checkpointing, (4) save limited battery life of MHs and low bandwidth of wireless channels, and (5) reduces the latency associated with checkpoint request propagation compared to the traditional checkpointing algorithms [Koo & Toueg, 1987; Bidyut et al., 2006; Surender et al., 2010].

Hence the minimize the number of synchronous messages and the number of checkpoints and to make checkpointing process non-blocking are the two new areas for further research and study in mobile computing system.

REFERENCES

[1] G. Barigazzi & L. Strigni (1983), "Application -Transparent Setting of Recovery Points", *Digest of Papers Fault Tolerant Computing Systems-13*, Pp. 48–55.

[2] K.M. Chandy & L. Lamport (1985), "Distributed Snapshots: Determining Global State of Distributed Systems", *ACM Transaction on Computing Systems*, Vol. 3, No. 1, Pp. 63–75.

[3] R. Koo & S. Toueg (1987), "Checkpointing and Rollback-Recovery for Distributed Systems", *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 1, Pp. 23–31.

[4] L.M. Silva & J.G. Silva (1992), "Global Checkpointing for Distributed Programs", *Proceedings of 11th Symposium on Reliable Distributed Systems*, Pp. 155–162.

[5] E.N. Elnozahy, D.B. Johnson & W. Zwaenepoel (1992), "The Performance of Consistent Checkpointing", *Proceedings of the 11th Symposium on Reliable Distributed Systems*, Pp. 39–47.

[6] L. Alvisi, B. Hoppoe & K. Marzullo (1993). "Non-Blocking and Orphan-Free Message Logging Protocol", Proceedings of the 23rd International Symposium on Fault Tolerant Computing Systems, Pp. 145–154.

[7] A. Acharya & B.R. Badrinath (1994), "Checkpointing Distributed Applications on Mobile Computers", *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems*, Pp. 73–80.

[8] L. Alvisi & K. Marzullo (1995), "Message Logging: Pessimistic, Optimistic and Causal", *Proceedings of the 15th International Conference on Distributed Computing Systems*, Pp. 229–236.

[9] D.K. Pradhan, P. Krishna & N.H. Vaidya (1996), "Recovery in Mobile Environments: Design and Trade-Off Analysis", *Proceeding of the 26th IEEE International Symposium on Fault-Tolerant Computing*, Sendai, Japan, Pp. 16–25.

[10] R. Prakash & M. Singhal (1996), "Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems", *IEEE Transactions on Parallel Distributed System*, Vol. 7, No. 10, Pp. 1035–1048.

[11] L.A.S. Rao & H. Vin (1998), "The Cost of Recovery in Message Logging Protocols", *Proceedings of the 17th Symposium on Reliable Distributed Systems*, Pp. 10–18.

[12] G. Cao & M. Singhal (1998), "On the Impossibility of Min-process Non-blocking Checkpointing and an Efficient Checkpointing Algorithm for Mobile", *Proceedings of International Conference on Parallel Processing*, Pp. 37–44.

[13] G. Cao & M. Singhal (1998), "On Coordinated Check-Pointing in Distributed Systems", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 12, Pp. 1213–1225.

[14] H. Higaki & M. Takizawa (1999), "Checkpoint-Recovery Protocol for Reliable Mobile Systems", *Transactions of Information Processing Japan*, Vol. 40, No.1, Pp. 236–244.

[15] K.F. Ssu, B. Yao, W.K. Fuchs & N.F. Neves (1999), "Adaptive Checkpointing with Storage Management for Mobile Environments", *IEEE Transactions on Reliability*, Vol. 48, No. 4, Pp. 315– 324.

[16] C. Guohong & S. Mukesh (2001), "Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing Systems", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, No. 2, Pp. 157–172.

[17] Nigamanth Sridhar & Paolo A.G. Siviloti (2002), "Lazy Snapshots", *Proceedings of the 14th International Conference on Parallel and Distributed Computing and Systems*, Cambridge, MA, Pp. 96–101.

[18] C. Guohong & S. Mukesh (2003), "Checkpointing with Mutable Checkpoints", *Theoretical Computer Science*, Vol. 290, No. 2, Pp. 1127–1148.

[19] Weigang Ni, Susan V. Vrbsky & R. Sibabrata (2003), "Low-Cost Coordinated Nonblocking Checkpointing in Mobile Computing Systems", *Proceedings of the Eighth IEEE International Symposium on Computers and Communications*, Vol. 2, Pp. 1427–1434.

[20] S. Neogy, A. Sinha & P.K. Das (2004), "CCUML: A Check Pointing Protocol for Distributed System Processes", *IEEE Region 10 Conference (TENCON 2004)*, Thailand, Vol. B, No. 2, Pp. 553 – 556.

[21] Parveen Kumar, Lalit Kumar, R.K. Chauhan & V.K. Gupta (2005), "A Non-Intrusive Minimum Process Synchronous Checkpointing Protocol for Mobile Distributed Systems", *IEEE International Conference on Personal Wireless Communications*, Pp. 491–95.

[22] G. Bidyut, R. Shahram & L. Ziping (2006), "A New High Performance Checkpointing Approach for Mobile Computing", *International Journal of Computer Science and Network Security (IJCSNS)*, Vol. 6, No. 5B, Pp. 95–104.

[23] Parveen Kumar (2007), "A Low-Cost Hybrid Coordinated Checkpointing Protocol for Mobile Distributed Systems", *Mobile Information Systems*, Vol. 4, No. 1, Pp. 13–32.

[24] Parveen Kumar (2008), "A Low-Cost Hybrid Coordinated Checkpointing Protocol for Mobile Distributed Systems", *Journal of Mobile Information Systems*, Vol. 4, No. 1, Pp. 13-32.

[25] K. Surender, R.K. Chauhan & K. Parveen (2010), "A Low Overhead Minimum Process Global Snapshot Collection Algorithm for Mobile Distributed Systems", *International Journal of Multimedia and its Applications (IJMA)*, Vol. 02, No. 02, Pp. 12–30.

[26] K. Parveen & G. Poonam (2010), "A Low-Overhead Minimum Process Coordinated Checkpointing Algorithm for Mobile Distributed System", *Global Journal of Computer Science and Technology (GJCST)*, Vol. 10, No. 6, Pp. 30–36.

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

[28] B. Suparna & N. Sarmistha (2010), "A Mobility based Checkpoint Protocol for Mobile Computing System", *International Journal of Computer Science and information Technology (IJCSIT)*, Vol. 2, No. 1, Pp. 135–151.

[29] G. Poonam & K. Parveen (2010), "Review of Some Check-Pointing Algorithms in Distributed and Mobile Systems", *International Journal of Engineering Science and Technology*, Vol. 2, No. 6, Pp. 1594–1602.

[30] K. Parveen & G. Rachit (2010)," Soft-Check-Pointing based Coordinated Checkpointing Protocol for Mobile Distributed Systems", *International Journal of Computer Science Issues (IJCSI)*, Vol. 7, Issue 3, No. 5, Pp. 40–46.

Dr. Amit Chaturvedi, Head, MCA Department, Government Engineering College, Ajmer have completed Ph.D. (Computer Science) in March, 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 finesse.

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.

Vikas Kumar has received M. Tech degree in (CSE) branch. He is currently Head of Department of Computer Science at Bhagwant University Ajmer Rajasthan. He has nine years of teaching experience in the said field.