Rollback-Recovery Protocols in Message-Passing Systems

Presented by Meilin Zhang
CS458

1

Agenda

- Objective & Background
- Checkpoint-based Rollback Recovery
- Log-based Rollback Recovery
- Conclusion

2

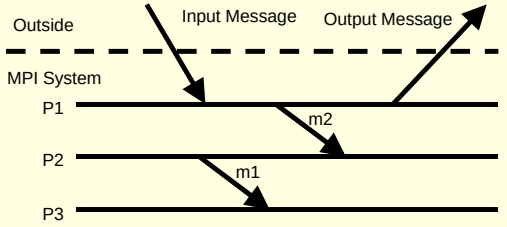
Objective

- To recover distributed system after failures, rollback recovery protocols have been introduced.
  - Hardware: Device aging, electromagnetic radiation, crosstalk coupling, …
  - Software
  - Network: Load surges in the network, message lost in the communication channel, …

3

System Model

- A message-passing system consists of a fixed number of processes
- All processes communicate only through messages
- A process may fail; Network may be unreliable

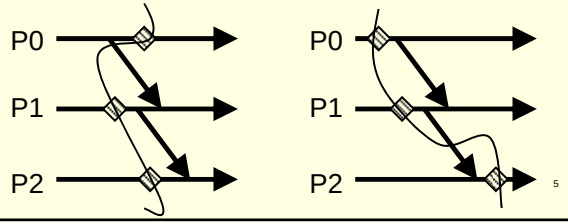


Correctness: "A system recovers correctly if its internal states is consistent with the observable behavior of the system before the failure."

4

Consistent System States

- A consistent global state is one that may occur during a failure-free execution of a distributed system.
- More precisely, a consistent system state is one in which if a process's state reflects a message receipt, then the state of the corresponding sender reflects sending that message.



5

Interactions with the Outside World

- The outside world can't roll back.
- Outside world process (OWP): A special process can't fail, maintain state or participate in the recovery protocol.
- Output commit problem: Before sending a message to OWP, the system must ensure the state from which the message is sent will be recovered in any case.

6

Agenda

- Objective & Background
- Checkpoint-based Rollback Recovery
- Log-based Rollback Recovery
- Conclusion

7

Rollback Recovery Mechanism

- Checkpoint-based Rollback Recovery
  - Upon a failure, a failed process use the saved information to restart the computation from an immediate state.
- Log-based Rollback Recovery
  - Upon a failure, a failed process use the saved information plus the logs.
  - Log: records the interactions with input and output devices, messages exchanged among the processes, and some internal events.

8

Uncoordinated Checkpointing

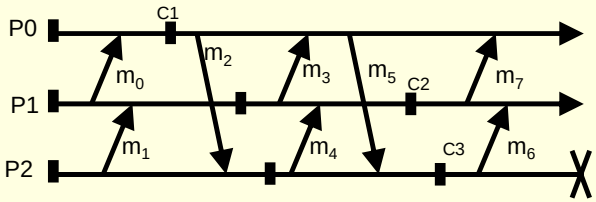
- Allows each process autonomy in deciding when to take checkpoints.

Pros	Cons
✓Each process may take a checkpoint when it is most convenient	✓A process may take a useless checkpoint that will never be part of a global consistent state.
✓A process may take checkpoints when the amount of state information to be saved is small	✓All processes have to maintain multiple checkpoints
	✓Domino Effect

9

Domino Effect

The cascaded rollback may cause the system to roll back to the beginning of the computation, in spite of all the saved checkpoints.



10

Blocking Checkpoint Coordination

Block communications while the checkpoint protocol executes

- Step 1. A coordinator takes a checkpoint and broad a request message to ask all the processes to take a checkpoint.
- Step 2. All the processes take a tentative checkpoint and send an acknowledgement.
- Step 3. The coordinator broadcasts a commit message.
- Step 4. All the processes remove the old permanent checkpoint and makes the tentative checkpoint permanent.

11

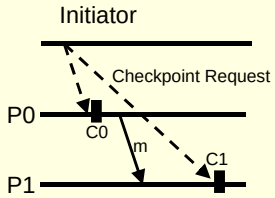
Pros & Cons

- Pros
  - Only needs one permanent checkpoint on stable storage
  - Reduces storage overhead and eliminate the need for garbage collection
- Cons
  - A global checkpoint is need before messages can be sent to OWP
  - Leads to large latency of committing output

12

Non-block Checkpoint Coordination

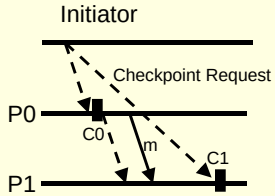
- A fundamental problem in coordinated checkpoint is to prevent a process from receiving application messages that could make the checkpoint inconsistency.



13

Distributed Snapshot

- The initiator takes a checkpoint and broadcasts a marker (a checkpoint request) to all processes.
- Each process takes a checkpoint upon receiving the first marker and rebroadcast the marker to all processes before sending any application message.

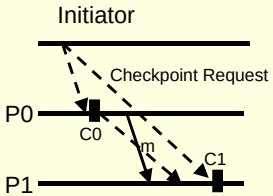


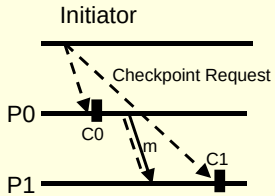
14

Distributed Snapshot with Non-FIFO Channel

Checkpoint Inconsistency

Checkpoint Request can be Piggybacked on every post-checking messages





15

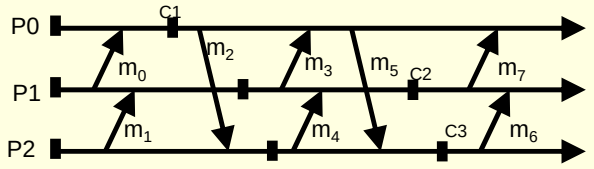
Agenda

- Objective & Background
- Checkpoint-based Rollback Recovery
- Log-based Rollback Recovery
- Conclusion

16

Log-based Rollback Recovery

- A process execution can be modeled as a sequence of deterministic state intervals, each starting with the execution of a nondeterministic event
- Nondeterministic event: the receipt of a message from another process or an event internal to the process
- Piecewise deterministic assumption (PWD)



17

Log-based Rollback Recovery (Cont.)

- During failure-free operation, each process logs the determinants of all the nondeterministic event.
- Each process can also takes checkpoints to reduce the extent of rollback during recovery.
- After a failure occur, the failed processes recover by using the checkpoints and logged determinants.

18

Always-no-orphans condition

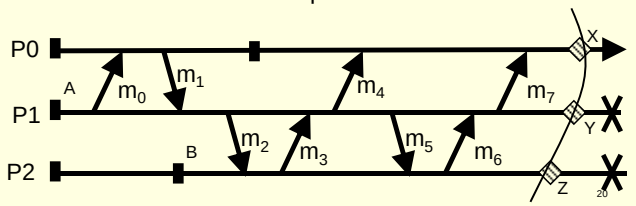
If any surviving process depends on an event e, either the event is logged on stable storage, or the process has a copy of the determinant of event e.

If neither condition is true, then the process is an orphan because it depends on an event that can't be generated during recovery

19

Pessimistic Logging

- Pessimistic logging protocols are designed under the assumption that a failure can occur after any nondeterministic event in the computation.
- The protocols log to stable storage the determinant of each nondeterministic event before the event is allowed to affect the computation.



20

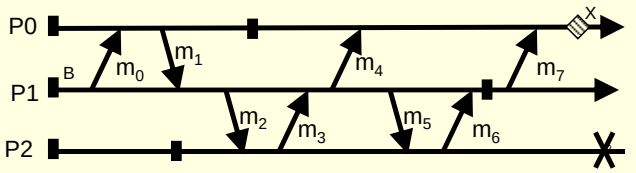
Pros & Cons

- Pros
  - Processes can send messages to the outside world without running a special protocol.
  - Processes restart from their most recent checkpoint upon a failure.
  - Recovery and garbage collection is simplified.
- Cons
  - The price is a performance penalty incurred by synchronous logging.

21

Optimistic Logging

- The protocols make the optimistic assumption that logging will complete before a failure occurs.
- Determinants are kept in a volatile log, which is periodically flushed to stable storage.
- Permit the temporary creation of orphan processes. However, always-no-orphans condition hold by the time recover is complete.



Pros & Cons

- Pros
  - Does not require the application to block waiting for the determinants to be actually written to stable storage, and therefore incurs little overhead during failure-free execution.
- Cons
  - May need to keep multiple checkpoints.
  - Output commit in requires multi-host coordination to ensure that no failure scenario can revoke the output

23

Conclusions

- Different rollback-recovery protocols offer different tradeoffs with respect to performance overhead, latency of output commit, storage overhead, ease of garbage collection, freedom from domino effect and orphan processes, and the extent of rollback.
- Coordinated checkpointing and pessimistic logging only need maintain one checkpoint, but incurring large overhead.
- Uncoordinated checkpoint can incur little overhead but have unbounded rollback.
- Several checkpoints may need to be kept under optimistic logging.

24