Consensus and Coordination - A Quick Overview

Ashvin Goel

Electrical and Computer Engineering
University of Toronto

ECE1724

Common Tasks in Distributed Systems

- Coordination: group of processes coordinate their interactions to perform common tasks:
  - Configuration management: save, use configuration values for cluster management, service discovery, failure recovery, etc.
  - Synchronization: locking, barriers
  - Leader election: select leader, let others know about leader
  - Group membership: get list of current members

- Replication
  - Provides fault tolerance (allows handling server/replica failures)
  - Improves latency (clients can access close by replica)
  - Improves performance (clients access different replicas)

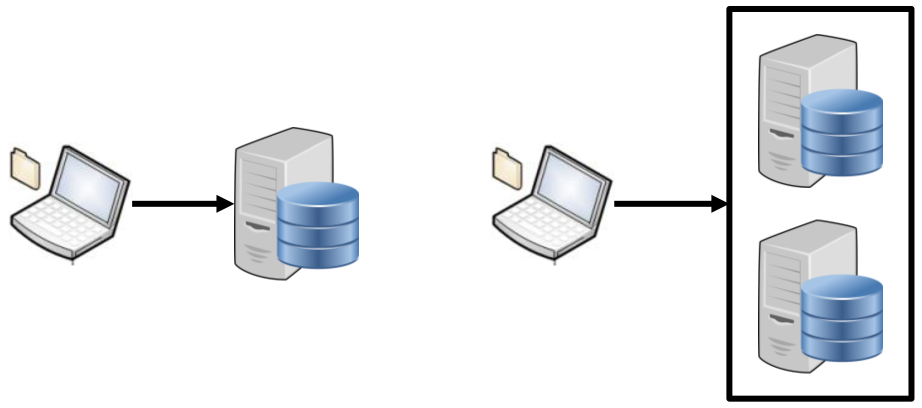
Coordination and Replication

- Failures in distributed systems are common
- Need coordination in the presence of failures
- Need replication to handle failures

- So, large-scale systems often require both coordination and replication services

Requirements for Replication

- Ideally, ensure that clients are unaware of replication, observe a single, highly-fault tolerant machine



Single server, service can fail

Replicated servers, service doesn't fail

- A key requirement is ensuring replica consistency
  - Client reads latest data, immaterial of which replica it accesses

Types of Replication

Definitions (by example):

Operation: set, increment a value in the KV store

State: old or new value

- Two main types
  - Primary-Backup (passive replication: replicate state)
    - One replica is primary, others backup
      - Primary receives and executes operations
      - Replicates updated state to backup (replication layer below storage layer)
    - Typically, failure detection based on timeout
  - State Machine Replication (active replication: replicate ops)
    - Symmetric replicas
      - Any replica receives and replicates operations
      - All replicas execute operations (replication layer above storage layer)
    - Failure detection based on quorum consensus

Primary-Backup (PB): Passive Replication

- Clients send operations to designated replica (primary)
- Primary executes client operations serially
  - Broadcasts any state updates to all backup replicas
  - Backups apply state updates in the same order as primary
  - Backups acknowledge when they are done
- When all backups respond, primary responds to client
  - If primary fails, a backup becomes primary
  - If backup fails, primary responsible for starting another backup
- Requirements:
  - Agreement: There should be only one primary at a time

State Machine Replication (SMR): Active Replication

- Clients send deterministic operations to any replica
  - Replicas may receive concurrent requests
- When a replica receives an operation, it broadcasts that operation to all replicas
- All replicas execute operations in the same order, producing a consistent response for the client
- Requirements:
  - Initial state: All replicas start in the same state
  - Determinism: All replicas receiving the same input on the same state produce the same output and resulting state
  - Agreement: All replicas process inputs in the same sequence

Understanding PB and SMR

- Primary-Backup (PB): think of it as output replication
    - Transferring updated state to backup is simpler to implement since updates only need to be applied idempotently, why?
    - Lower CPU needs since only primary executes operations
    - More network b/w needs when output size > input size
- State Machine Replication (SMR): think of it as input replication
    - Harder to implement correctly since operations need to be deterministic, why?
    - Higher CPU needs since all replicas execute operations
    - Lower network b/w needs when input size < output size

PB and SMR Under Failures

- Primary-Backup:
  - Pros:
    - Requires only f+1 replicas to handle f machine crash failures
  - Cons:
    - Requires a separate view server to detect primary failure
    - Primary failures are visible to client, failure recovery causes delays
    - Failure timeouts need to be conservative to avoid split brain issues
- State Machine Replication:
  - Pros:
    - f machine failures can be masked without performance degradation
    - Does not depend on timeouts for correctness
  - Cons:
    - Requires 2f+1 replicas to handle f machine crash failures

Consensus

- General problem in distributed systems
- A set of processes need to agree on a single data value in the presence of failures, e.g.,
  - PB: one primary, SMR: order of operations
- Non-trivial problem
- Requirements:
  - Agreement: No two correct nodes decide differently
  - Integrity: No node decides twice
  - Validity: Any value decided was proposed by some node

  correctness (safety)

  - Termination: Each correct node eventually decides a value

  progress (liveness)

Consensus

- Key to solution: get permission from majority of participants
- Avoids split brain issues
  - Core problem is perfect failure sensing is not possible
    - E.g., if we use a timeout to detect and remove a faulty leader, it may still believe and serve as a leader
  - Using a majority vote ensures correctness
- Allows handling network failures
- With 2f+1 participants, f failures are possible, with no loss of availability