International Journal of Information Technology and Knowledge Management
July-December 2011, Volume 4, No. 2, pp. 559-567

# NEW PARADIGMS IN CHECK-POINTING TECHNIQUES IN DISTRIBUTED MOBILE SYSTEMS

Ruchi Tuli[1] & Parveen Kumar[2]

Distributed systems today are ubiquitous and enable many applications, including client–server systems, transaction processing, the World Wide Web, and scientific computing, among many others. Distributed systems are not fault-tolerant and the vast computing potential of these systems is often hampered by their susceptibility to failures. Many techniques have been developed to add reliability and high availability to distributed systems. These techniques include transactions, group communication, and rollback recovery. These techniques have different tradeoffs and focus. This paper surveys the various algorithms from the literature for checkpointing Mobile Computing systems, which restore the system back to a consistent state after a failure.

Keyword: Mobile Computing Systems, Co-ordinated Checkpoint, Rollback Recovery, Mobile Host, Mobile Support Station

## 1. INTRODUCTION

Rollback recovery treats a distributed system application as a collection of processes that communicate over a network. It achieves fault tolerance by periodically saving the state of a process during the failure-free execution, enabling it to restart from a saved state upon a failure to reduce the amount of lost work. The saved state is called a checkpoint, and the procedure of restarting from a previously checkpointed state is called rollback recovery. A checkpoint can be saved on either the stable storage or the volatile storage depending on the failure scenarios to be tolerated.

In distributed systems, rollback recovery is complicated because messages induce inter-process dependencies during failure-free operation. Upon a failure of one or more processes in a system, these dependencies may force some of the processes that did not fail to roll back, creating what is commonly called a rollback propagation. To see why rollback propagation occurs, consider the situation where the sender of a message m rolls back to a state that precedes the sending of m. The receiver of m must also roll back to a state that precedes m's receipt; otherwise, the states of the two processes would be inconsistent because they would show that message m was received without being sent, which is impossible in any correct failure-free execution. This phenomenon of cascaded rollback is called the domino effect. In some situations, rollback propagation may extend back to the initial state of the computation, losing all the work performed before the failure.

In a distributed system, if each participating process takes its checkpoints independently, then the system is susceptible to the domino effect. This approach is called independent or uncoordinated checkpointing [1], [2], [3]. It is obviously desirable to avoid the domino effect and therefore several techniques have been developed to prevent it. One such technique is coordinated checkpointing [4], [5], [6] where processes coordinate their checkpoints to form a system-wide consistent state. In case of a process failure, the system state can be restored to such a consistent set of checkpoints, preventing the rollback propagation. Alternatively, communication-induced checkpointing [7], [8], [9] forces each process to take checkpoints based on information piggybacked on the application messages it receives from other processes. Checkpoints are taken such that a system-wide consistent state always exists on stable storage, thereby avoiding the domino effect.

The approaches implement checkpoint-based rollback recovery, which relies only on checkpoints to achieve fault-tolerance. Log-based rollback recovery [10], [11], [12], [13], [14], [15], [16] combines checkpointing with logging of nondeterministic events. Log-based rollback recovery relies on the piecewise deterministic (PWD) assumption, which postulates that all non-deterministic events that a process executes can be identified and that the information necessary to replay each event during recovery can be logged in the event's determinant. By logging and replaying the non-deterministic events in their exact original order, a process can deterministically recreate its pre-failure state even if this state has not been checkpointed. Log-based rollback recovery in general enables a system to recover beyond the most recent set of consistent checkpoints. It is therefore particularly attractive for applications that frequently interact with the outside world, which consists of input and output devices that cannot roll back.

[1]Lecturer, Yanbu University College, Yanbu, Kingdom of Saudi Arabia

[2]Professor, Meerut Institute of Engineering & Technology, Meerut (INDIA)

E-mail: [1]tuli.ruchi@gmail.com

## 2. SYSTEM MODEL

Most of the algorithms considered in this paper use the common system model in which a mobile computing system consists of a set of mobile hosts (MHs) and mobile support stations (MSSs). The static MSS provides various services to support the MHs and a region covered by a MSS is called a cell. A wireless communication link is established between a MH and a MSS; and a high speed wired communication link is assumed between any two MSSs. The wireless links support FIFO communication in both directions between a MSS and the MHs in the cell wired links.

A distributed computation is performed by a set of MHs or MSSs in the network. A process experiences a sequence of state transitions for its execution and the atomic action which causes the state transition is called an event. An event is internal if it causes no interaction with another process. The message-sending and message-receipt events are the external events. A sequence of events within a process is called a computation. In case of a failure, a process stops the execution and does not perform any malicious action.

## 3. CHECKPOINTING ALGORITHMS FOR MOBILE COMPUTING SYSTEMS

Checkpointing techniques are studied under Asynchronous or uncoordinated, Synchronous or coordinated and quasi-synchronous or communication-induced checkpointing schemes. In this section, we discuss the various algorithms that have been proposed in literature for each of these schemes. Figure 1 shows the classification of these schemes.

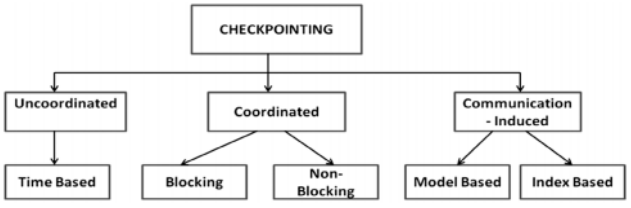


Fig.1: Classification of Checkpointing Schemes

### 3.1 Asynchronous or Uncoordinated or Independent Checkpointing

In uncoordinated checkpointing, each process has autonomy in deciding when to take checkpoints. This eliminates the synchronization overhead as there is no need for coordination between processes and it allows processes to take checkpoints when it is most convenient or efficient. The main advantage is the lower runtime overhead during normal execution, because no coordination among processes is necessary. Autonomy in taking checkpoints also allows each process to select appropriate checkpoints positions.

Park, Woo and Ycom[17] proposed an algorithm based on independent checkpointing and asynchronous message logging. All the messages are delivered to mobile host (MH) through MSS, so message logs are saved by MSS for all MHs in its vicinity. These logs are used to are used to recover state of process at MH after failure. Also, to reduce the message overhead, the mobile support stations also take care of the dependency tracking. The description of alroithms is as follows – When the mobility and failure rate of MHs is high then message logs of a MH may be distributed on number of MSSs which may be a number of hops away from current MSS. Distributed storage at MSS needs to be managed so as to reduce the cost of collection of message logs of MH after failure.

Park, Woo and Ycom[18] proposed a scheme based on the message logging and independent checkpointing, and for the efficient management of the recovery information, such as checkpoints and message logs, the movement-based scheme is suggested. This scheme allows the movement of checkpoint and message logs to a nearby MSS when either distance between MH and MSS on which latest checkpoint is saved exceed a threshold value, or, when number of handoffs that number of MSS carrying message logs of a MH exceeds a threshold value. These schemes keep the recovery information of MH in certain range. The movement-based scheme considers both of the failure-free execution cost and the failure-recovery cost. The scheme is explained as – Each mobile host MH periodically takes a checkpoint and the time period between two consecutive checkpointing of MH is determined by itself. For checkpointing, each MH first saves its current state as a checkpoint and assigns a unique sequence number to the checkpoint. $C_i^{\alpha}$ denotes the $\alpha$-th checkpoint of $MH_i$ and the pair of integers $(i, \alpha)$ is used as identifier for $C_i^{\alpha}$. Each MH also stores Id of center MSS in its stable storage and each MSS has a set of MSS which comes under its scope which is measured according to threshold distance between two MSS. When a MH moves between two cells, its recovery information is not required to be moved, if a MH moves from $MSS_p$ to $MSS_q$ belonging to the scope of $MSS_p$. The biggest disadvantage of this scheme is that once a MH moves to another cell not in the scope of center MSS of MH, all the recovery information of MH need to be migrated and if it returns back partial recovery information may be unnecessary migrated.

Zhang, Zuo, Zhi- Bowu and Yang [19] improved this scheme by migrating only partial recovery information of a MH when a MH moves out of the range. It means that recovery information of MH which is contained in some MSS due to mobility, is mapped to another set of MSSs. These MSSs are given by route function. The main advantage of this scheme is that one MSS is not burdened by transferring all the information to it.

Another movement-based algorithm was proposed by E. George, Chen and Jin [20] in which each MH maintain a counter which is incremented by 1 when MH performs a

handoff to another cell. Once this counter becomes greater than a predefined value, a checkpoint is taken. This counter depends on the user's mobility rate, failure rate and log arrival rate. Each MH also maintains a set of MSS which stores MH's log after latest checkpoint. When a MH performs a handoff, a new MSS is added to this set if MH sends atleast one message in the cell to the new MSS and if MSS has already not been added to the set. MSS logs messages before sending them to the destination. These messages are retrieved from MSS to recover a failure free state of MH after failure occurs. Once a new checkpoint is successfully taken by MH, set of MSS stored in MH is cleared and a message is sent to the MSS in the set to clear the log related to MH. Thus, storage overhead is reduced.

## 3.2 Coordinated Checkpointing

In this we will discuss the algorithms for both blocking and non-blocking coordinated checkpointing schemes.

### 3.2.1 Blocking Coordinated Checkpointing

A straightforward approach to coordinated checkpointing is to block communications while the checkpointing protocol executes. After a process takes a local checkpoint, to prevent orphan messages, it remains blocked until the entire checkpointing activity is complete. The coordinator takes a checkpoint and broadcasts a request message to all processes, asking them to take a checkpoint. When a process receives this message, it stops its execution, flushes all the communication channels, takes a tentative checkpoint, and sends an acknowledgment message back to the coordinator. After the coordinator receives acknowledgments from all processes, it broadcasts a commit message that completes the two-phase checkpointing protocol. After receiving the commit message, a process removes the old permanent checkpoint and atomically makes the tentative checkpoint permanent and then resumes its execution and exchange of messages with other processes. A problem with this approach is that the computation is blocked during the checkpointing.

A two-level blocking checkpointing algorithm was proposed by Lotfi, Motamedi and Bandarabadi [21] in which local and global checkpoint are taken. Nodes take local checkpoint according to checkpoint interval calculated previously based on failure rate and save it in their local disk. These checkpoints when sent to stable storage become global checkpoint. Local checkpoints are used to recover from more probable failures where as global checkpoints are used to recover from less probable failures. After each checkpointing interval, system determines expected recovery time in case of permanent failure. System calculates amount of time taken (T1) to recover if system does not take global checkpoint and amount of time taken (T2) to recover if system takes global checkpoint. Then system compares these two times. If T2 < T1, system will take global checkpoint else system will only store checkpoint locally.

Lalit - P. Kumar algorithm for mobile distributive systems [22]: in 2007 Lalit Kumar Awasthi and P. Kumar presented a new algorithm for synchronous checkpointing protocol for mobile distributed systems. In the algorithm they reduced the useless checkpoints and blocking using a probabilistic approach that computes an interacting set of processes on checkpoint initiation. A process checkpoint if the probability that it will get a checkpoint request in current initiation is high. A few processes may be blocked but they can continue their normal computation and may send messages. They also modified methodology to maintain exact dependencies. They show that their algorithm imposes low memory and computation overheads on MHs and low communication overheads on wireless channels. It avoids awakening of a MH if it not required to take its checkpoint. A MH can remain disconnected for an arbitrary period of time without affecting checkpointing activity.

Another blocking coordinated scheme is proposed by Suparna Biswas and Sarmistha Neogy [23] in which each MSSp is required to maintain an array A[n] in which A[1] is 1 when MH1 is present in vicinity of cell of MSSp where number of MH (Mobile Host) are n starting from 0 to n-1. A MH initiates checkpointing procedure, calculates its dependency vector D and sends request to all the MH whose bit in dependency vector D is 1 via its MSS. If a MH is present in vicinity of current MSS, then checkpoint request is send directly to MH. Else current MSS will broadcast checkpoint request message to other MSS so that it can reach all those processes whose bit is 1 in dependency vector D calculated by checkpoint initiator. Thus all these processes take checkpoint and sends information to initiator via their local MSS.

Guohui Li and LihChyun Shu [24] designed an algorithm to reduce blocking time for checkpointing operation, in which each process Pi maintains a set of processes Si. A process Pj is included in this set if Pj has sent at least one message to Pi in current checkpoint interval. Checkpointing dependency information is transferred from sending process to destination process during normal message transmission. So when a process starts a checkpointing procedure, it knows in advance the processes on which it depends both transitively and directly.

Biswas & Neogy [25] proposed a checkpointing and failure recovery algorithm where mobile hosts save checkpoints based on mobility and movement patterns. Mobile hosts save checkpoints when number of hand-offs exceed a predefined handoff threshold value. They introduced the concept of migration checkpoint An MH upon saving migration checkpoint, sends it attached with migration message to its current MSS before disconnection. The latest MSS of disconnected MH participates in checkpointing with m_checkpoint hiding the fact that the MH is still disconnected. During checkpointing participating MHs are barred only from receiving execution

message as it will change list of dependent MHs in current checkpoint interval. The algorithm is described as - In a mobile computing system all the MHs are connected to MSSs. MHs move in any possible direction with a fixed mobility rate in the system assumed in algorithm. As MHs move hand-off will occur. If hand-off count exceeds predefined threshold value, it initiates checkpoint protocol, saves a temporary checkpoint and sends checkpoint initiation message to its current MSS. Current MSS now coordinates checkpointing. MSSco forwards checkpoint request message to all MHs dependant on initiator MH during current checkpoint interval. MHDs save temporary checkpoint and send reply to MSSco. MSSco converts MHi's temporary checkpoint to permanent checkpoint and forwards reply all MHDs. MHDs convert temporary checkpoint to permanent checkpoint and send commit message to MSSco.

### 3.2.2. Non-Blocking Coordinated Checkpointing Algorithm

In this approach the processes need not stop their execution while taking checkpoints. A fundamental problem in coordinated checkpointing is to prevent a process from receiving application messages that could make the checkpoint inconsistent. Consider the example in Figure 2: message m is sent by $P_0$ after receiving a checkpoint request from the checkpoint coordinator. Assume m reaches $P_1$ before the checkpoint request. This situation results in an inconsistent checkpoint since checkpoint $c_{1,x}$ shows the receipt of message m from $P_0$, while checkpoint $c_{0,x}$ does not show m being sent from $P_0$. If channels are FIFO, this problem can be avoided by preceding the first post-checkpoint message on each channel by a checkpoint request, forcing each process to take a checkpoint before receiving the first post-checkpoint message, as illustrated in Figure 2(b).

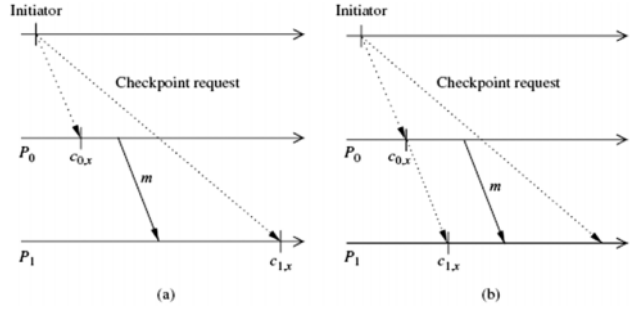


Fig.2: Non-Blocking Coordinated Checkpointing : (a) Checkpoint inconsistency (b) Solution with FIFO channels

Cao and Singhal presents in [26] 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, e.g., the main memory or local disk of MHs. In this 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. In this way, taking a mutable checkpoint avoids the overhead of transferring large amounts of data to the stable storage at MSSs over the wireless network. They presented techniques to minimize the number of mutable checkpoints. By simulation results they show that the overhead of taking mutable checkpoints is negligible. Based on mutable checkpoints, non-blocking algorithm avoids the avalanche effect and forces only a minimum number of processes to take their checkpoints on the stable storage.

Cao-Chen-Zhang-He [27] proposed an algorithm for Hybrid Systems. He presented an algorithm which was developed for integrating independent and coordinated checkpointing for application running in a hybrid distributed system containing multiple heterogeneous systems. The algorithm has many advantages mainly its easy to implement, no change is required for subsystems with coordinated checkpointing schemes and low extra workload for the coordinated checkpointing subsystem.

Bidyut – Rahimi- Liu [28]: in 2006 presented their work for mobile computing systems. In that work they presented a single phase non blocking coordinated checkpointing suitable for mobile systems. This algorithm produces a consistent set of checkpoints without the overhead of temporary checkpoints.

Bidyut-Rahimi-Ziping Liu [29] presented non blocking checkpointing and recovery algorithms for bidirectional networks. The proposed algorithm allowed the process to take permanent checkpoints directly, without taking temporary checkpoint global snapshot algorithms for large scale distributed systems. Whenever a process is busy it takes a checkpoint after completing its current procedure. The algorithm was designed and simulate for Ring network.

Ch. D. V Subha Rao and M.M Naidu [30] introduced the concept of active interval for handling orphan messages and lost messages. Active interval is the time that elapses between two events of sending "prepare checkpoint" and "take checkpoint" message by initiator to all the processes. Messages are said to be lost if it is sent in active interval of a process $P_i$ to $P_j$ but received after active interval of Pj or not received at all. A message becomes an orphan message if it is sent by sender after its active interval and received by receiver before or in the active interval (AI). Three Counters are maintained – one for counting number of messages sent by a process i in its AI denoted by Sendi, two counters for receiving messages by process j before AI and in AI denoted by RMj_before and RMj_in respectively. By simple arithmetic (number of messages that are lost by process j = Sendi – (RMj_before RMj_after )) we can determine number of messages that are lost so that they can be replayed from sender side log. Orphan messages are handled by

allowing receiver to maintain SSN for each message in its latest checkpoint which is count of number of messages received by the process uptil the last checkpoint.

If sender tries to replay any message whose SSN is less than or equal to SSN of receiver, receiver discard it as orphan message.

Awadesh kumar Singh in [31] proposed a non blocking algorithm in which a predefined checkpoint interval T is set on timers of all the MHs which is a deadline to take next checkpoint if process has sent any message in current checkpoint interval. Initially, every process sends a snapshot of its initial state (termed as 0th checkpoint) to its local MSS. A computation message is piggybacked with two values i.e. checkpoint sequence number Nj of sender Pj and local timer of MSS which is closest to receiver. Receiver sets its timer equal to this received time to achieve synchronization. On reception of computation message m by a process Pi, it checks its local timer has expired or not. If local timer has not expired, then it checks whether Nj of sender Pj is greater than Ni of receiver Pi. If it is true then Pi sets flag variable (If set to 0, process take checkpoint after expiry of its local timer otherwise checkpoint is not taken after expiry of local timer) to 1 and takes checkpoint if atleast one message is sent by process Pi in current checkpoint interval. Afterwards, Pi processes the received message m. As flag is set to 1 so Pi does not take checkpoint after expiry of its local timer. If Nj of sender Pj is not greater than Ni of receiver Pi, Pi processes the message m without taking checkpoint. Pi then takes checkpoint after expiry of local timer if it has sent atleast one message in current checkpoint interval. A global checkpoint consists of all the Nth checkpoints sent to stable storage by each process.

Partha Sarathi Mandal and Krishnendu Mukhopadhyaya [32] proposed a non blocking algorithm that uses the concept of mobile agent to handle multiple initiations of checkpointing. Mobile Agent has id same as its initiator id and it migrates among processes, perform some work, take some actions and then moves to other node together with required information. Each process takes initial permanent checkpoint and sets version number of checkpoint to 0. Process sends application message m by piggybacking it with version number of its latest checkpoint. Receiver compares application message's version number with its own current checkpoint version number to decide whether to take checkpoint first or simply to process message only. There is a DFS which is maintained by each process which contains id of neighbors on which the process depends.

### 3.3 Communication Induced or Quasi-Synchronous Checkpointing

It lies between synchronous and asynchronous (independent) checkpointing. Process takes communication induced checkpoints besides independent checkpoint to reduce number of useless checkpoints taken in independent checkpointing approach. Processes takes two kinds of checkpoints called local checkpoints and forced checkpoints. Local checkpoints are just like independent checkpoints taken in independent checkpointing approach. Forced checkpoints are taken to guarantee eventual progress of recovery line. During normal operation, checkpoints are taken normally but when failure happens then a recovery line is found to determine consistent global checkpoint among multiple checkpoints taken by each process.

Qiangfeng Jiang and D. Manivannan [33] presented an optimistic checkpointing and selective message logging approach for consistent global checkpoint collection in distributed systems. In this work they presented a novel quasi-synchronous checkpointing algorithm that makes every checkpoint belong to a consistent global checkpoint. Under this algorithm every process takes tentative checkpoints and optimistically logs messages received after a tentative checkpoint is taken and before the tentative checkpoint is finalized. Since tentative checkpoint can be taken any time and sorted in local memory, tentative checkpoints taken can be flushed to stable storage anytime before that checkpoint is finalized.

Ajay D Kshemkalyani algorithm [34] presented a fast and message efficient algorithm and show that new algorithm is more efficient. He presented two new algorithms Simple Tree and Hypercube that use fewer message and have lower response time and parallel communication times. In addition the hypercube algorithm is symmetrical and has greater potential for balanced workload and congestion freedom. This algorithm have direct applicable in large scale distributed systems such as peer to peer and MIMD supercomputers which are a fully connected topology of a large number of processors. This algorithm is also useful for determine checkpoint in large scale distributed mobile systems.

Jin Yang, Jiannong Cao, Weigang Wu [35] proposed a communication induced checkpointing scheme in which communication induced or forced checkpoints are taken by a process by analyzing piggybacked information that comes with received message. Each process has a logical clock or counter which is increased with every new checkpoint taken. When a process sends an application message, it piggybacks recent value of logical clock on message. Receiver compares its LC (logical clock) with received LC to decide whether to take a forced checkpoint before processing message or simply process the message.

Algorithm uses a Mobile Agent (MA) system which has a globally unique id. Each MA executes on a node and takes an independent checkpoint before migration. It then determines next host to which it has to migrate, it reaches on that host and takes a checkpoint on it. This process will continue until all hosts have been visited. These checkpoints are called local checkpoints.

Table 1
Comparison of Various Checkpoint Algorithms for Mobile Computing Systems

| S. No. | Algorithm | Features | Advantages | Disadvantages | Approach |
|---|---|---|---|---|---|
| | | | Uncoordinated Checkpointing | | |
| 1. | T.Park [17], 2002 | An algorithm based on independent checkpointing, optimistic and asynchronous message logging is proposed | 1. Recovery of an MH is performed independently<br>2. Message logging & dependency tracking is performed by MSS to utilize volatile space at MSS | 1. Distributed storage at MSS need to be managed so as to reduce the cost collection of message logs of MH after failure | Centralized |
| 2. | T.Park [18], 2003 | Distance and frequency based movement scheme in which managing distributed storage of message logs of MH when a mobile host is saving message logs at these MSSs is described. | 1. Movement based scheme used to handle distributed storage, controls transfer cost as well as recovery cost | 2. All the recovery information of MH needs to be migrated to MSS when it moves out of the scope of current MSS<br>3. Only partial recovery information may be available if MH returns back to previous MSS | Centralized |
| 3. | Zhang [19], 2008 | Recovery information of MH which is contained in some MSS due to mobility, is mapped to another set of MSSs when distance between MH & MSS exceed threshold value. These MSSs are given by route function. | 1. Independent checkpointing algorithm<br>2. Failed nodes can recover independently<br>3. All the recovery information of a MH is not transferred to single MSS, so one MSS will not be bottleneck of failure and access | 1. Storage overhead at MH is still not reduced | Centralized |
| 4. | E.George [20], 2006 | Independent checkpointing and optimistic message logging is used. MH takes checkpoint when its handoff_counter becomes greater than a predefined threshold. | 1. When a MH crosses a distance threshold from the location of latest checkpoint, recovery information is collected and transferred to MH's local MSS<br>2. Storage management is done by removing the log entry from MSS when checkpoint is successfully taken. | | Distributed |
| Blocking Coordinated Checkpointing | | | | | |
| 5. | Lofti [21], 2009 | A two level blocking checkpointing algorithm in which local and global checkpoints are taken | 1. Low overhead than one level scheme as local checkpoints are used for more probable failures and global checkpoints are used for less probable failures. | | Distributive |
| 6. | P. Kumar [22], 2007 | Synchronous checkpointing protocol for mobile computing systems in which useless checkpoints are reduced by using probabilistic approach that computes an interacting set of processes on checkpoint initiation | 1. low memory and computation overhead disconnected<br>2. Llow communication overhead on wireless channels | 1. A MH can remain disconnected for an arbitrary period of time without affecting checkpointing activity. | Distributive |

Table Contd...

Table 1 Contd...

| | | | | | |
|---|---|---|---|---|---|
| 7. | S. Negi [23], 2007 | A Blocking coordinated checkpointing scheme is described | 1. A solution how to handle mobility is given | 1. If a MH is present in vicinity of current MSS, then checkpoint request is send directly to MH. Else current MSS will broadcast checkpoint request message to other MSS<br>2. Thus all these processes take checkpoint and sends information to initiator via their local MSS | Centralized |
| 8. | Guohui [24], 2005 | A blocking coordinated checkpointing sceheme which forces only minimum number of processes to take checkpoint when a process initiate checkpoint | 1. Blocking time is reduced | 1. Storage overhead | Centralized |
| 9. | Biswas [25], 2010 | Algorithm to save checkpoints based on mobility and movement patterns. Mobile hosts save checkpoints when number of hand-offs exceed a predefined handoff threshold value | 1. They introduced the concept of migration checkpoint<br>2. An MH upon saving migration checkpoint, sends it attached with migration message to its current MSS before disconnection | 1. During checkpointing participating MHs are barred only from receiving execution | Distributive |
| Non-Blocking Coordinated Checkpointing | | | | | |
| 10. | Cao & Mukesh Singhal [26], 2001 | Introduced Mutable checkpoint, which is neither tentative or permanent checkpoint. | 1. Only dependent processes are forced to take checkpoints<br>2. Storage overhead is reduced | - | Distributive |
| 11. | Cao & Chen [27], 2004 | Algorithm for hybrid distributed systems | 1. Easy to implement<br>2. No change required for subsystems<br>3. No extra workload | - | Distributed |
| 12. | Bidyut, Rahimi and Liu [28], 2006 | Produce consistent set of checkpoint without overhead of temporary checkpoints, with no useless checkpoint, non blocking algorithm | 1. Checkpoint overhead is reduced | 1. Channel can loss messages | Distributive |
| 13. | Bidyut-Rahimi-Ziping Liu [29], 2008 | Directly permanent checkpoint without any temporary checkpoint sin ring network | 1. Eliminates the need of taking temporary checkpoint | - | Distributive |
| 14. | Ch. D. V Subha Rao and M.M Naidu [30], 2008 | Concept of Active interval is used to handle orphan and lost message | 1. Easier scheme to determine lost and orphan message | - | Distributive |

Table Contd...

Table 1 Contd...

| | | | | | |
|---|---|---|---|---|---|
| 15. | Awadesh kumar Singh [31], 2007 | A scheme based on independent checkpointing, optimistic and asynchronous message logging is proposed | 1. Recovery of an MH is performed independently<br>2. Message logging & dependency tracking is performed by MSS to utilize volatile space at MSS | - | Distributive |
| 16. | Partha Sarathi Mandal and Krishnendu Mukhopadhyaya [32], 2007 | A scheme uses the concept of mobile agents to handle multiple initiations of checkpointing is proposed. DFS tree saved at initiator is used to find neighbor node that has been visited by mobile agent. | 1. Number of moves that an agent takes to complete checkpointing may be very large | 1. Multiple concurrent checkpoints are possible | Distributive |
| Communication Induced Checkpointing | | | | | |
| 17. | Qiangfeng Jiang and D. Manivannan [33], 2007 | Every process can take their local tentative checkpoint and store in local memory | 1. Fast response time<br>2. Reduce overhead of checkpints | - | Distributive |
| 18. | Ajay D Kshemkalyani algorithm [34], 2010 | Simple tree and hypercube algorithms that use fewer message and have lower response time | 1. Useful in large distributive systems like supercomputers, MIMD, required less message and response time | - | Distributive |
| 19. | Jin Yang, Jiannong Cao, Weigang Wu [35], 2006 | Communication induced or forced checkpoints are taken by a process by analyzing piggybacked information that comes with received message | 1. Forced checkpoints maintain consistent recovery line | 1. Deferred message processing scheme allow delaying the processing of received message (that can lead to forced checkpoint) until mobile agent takes a basic checkpoint. Thus, forced checkpoints are avoided but some messages cannot be processed immediately | Distributive |

## 4. CONCLUSION

We have reviewed and compared different approaches for failure free execution of a mobile host and to a greater extent failure free execution of mobile environment. We studied three checkpointing scheme- independent, coordinated and communication induced checkpointing and the various algorithms that have been developed under each of these scheme. We have also compared different approaches of checkpointing and compared the salient features of various snapshot recording algorithms in Table 1. Clearly, 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.

## REFERENCES

1. Bhargava B. and Lian S.R., "Independent Checkpointing and Concurrent Rollback for Recovery in Distributed Systems – An Optimistic Approach," Proceedings of 17th IEEE Symposium on Reliable Distributed Systems, pp 3-12, 1988.
2. Storm R., and Temini, S., "Optimistic Recovery in Distributed Systems", ACM Trans. Computer Systems, Aug, 1985, pp. 204-226.
3. Weigang Ni, Susan V. Vrbsky and Sibabrata Ray, "Low-cost Coordinated Checkpointing in Mobile Computing Systems", Proceeding of the Eighth IEEE International Symposium on Computers and Communications, 2003.
4. Chandy K.M. and Lamport L., "Distributed Snapshots: Determining Global State of Distribited Systems", ACM Transaction on Computing Systems, 3 No. 1, pp 63-75, February, 1985

5. Koo R. and Tueg S., "Checkpointing and Rollback Recovery for Distributed Systems", IEEE Trans. On Software Engineering", 13 no. 1, pp 23-31, January 1987.
6. Elonzahy E.N., Alvisi L., Wang Y.M. and Johnson D.B., "A Survey of Rollback-Recovery Protocols in Message-Passing Systems", ACM Computing Surveys, 34 no. 3, pp 375-408, 2002.
7. Baldoni R., Hélary J-M., Mostefaoui A. and Raynal M., "A Communication- Induced Checkpointing Protocol that Ensures Rollback-Dependency Trackability," Proceedings of the International Symposium on Fault-Tolerant-Computing Systems, pp. 68-77, June 1997.
8. Hélary J. M., Mostefaoui A. and Raynal M., "Communication-Induced Determination of Consistent Snapshots," Proceedings of the 28th International Symposium on Fault-Tolerant Computing, pp. 208- 217, June 1998.
9. Manivannan D. and Singhal M., "Quasi-Synchronous Checkpointing: Models, Characterization, and Classification," IEEE Trans. Parallel and Distributed Systems, 10, no. 7, pp. 703-713, July 1999.
10. Alvisi, Lorenzo and Marzullo, Keith," Message Logging: Pessimistic, Optimistic, Causal, and Optimal", IEEE Transactions on Software Engineering, 24, No. 2, February 1998, pp. 149-159.
11. L. Alvisi, Hoppe, B., Marzullo, K., "Nonblocking and Orphan-Free Message Logging Protocol," Proc. of 23rd Fault Tolerant Computing Symp., pp. 145-154, June 1993.
12. L. Alvisi, "Understanding the Message Logging Paradigm for Masking Process Crashes," Ph.D. Thesis, Cornell Univ., Dept. of Computer Science, Jan. 1996. Available as Technical Report TR-96-1577.
13. Elnozahy and Zwaenepoel W, "On the Use and Implementation of Message Logging," 24th int'l Symp. Fault Tolerant Computing, pp. 298-307, IEEE Computer Society, June 1994.
14. D. Johnson, "Distributed System Fault Tolerance Using Message Logging and Checkpointing," Ph.D. Thesis, Rice Univ., Dec. 1989.
15. S. Venketasan and T.Y. Juang, "Efficient Algorithms for Optimistic Crash recovery", Distributed Computing, 8, no. 2, pp. 105-114, June 1994.
16. S. Venketasan and T.T.Y. Juang, "Low Overhead Optimistic Crash Recovery", Proc. 11th Int.
17. Taesoon Park, Namyoon Woo and Heon Y. Ycom, "An Efficient Optimistic Message Logging Scheme for Recoverable Mobile Computing Systems", IEEE Tran. On Mobile Computing, 2002.
18. Taesoon Park, Namyoon Woo and Heon Y. Yeom, "An Efficient Recovery Scheme for Fault Tolerant Mobile Computing Systems", FGCS- 19, 2003.
19. Yi-Wei ci, Zhan Zhang, De- Ching Zuo, Zhi- Bowu and Xiaa-Zong Yang, "Area Difference Based Recovery Information Placement for Mobile Computing System", 14th IEEE international Conference on Parallel and Distributed Systems, 2008.
20. Sapna E. George, Ing-Ray Chen and Ying Jin "Movement Based Checkpointing and Logging for Recovery in Mobile Computing Systems", ACM, June 2006.
21. Mehdi Lotfi, Seyed Ahmad Motamedi and Mojtaba Bandarabadi, "Lightweight Blocking Coordinated Checkpointing for Cluster Computer Systems", Sym. On System Theory, 2009.
22. Lalit Kumar P. Kumar "A Synchronous Ckeckpointing Protocol for Mobile Distributed Systems: Probabilistic Approach", Int Journal of Information and Computer Security, 2007.
23. Suparna Biswas and Sarmistha Neogy, "A Low Overhead Checkpointing Scheme for Mobile Computing Systems", Int. Conf. Advances Computing and Communications, IEEE 2007.
24. Guohui Li and LihChyun Shu "A Low-Latency Checkpointing Scheme for Mobile Computing Systems", Int. Conf. Computer Software and Applications, IEEE, 2005.
25. Biswas, S., & Neogy, S., "A Mobility-Based Checkpointing Protocol for Mobile Computing System", International Journal of Computer Science & Information Technology, 2, No.1, pp135-151, 2010.
26. G.Cao, M.Singhal, "Mutable Checkpoints: A New Checkpointing Approach for Mobile Computing Systems", IEEE Transactions on Parallel and Distributed System, 12, Issue 2, Feb., 2001, pages: 157-172, ISSN: 1045-9219.
27. Jiannog Cao, Yifeng Chen, Kang Zhang, Yanixing He: "Checkpointing In Hybrid Distributed Systems", Proceedings of 7th International Symposium of Parallel Architetures, Algorithms and Network, IEEE, 2004.
28. Bidyut Gupta, Shahram Rahimi, Ziping Liu: "A New High Performance Checkpointing Approach for Mobile Computing Systems", International Journal of Computer Science and Network Security, 2006.
29. Bidyut Gupta, Shahram Rahimi, and Ziping Liu: "Design of High Performance Distributed Snapshot/recovery Algorithms for Ring Network", Journal of Computing and Information Technology-CIT, 2008.
30. Ch. D. V Subha Rao and M.M Naidu "A New Efficient Coordinated Checkpointing Protocol Combined with Selective Sender based Message Logging", 2008.
31. Awadhesh Kumar Singh, "On Mobile Checkpointing Using Index and Time Together", World Academy of Science, Engineering and Technology, 2007.
32. Partha Sarathi Mandal and Krishnendu Mukhopadhyaya, "Mobile Agent Based Checkpointing with Concurrent Initiations", International J. of Foundation of Computer Science, 2007.
33. Qiangfeng Jiang and D. Manivannan: "An Optimistic Checkpointing and Selective Message Logging Approach for Consistent Global Checkpoint Collection in Distributed Systems", IEEE, 2007.
34. Ajay D Kshemkalyani:, "A Symmetric O(n log n) Message Distributed Snapshot Algorithm for Large Scale Systems", IEEE, 2010.
35. Jin Yang, Jiannong Cao and Weigang Wu, CIC: "An Integrated Approach to Checkpointing in Mobile Agent System", Proceedings of Second International Conference on Semantics, Knowledge and Grid, 2006.