Checkpoint-based Recovery

Y. V. Lokeswari

Reference: Kshemkalyani, Ajay D., and Mukesh Singhal. Distributed computing: principles,
algorithms, and systems. Cambridge University Press, 2011.

Overview

- Checkpoint –based Recovery
  - Uncoordinated Checkpointing
    - Direct Dependency Tracking Technique.
  - Coordinated Checkpointing
    - Blocking Coordinated Checkpointing
    - Non-Blocking Coordinated Checkpointing
  - Communication-induced Checkpointing
    - Model-based Checkpointing
    - Index-based Checkpointing

Checkpoint-based Recovery

- In the checkpoint-based recovery approach, the **state of each process and the communication channel is checkpointed** frequently so that, upon a failure, the system can be restored to a globally consistent set of checkpoints.
- Checkpoint-based protocols are therefore **less restrictive and simpler to implement** than log-based rollback recovery.
- Checkpoint-based rollback recovery does not guarantee that **prefailure execution can be deterministically regenerated after a rollback.**
- Checkpoint-based rollback recovery may **not be suitable for applications** that require frequent interactions with the outside world.

Uncoordinated 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.
- **Advantages:**
  - **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.

Uncoordinated checkpointing

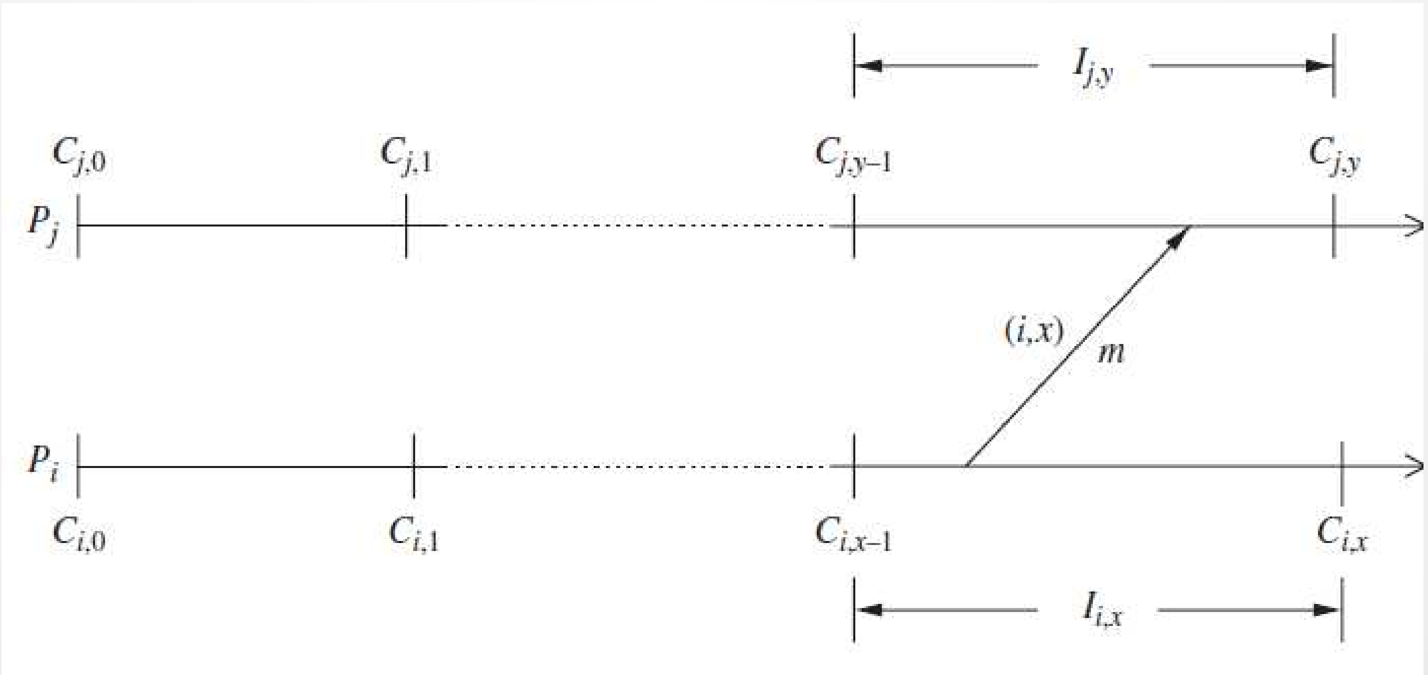
- **Disadvantages:**
  - The possibility of **the domino effect** during a recovery, which may cause the loss of a large amount of useful work.
  - **Recovery from a failure is slow** because processes need to iterate to find a consistent set of checkpoints.
  - Checkpoints taken by a process may be ***useless checkpoints***
  - Uncoordinated checkpointing forces each process to maintain multiple checkpoints, and to **periodically invoke a garbage collection algorithm** to reclaim the checkpoints that are no longer required.
  - **Not suitable for applications with frequent output commits** because these require global coordination to compute the recovery line

Uncoordinated checkpointing

- To determine a consistent global checkpoint during recovery, the processes **record the dependencies among their checkpoints**.

Direct Dependency Tracking Technique

- Assume each process Pi starts its execution with an initial checkpoint Ci,0
- Ii,x : checkpoint interval, interval between Ci,x−1 and Ci,x
- When Pj receives a message m during Ij,y , it records the dependency from $I i,x$ to Ij,y, which is later saved onto stable storage when Pj takes Cj,y



Uncoordinated checkpointing

Direct Dependency Tracking Technique

- When a failure occurs, the recovering process initiates rollback by **broadcasting a *dependency request message* *to collect all the dependency information*** maintained by each process.
- Receiving process stops its execution and **replies with the dependency information** saved on the stable storage as well as with the dependency information.
- The initiator then **calculates the recovery line** based on the global dependency information and **broadcasts a *rollback request*** *message containing the recovery line.*
- Upon receiving this message, **a process whose current state belongs to the recovery line** simply resumes execution; otherwise, **it rolls back to an earlier checkpoint as indicated by the recovery line.**

Coordinated checkpointing

- In coordinated checkpointing, processes **orchestrate their checkpointing activities** so that all local checkpoints form a consistent global state.
- **Advantages**
  - Coordinated checkpointing **simplifies recovery** and is **not susceptible to the domino effect,** since every process always restarts from its most recent checkpoint
  - Coordinated checkpointing requires **each process to maintain only one checkpoint** on the stable storage, reducing the storage overhead and **eliminating the need for garbage collection.**
- **Disadvantages:**
  - **Large latency** is involved in **committing output**, as a global checkpoint is needed before a message is sent to the OWP.

Coordinated checkpointing

- If perfectly **synchronized clocks were available** at processes, all processes agree at what instants of time they will take checkpoints, and the **clocks at processes trigger the local checkpointing actions at all processes**.
- Perfectly synchronized clocks are not available, either the **sending of messages is blocked for the duration of the protocol, or checkpoint indices are piggybacked to avoid blocking.**
- **Blocking coordinated checkpointing**
- **Non-Blocking coordinated checkpointing**

Coordinated checkpointing

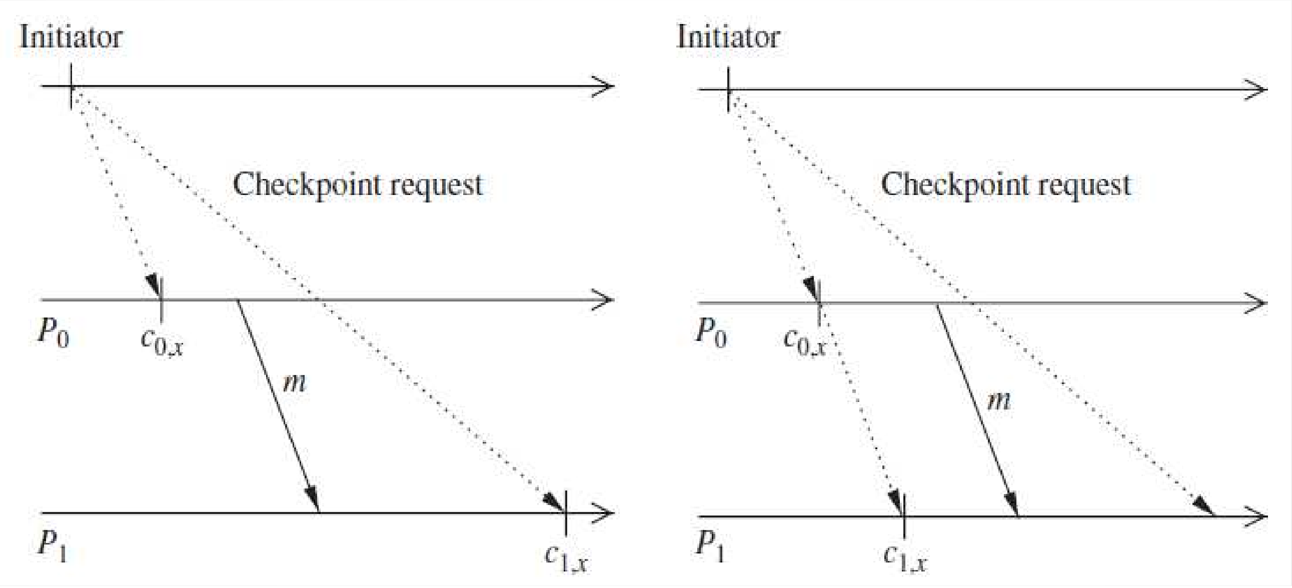
Blocking coordinated checkpointing

- 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.
- A problem with this approach is that the **computation is blocked during the checkpointing**

Coordinated checkpointing

Non-Blocking coordinated checkpointing

- The processes **need not stop their execution while taking checkpoints**.
- **Prevent a process** from **receiving application** messages that could make the checkpoint inconsistent.
- **Message m is sent by $P0$ after receiving a checkpoint** request from the checkpoint coordinator. Assume **m reaches $P1$ before the checkpoint request.**
- This situation results in an inconsistent checkpoint since checkpoint $C1,x$ shows the receipt of message m from $P0$, while checkpoint $C0,x$ does not show m being sent from $P0$. **Forcing each process to take a checkpoint before receiving the first post-checkpoint message.**



Coordinated checkpointing

Non-Blocking coordinated checkpointing

- <u>**Communication Channels are Reliable & FIFO**</u>
- **Chandy and Lamport [1]** in **which *markers play the role of the checkpoint request*** essages.
- The **initiator takes a checkpoint and sends a marker** (a checkpoint request) on **all outgoing** channels.
- Each process takes a checkpoint upon receiving the **first marker and sends the marker on all outgoing channels before sending any application message**.
- <u>**Communication Channels are Non- FIFO**</u>
- The marker can be **piggybacked on every post-checkpoint message**.
- When a process receives an **application message with a marker**, it treats it as if it has received a marker message, followed by the application message.

Coordinated checkpointing

- Coordinated checkpointing **requires all processes to participate in every checkpoint**.
- This **affects scalability.**
- **Reduce the number of processes** involved in a coordinated checkpointing session.
- Only those **processes that have communicated with the checkpoint initiator** either directly or indirectly since the last checkpoint, need to take new checkpoints.
- A two-phase protocol by **Koo and Toueg [2]** achieves minimal checkpoint coordination.

Communication Induced Checkpointing

- ***Communication-induced checkpointing*** *is another way to avoid the domino* effect, while **allowing processes to take some of their checkpoints independently.**
- Processes may be forced to take **additional checkpoints** (over and above their autonomous checkpoints), and thus **process independence is constrained to guarantee the eventual progress of the recovery line**.
- Communication-induced checkpointing **reduces or completely eliminates the useless checkpoints.**
- **Two types of checkpoints**: Autonomous and Forced checkpoints
- **Autonomous checkpoints:** The checkpoints that a process takes **independently are called *local checkpoints.***
- **Forced checkpoints** : A process is **forced** to take a checkpoint are called ***forced checkpoints***

Communication Induced Checkpointing

- Communication-induced checkpointing **piggybacks protocol-related information on each application message**.
- The receiver of each application message uses the piggybacked **information to determine if it has to take a forced checkpoint to advance the global recovery line**.
- The **forced checkpoint** must be taken **before** the application may **process** the **contents of the message**, possibly incurring some latency and overhead.
- No special coordination messages are exchanged unlike coordinated checkpointing.

Communication Induced Checkpointing

- **Two types: Model-based checkpointing and Index-based checkpointing.**
- **Model-Based:** The system maintains **checkpoints** and **communication structures** that prevent the domino effect or achieve some even stronger properties.
- **Index-Based:** The system uses an **indexing scheme** for the **local** and **forced checkpoints**, such that the checkpoints of the **same index at all processes form a consistent state.**

Communication Induced Checkpointing

- **Model-based checkpointing**
- **Model-based checkpointing prevents patterns of communications and checkpoints** that could result in inconsistent states among the existing checkpoints.
- **All information** necessary to execute the protocol is ***piggybacked on application messages.*** The decision to take a forced checkpoint is done locally using the information available.
- The ***MRS (mark, send, and receive) model of Russell [3]*** *avoids the* domino effect by ensuring that **within every checkpoint interval all message receiving events precede all message-sending events**.
- This model can be maintained by taking an **additional checkpoint before every message-receiving event**
- Another way to prevent the domino effect by avoiding rollback propagation completely is by taking a **checkpoint immediately after every message-sending event.**
- Recent work has **focused** on ensuring that every checkpoint can belong to a **consistent global checkpoint** and therefore is not useless.

Communication Induced Checkpointing

- **Index-based checkpointing**
- This assigns monotonically increasing **indexes** to **checkpoints**, such that the **checkpoints having the same index at different processes form a consistent state**.
- Inconsistency between checkpoints of the same index can be avoided in a lazy fashion if **indexes are piggybacked on application messages** to help receivers decide when they should take a forced a checkpoint.
- The protocol by **Briatico *et al [4]***. *forces a process to take a checkpoint upon* ***receiving a message* with a piggybacked index greater than the local index.**

References

1. K. M. Chandy and L. Lamport, Distributed snapshots: determining global states of distributed systems, *ACM Transactions on Computer Systems 3(1), 1985, 63–75.*
2. R. Koo and S. Toueg, Checkpointing and rollback-recovery for distributed systems, *IEEE Transactions on Software Engineering, 13(1) 1987, 23–31.*
3. D. L. Russell, State restoration in systems of communicating processes, *IEEE Transactions of Software Engineering, 6(2), 1980, 183–194.*
4. D. Briatico, A. Ciuffoletti, and L. Simoncini, A distributed domino-effect free recovery algorithm, *Proceedings of the Symposium on Reliability in Distributed Software and Database Systems, Silver Spring, MD, October 1984, 207–215.*

Summary

- Checkpoint –based Recovery
  - Uncoordinated Checkpointing
    - Direct Dependency Tracking Technique.
  - Coordinated Checkpointing
    - Blocking Coordinated Checkpointing
    - Non-Blocking Coordinated Checkpointing
  - Communication-induced Checkpointing
    - Model-based Checkpointing
    - Index-based Checkpointing