Consensus in Distributed Systems

Jeff Chase
Duke University

Fischer-Lynch-Patterson (1985)

- No consensus can be guaranteed in an asynchronous communication system in the presence of any failures.
- Intuition: a "failed" process may just be slow, and can rise from the dead at exactly the wrong time.
- Consensus may occur recognizably on occasion, or often.
  - e.g., if no inconveniently delayed messages
- FLP implies that no agreement can be guaranteed in an asynchronous system with byzantine failures either.

Consensus in Practice I

- What do these results mean in an asynchronous world?
  - Unfortunately, the Internet is asynchronous, even if we believe that all faults are eventually repaired.
  - Synchronized clocks and predictable execution times don't change this essential fact.
- Even a single faulty process can prevent consensus.
- The FLP impossibility result extends to:
  - Reliable ordered multicast communication in groups
  - Transaction commit for coordinated atomic updates
  - Consistent replication
- These are practical necessities, so what are we to do?

Consensus in Practice II

- We can use some tricks to apply synchronous algorithms:
  - Fault masking: assume that failed processes always recover, and define a way to reintegrate them into the group.
    - If you haven't heard from a process, just keep waiting...
    - A round terminates when every expected message is received.
  - Failure detectors: construct a failure detector that can determine if a process has failed.
    - A round terminates when every expected message is received, or the failure detector reports that its sender has failed.
  - But: protocols may block in pathological scenarios, and they may misbehave if a failure detector is wrong.

Consistency

Availability

Partition-Resilience

Three Properties You Want
Pick Two
Committing Distributed Transactions

- Transactions may touch data stored at more than one site.
  - Each site commits (i.e., logs) its updates independently.
- Problem: any site may fail while a commit is in progress, but after updates have been logged at another site.
  - An action could “partly commit”, violating atomicity.
  - Basic problem: individual sites cannot unilaterally choose to abort without notifying other sites.
  - “Log locally, commit globally.”

Two-Phase Commit (2PC)

- Solution: all participating sites must agree on whether or not each action has committed.
  - Phase 1. The sites vote on whether or not to commit.
    - procommit: Each site prepares to commit by logging its updates before voting “yes” (and enters prepared phase).
    - Phase 2. Commit if all sites voted to commit.
      - A central transaction coordinator gathers the votes.
      - If any site votes “no”, the transaction is aborted.
      - Else, coordinator writes the commit record to its log.
      - Coordinator notifies participants of the outcome.
- Note: one server => no 2PC is needed, even with multiple clients.

The 2PC Protocol

1. Tx requests commit, by notifying coordinator (C).
   - C must know the list of participating sites.
2. Coordinator C requests each participant (P) to prepare.
3. Participants validate, prepare, and vote.
   - Each P validates the request, logs validated updates locally, and responds to C with its vote to commit or abort.
   - If P votes to commit, Tx is said to be “prepared” at P.
4. Coordinator commits.
   - If all P votes are unanimous to commit, C writes a commit record to its log, and reports “success” for a commit request. Else abort.
5. Coordinator notifies participants.
   - C asynchronously notifies each P of the outcome for Tx.
   - Each P logs outcome locally and releases any resources held for Tx.

Handling Failures in 2PC

1. A participant P fails before preparing.
   - Either P recovers and votes to abort, or C times out and aborts.
2. Each P votes to commit, but C fails before committing.
   - Participants wait until C recovers and notifies them of the decision to abort. The outcome is uncertain until C recovers.
3. P or C fails during phase 2, after the outcome is determined.
   - Carry out the decision by reinitiating the protocol on recovery.
   - Again, if C fails, the outcome is uncertain until C recovers.