CSCI 5105

Instructor: Abhishek Chandra

Today

- Recovery

Recovery

- Operations to be performed to move from an erroneous state to an error-free state
- Backward recovery: Go back to a previous correct state
  - E.g.: packet retransmission
- Forward recovery: Go to a new correct state
  - E.g.: Error-correction codes

Recovery techniques

- Checkpointing
- Message logging
- Rebooting

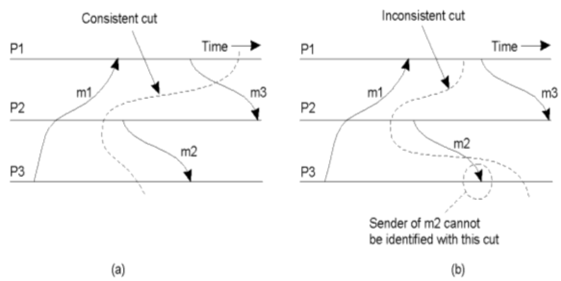
Checkpointing

- Periodically store state on stable storage
  - Mirrored/RAID disks, etc.
- At error-recovery, go back to the last checkpointed state
- Problem: How do we rollback so that all process go back to a consistent global state?

5

Cuts in Global State Space

- Cut: Partition of events representing a global state
  - Set of last recorded event for each process



6

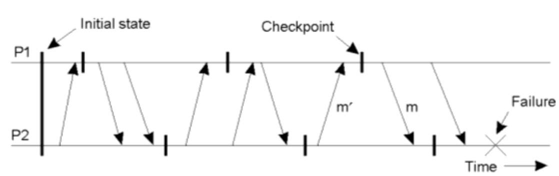
Distributed Snapshot

- Consistent cut:
  - Receipt of a message m in the cut => sending of m also in the cut
  - If event a is in the cut, then all b s.t. b->a are in the cut
- Distributed Snapshot:
  - A consistent global state of the distributed system
- Recovery line: The most recent distributed snapshot

7

Independent Checkpointing

- Each process periodically checkpoints independently of other processes
- Upon a failure, work backwards to locate a consistent cut
- Domino effect: Cascading rollbacks



8

Coordinated Checkpointing

- Processes synchronize before checkpointing locally
- Synchronization techniques:
  - Two-phase protocol
  - Incremental snapshot

9

Two-phase protocol

- One process sends Checkpoint request
- Each recipient checkpoints current state, queues up new local messages
- Send checkpoint-done message

10

Incremental snapshot

- Checkpointing between causally related processes since last checkpointing
- Identify causally related processes incrementally
- Apply two-phase commit between these processes

11

Message Logging

- Checkpointing is expensive
  - Coordination, writing to stable storage
  - Too few checkpoints => can lose lot of state, need lot of recomputation, message passing
- Message logging
  - Take infrequent checkpoints
  - Log messages between checkpoints
- Recovery: Replay messages since last checkpoint

12

Piecewise Deterministic Model

- Execution of each process takes place in a series of intervals
  - Within each interval, the execution is deterministic
  - E.g.: sequence of instructions, message sending
- Start of each interval is a non-deterministic event
  - E.g.: receipt of a message
- Can replay the intervals if we log the non-deterministic events

13

Orphan Process

- Process whose state becomes inconsistent because of another process's crash/recovery
  - Dependent on messages unlogged at crashed process
- Goal: Prevent orphan processes
  - When to log messages?

14

Orphan Process Definition

- Stable message: A message that cannot be lost
- DEP(m): Processes dependent on message m
  - Receivers of m, causally dependent on m
- COPY(m): Processes with non-stable copy of m
- Orphan process: P in DEP(m), no process in COPY(m)

15

Message-Logging Schemes

- Avoid orphan process:
  - no process in COPY(m) => no process in DEP(m)
- Pessimistic logging:
  - Ensures above property at time of message sending
  - At most one dependent process for any non-stable message m
- Optimistic logging:
  - Ensures above property after crash
  - Roll back all orphan processes to a state where they are not in DEP(m)

16

Rebooting

- Localize fault and reboot faulty component
  - Applied to software components
  - Requirement: Modularity, decoupling
- Basic idea: Fault dependent on a rare, transient event
- Recursive rebooting
  - Try shutting down the smallest faulty component first
  - Continue rebooting successively larger componets

17