Volume 1, No. 2, July-August 2010
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Hybrid Checkpointing and Recovery in Distributed Mobile Computing

Anil Kumar*
Research Scholar, CSE Department
Singhania University
Rajasthan, India
anil.panghal@rediffmail.com

Parveen Kumar
Professor, CSE Department
Meerut Institute of Engineering and Technology(MIET)
Meerut, India
pk223475@yahoo.com

Abstract: Checkpointing and restarting techniques are familiar ways to realize the reliable distributed mobile computation. In this paper, we propose a hybrid checkpointing protocol where the mobile Host (MH) asynchronously takes checkpointing protocol while the Mobile support stations (MSS) synchronously takes checkpoints. We have also shown that the hybrid checkpoint protocol implies the less total processing time than the synchronous checkpointing.
Keywords: Mobile Computing, Hybrid Checkpointing, Tentative checkpoint, message ordering, Restart Protocol.

I. INTRODUCTION

The mobile distributed system is composed of mobile hosts (MH) and mobile support stations (MSS) interconnected by communication networks. The mobile host move from one location to another in the network. There is a mobile support station (MSS) in each cell of the network. The mobile host can communicate with the MSS in the cell through the wireless channel. The Computation in distributed mobile systems is realized by cooperation of mobile hosts M_1, M_2 …M_m and fixed station $F_1,F_2,…F_f$. Each Mi is in one of the cells supported by MSSs S_1, S_2 , ..S_s. Here Mi is supported by S_j and S_j is the current MSS of Mi. The host exchange messages by using some mobile communication protocol. Each host can communicate with the others without being conscious of the locations of the hosts. We assume that state of every host is changed only if communication event occur.[2]

The synchronous checkpointing schemes have an advantage that the host can restart without domino effect. However, it is difficult for mobile hosts to take local checkpointing synchronously. Hence, hybrid checkpointing schemes have been proposed [1]

That having following properties:

- The fixed stations take local checkpoints by the synchronous scheme. A collection of the checkpoints taken by the fixed stations is referred to as a coordinated checkpoint
- The mobile hosts take local checkpoints by the asynchronous scheme.

For a local checkpoint of M_i, the state information of M_i is stored in the stable storage of a current MSS S_j. In addition, the messages sent and received by M_i are also stored in the stable storage of S_j. M_i fails to take checkpoint if the channel between M_i and S_j is disconnected. Thus, M_i can take checkpoint only if M_i does not move out of the cell and has enough capacity only if M_i does not move out of the cell and has enough capacity of the battery to take checkpoint. If both of these conditions are satisfied, M_i asynchronously takes the local checkpoint, i.e. independently of the other. The wireless channels are not so reliable as the cable channel. The mobile do not have so much capacity of battery that they communicate with other for a longer time.

II. HYBRID CHECKPOINTING

In the hybrid checkpointing protocol, the fixed stations $F_1,F_2,F_3,……F_f$ synchronously take a coordinated checkpoint (CC)=< $C_{f1},C_{f2},…..C_{Ff}$> While the mobile hosts $M_1,M_2,…..M_m$ take local checkpoint $C_{M1},C_{M2},……..C_{Mm}$ asynchronously. Each M_i has to restart the computation from a state consistent with CC. However, C_{Mi} is not always consistent with CC because each mobile host takes checkpoint independently of the other hosts. Hence, M_i restarts the computation by using message log. M_i stores the message sent and received after checkpoint C_{Mi} in the message log of the current MSS S_j. If M_i restarts the computation, M_i recomputes the messages stored in the message log to get the state consistent with CC.

Now, we discuss how each M_i takes C_{Mi}. Suppose that M_i is supported by S_j. Since every message sent and received by M_i is transmitted via S_j, the message can be stored in the stable storage of S_j even if M_i has no stable storage. The checkpoint agent process, A_{ij} in corresponding MSS, records messages sent and received by the mobile host M_i in the message log on the behalf of M_i. Moreover, A_{ij} takes the local checkpoint C_{Mi} of mobile host by recording the state information in the state log on the request of M_i.[1]

A checkpoint agent in MSS S_j takes a tentative local checkpoint (TC_{Mi}) independently of the other hosts, which is a collection of snapshots taken by A_{ij} and stored temporarily in the volatile storage of S_j. The information required for M_i to restart the computation from TC_{Mi} is carried by a tentative checkpoint request message. On receipt of checkpoint request message, the checkpoint agent A_{ij} stores the required information supplied by M_i in the volatile storage of S_j.

A. *Tentative Checkpointing*

1) Mobile host Mi sends a checkpoint request to a checkpoint agent Aij. Checkpoint request carries the state information of mobile host Mi.

2) On receipt of checkpoint request, checkpoint agent Aij takes the tentative local checkpoint TCMi of Mi by storing state information in volatile storage of Sj. If some checkpoint agent Aik (k<j) has taken another tentative checkpoint of mobile host Mij, Checkpoint agent Aij request Aik to discard this tentative checkpoint.

3) Now, on receiving the checkpoint request from coordinator station, checkpoint agent moves the state information from volatile storage to stable storage of mobile support station MSS in the state log.

4) Again if the MSS receives the request from checkpoint agent that has information about messages. It moves the message from volatile to stable storage of mobile support station in the message log.

B. Coordinated Checkpointing

Fixed stations take coordinated checkpoint by using the following protocol.

1) A coordinator station, CS sends a checkpoint request message to fixed stations and mobile support stations.

2) On receipt of checkpoint request, each fixed stations and MSSs take a tentative checkpoint and sends back a reply message (Crep) to coordinator station.

3) If coordinator station receives Crep from all the stations, CS sends a final message to all stations.

4) Now, each fixed stations and mobile support stations makes tentative checkpoint, permanent.

In order for CC to be consistent, each station suspends the computation and the transmission of application messages during the process of checkpointing.

The checkpointing protocol proposed here has the following properties:

- Each mobile host M_i has one permanent checkpoint that is consistent with recently coordinated checkpoint CC taken by the fixed stations.
- Each mobile host M_i at most one tentative checkpoint.

III. MESSAGE ORDERING FOR RECOVERY

In order that the checkpoint agent A_{ij} records the messages in the same order as handled in M_i, each message m carries two sequence numbers m.seq and m.ack. Here, let m.sender and m. receiver mean the sender and receiver of m, respectively.

- Message m has a unique sequence m.seq. If m is sent by a mobile host M_i after a message m', m.seq> m'seq. Sequence in M_i is incremented by one each time M_i sends a message.
- m.ack means that the sender m.sender, i.e. mobile host M_i or a checkpoint agent A_{ij} has received every message m' where m'.seq<= m.ack. Each time m.sender sends m, the sequence number m.seq of message m which is most recently received is stored to ack message m.

Message exchanged between checkpoint agent A_{ij} and mobile host M_i is recorded in the message log of stable storage of MSS in the order determined by the following rule. Suppose that checkpoint agent A_{ij} records m and m' message log of MSS

Ordering Rule

- If both messages m and m' are sent by the same sender, i.e. m.sender=m'.sender, m precedes m' if m.seq< m'.seq.
- If message m and m' are sent by different senders, i.e. m.sender ≠ m.sender, m precedes m' if m.seq< m'.ack. Otherwise, m' precedes m.

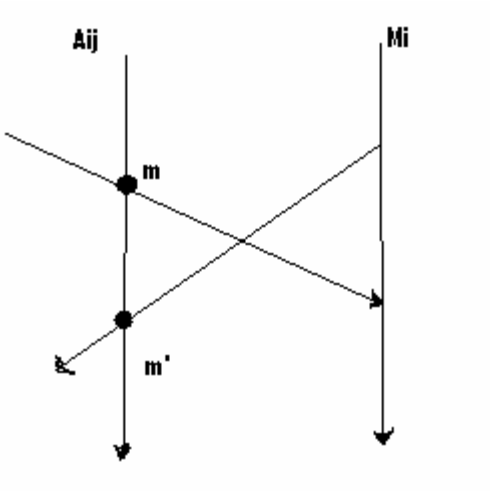

Figure 1- Crossing Messages

In Figure 1, m.seq< m'.ack because mobile host M_i sends m' before receiving m. Hence, a checkpoint agent A_{ij} stores m' before m in message log of MSS. Thus the sequence of messages in message log is same as M_i sends and receives the messages.

IV. RESTART PROTOCOL FOR RECOVERY

We now discuss how the fixed stations and mobile host restart. Fixed station restart from the state consistent with coordinated checkpoint CC by using the restart protocol.

1. The coordinator station CS sends request messages for restarting to all the fixed and mobile support stations.
2. On receipt of request message, each fixed station F_i and mobile support station S_j sends back a reply message to coordinator station.
3. If coordinator station CS receives reply message from all the stations. CS sends a final message to all fixed and mobile support stations.
4. On receipt of final message to all fixed station F_i and mobile support station Sj restart the computation from their local checkpoints (C_{Fi}) and (C_{Sj}), respectively.

In order to restart mobile hosts M_1, M_2,….,M_m from the states consistent with coordinated checkpoint CC, the mobile checkpoint agents have to cooperate. A recovery protocol for mobile hosts M_i is as follows:

1. If mobile supports station receives request message to restart, it sends a state log request to checkpoint agent A_{i1} that has the C_{Mi} and a message log request to every other checkpoint agents in mobile host.
2. On receipt of this request message, corresponding checkpoint agents sends back a reply message containing state and message information.
3. Then mobile host sends a tentative state and message log cancellation request message to checkpoint agent A_{it} that is handling the tentative checkpoint request on behalf of the mobile host M_i.
4. On receipt of state and message log cancellation request message, A_{it} discards further tentative checkpoint request and sends back a reply message to mobile support station.
5. Then mobile host gets a state consistent with CC by using the state information at C_{Mi}, carried state log information and recomputing the messages carried by the message log. The order of events to occur in recomputation in mo-

bile host M_i is determined by the sequence number as stored in message log table. Hence every M_i gets the state consistent with CC.

V. EXISTING CHECKPOINTING PROTOCOL

In the existing checkpointing protocol, each host takes checkpoints independently. Independently taken checkpoints are called basic checkpoints and those triggered by some message reception are called forced checkpoints. Each basic checkpoint request is generated after a fixed time interval. The checkpoint interval can very but no condition is checked for time interval. The checkpoint interval can vary but no condition is checked for checkpoints. Whenever faults occur, hosts rollback to some consistent state. This technique is although simple but may suffer form domino effect hence recovery time may be larger.[3][4]

Algorithm executed at a mobile host (M_i)

1. Begin
2. Initialize network variables like self and neighbor's id.
3. Create a message queue.
4. Generate the message.
5. Check (message_type)
6. If(message_type= = Normal)
   {
   Examine whether it is send or not.
   If (send)
   Send the message to particular host.
   Else
   Receive the message.
   }
7. If(message_type= = Checkpoint request)
   {
   Execute procedure checkpoint.
   }
8. If(message_type= =Failure Signal)
   {
   Send failure signal to MSS.
   Receive the message from MSS to suspend operation.
   }
9. If(message_type= =Recovery Signal)
   {
   Send signal to MSS that I'm going to recover.
   Wait,
   Collect all information for restarting by MSS.
   Restart from the last consistent point.
   }
10. Go to step 4
11. End

Algorithm executed at a Mobile Host for checkpoint

1. Begin procedure checkpoint
2. Send checkpoint_request to MSS.
3. {Record its local state, messages of mobile host.
   Transfers local_state, messages to MSS.
   }
4. Receive acknowledgement from MSS.
5. End procedure checkpoint

The mobile support station is the backbone of mobile computing system. Each mobile host can talk to other hosts only through MSS. Mobile support station takes the checkpoint of the particular mobile host when checkpoint request is generated. MSS send signal like suspend operation, wake up and also gives the consistent point from where mobile and fixed station can start their execution after failure and recovery.

Algorithm executed at a Mobile support station

1. Begin
2. For all mobile host and fixed station in the system
3. Receive the message.
4. Check(message_type)
5. If(message_type= = Normal)
   {
   Forward the messages to destination host.
   }
6. If(message_type= = Checkpoint request)
   {
   Take checkpoint for source mobile host.
   Update the message and state log for mobile host.
   Send acknowledgement to source mobile host.
   }
7. If (message_type= =Failure Signal)
   {
   Send signal to suspend working to source host and all the hosts which are communicating with faulty host.
   }
8. If(message_type= =Recovery Signal)
   {
   Scan the message and state log.
   Make a consistent global state.
   Send rollback signal to host and,
   Dependent hosts to their last consistent checkpoint state.
   }
9. Goto step 3
10. End

Fixed station simply generates the messages and when the checkpoint request message is generated it signals MSS sends checkpoint request to all Hosts. Then every host giving their checkpoint status to MSS and then MSS transfers to the fixed station and it make it to permanent. Because checkpoints are taken synchronously there is always a consistent state.

Algorithm executed at a Fixed Station

1. Begin
2. Initialize the network variables like self and neighbor's id.
3. Create a queue.
4. Generate the message.
5. Check(message_type)
6. If(message_type= = Normal)
   {
   Execute normal operation.
   }
7. If(message_type= = Checkpoint request)
   {
   Checkpoint request generated.
   Send signal to MSS, for taking checkpoint of all the hosts.
   Wait,
   Receive acknowledgement from MSS.

Collect information of all the hosts.
Make them permanent.
}

8. If(message_type= =Failure Signal)
{
Send failure signal to all hosts and MSS.
Suspend operation.
}

9. If(message_type= =Recovery Signal)
{
Send recovery signal to MSS and all hosts
Wait,
Collect all information for restarting by MSS.
Restart from the last consistent point.
}

10. Go to step 4

VI. PROPOSED MODIFICATION

We propose to verify certain conditions after reception of message instead of taking checkpoints after fixed time interval. This helps in reducing the recovery time.

When it is time for a host to take a basic checkpoint, it takes a basic checkpoint only if it did not already take a forced checkpoint with the sequence number that is expected to be assigning to the next basic checkpoint; otherwise, it skips taking the basic checkpoint. When a host receives a message, it takes a forced checkpoint before processing the message if the sequence number of its current checkpoint is less than the checkpoint sequence number received with message.

Each mobile host has two variables seq_noi and next. The value of the variable seq_no denotes the sequence number of the latest checkpoint taken by mobile host. The value of next denotes the sequence number to be assigned to the next basic checkpoint that mobile host will take. The variable next is incremented by host after every checkpoint interval. The sequence number piggybacked with the message is denoted by Message. Seq_no. The temporary storage of sequence number is denoted by tem. Seq_no. Since the protocol requires the control information to be piggybacked, while sending a message. The check pointing decision is taken while receiving a message.

Algorithm executed at a Mobile Host

- **Mobile host increment $next_i$ periodically after every checkpoint interval**

  $Next_i = next_i + 1$;

- **When it is time for mobile host to take basic checkpoints**
  1. If $next_i > seq\_no_i$
  2. Execute procedure checkpoint.
  3. Temp.seq_no = $next_i$.
  4. Seq_no_i = temp.seq_no.
- **While mobile host sending a message**
  1. Message.seq_no = seq_no_i.

  Sequence number of the current checkpoint piggybacked with message**.**
  2. Send it to particular mobile host.

While mobile host m_j receives a message from m_i

1. If message.seq_no > seq_no_j.
2. Execute procedure checkpoint.
3. Temp.seq_no = message.seq_no.
4. seq_noj = temp.seq_no.
5. Process the message.

VII. CONCLUSION

In the existing scheme the decision of checkpoint is taken after a fixed interval of time when there is a checkpoint request message generated. In this case, if we reduce the checkpoint interval then the number of checkpoints is increased but the rollback distance is decreased. The overall effect of time needed to process the messages is increased with increase of number of messages and mobile costs because the reduction of rollback time is counterbalanced by the overhead of checkpoint time. We propose the modification in which we verify certain condition after receiving the message, because, the checkpointing decision is selective and hence, the total number of checkpoints is reduced. Again the algorithm has no additional control message overhead. So total time to process the message is less as compared to existing protocol where the checkpoint overhead is high.

VIII. REFERENCES

[1] Hiroki Higaki, Makoto Takizawa "Hybrid Checkpointing in mobile Computing system"

[2] Perkins, C.,"IP Mobility Support" Internet Draft:draft-ietf-mobileip-protocol-12,1995

[3] Teraoka, F., Uehara, K., Sunahara, H., and Murai, J., "VIP: A Protocol providing Host Mobility," Comm. ACM,37(8),67-75,1994

[4] Koo, R. and Toueg, S., "Checkpointing and Rollback recovery for Distributed systems" IEEE Trans. on Software engineering, SE-13(1)

[5] Birman, K., and T. Joseph," Reliable Communications in the Presence of Failures," ACM Transaction on Computer System, vol. 5, no. 1, Feb. 1989, pp. 47-76.

[6] P. Kumar "A Low Cost Hybrid Coordinated Checkpointing Protocol for Mobile Distributed System" Mobile Information system" pp13-22 , Vol. 4 2007

[7] Parveen Kumar, Lalit Kumar, R K Chauhan "A Hybrid coordinated Checkpointing Protocol For Mobile computing systems", IETE Journal of Research Vol. 52 pp 247-254, 2006

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

[9] Singhal, M. , Shivaratri, N.-G.: Advanced Concept in Operating System. McGraw Hill,(1994)

[10] Kshemkalyanl Ajay D, Singhal, M.: Distributed Computing Principals, Algorithms, and System (2008)