L11: Two-Phase Commit

CSE 452 Winter 2016
Transactions: Motivating Example

send_money(user1, user2, amount) {
    Begin_Transaction();
    if (user1.balance - amount >= 0) {
        user1.balance = user1.balance - amount;
        user2.balance = user2.balance + amount;
        Commit_Transaction();
    } else {
        Abort_Transaction();
    }
}
ACID Guarantees

• Atomicity: all parts of the transaction execute or none (user1’s decreases and user2’s balance increases)

• Consistency: the transaction only commits if it preserves invariants (user1’s balance never goes below 0)

• Isolation: the transaction executes as if it executed by itself (even if user3 is accessing user1’s account, that will not interfere with this transaction)

• Durability: the transaction’s effects are not lost after it executes (updates to the balances will remain forever)
What if the transaction is distributed?

• Databases are often partitioned for scalability (user1 and user2 might not share a server)

• A transaction might touch more than one partition

• How do we guarantee that all of the partitions commit the transactions or none?
Two-Phase Commit (2PC)

- An atomic commitment protocol (ACP)
- Guarantees that participants all agree to execute a transaction or none of them will execute the transaction
- There are other ACPs: three-phase commit, etc.
- Think about why you need at least two phases …
2PC Overview

• Participants: nodes that have parts of the transaction to update

• Coordinator: node that will be responsible to running the protocol, can be a participant

• RPCs:
  
  • **Prepare** - the request for votes, responses **Yes** or **No**
  
  • **Commit** - commit the transaction
  
  • **Abort** - abort the transaction
The 2PC Protocol
Protocol Invariants

• All processes that reach a decision, reach the same one.

• A process cannot reverse its decision once it has reached one.

• The commit decision can only be reached if all participants vote Yes.

• If there are no failures and all participants vote Yes, then the transaction will commit.

• If failures are eventually repaired, then every process will eventually reach a decision.
Maintaining invariants with failures
Participant failures: Before sending response?

Coordinator  Participant  Participant

Prepare  Prepare

Yes  Decision?

No  Abort

Abort
Participant failures: After sending vote?
Participant failures: Lost vote?

- **Coordinator**
  - **Prepare**
  - Yes
  - No

- **Participant**
  - **Prepare**
  - Yes
  - Decision?
  - Abort

- **Participant**
  - Log
  - Log
Coordinator failures
Coordinator failures:
Before sending prepare

Coordinator

Participant

Participant

Prepare

Prepare

Yes

Yes

Yes

Commit

Commit
Coordinator failures: After sending prepare

Coordinator | Participant | Participant
---|---|---
**Prepare** | **Prepare** | **Prepare**
**Log** | | |
**Prepare** | **Prepare** | **Prepare**
**Yes** | **Yes** | **Yes**
**Commit** | **Commit** | **Commit**
Coordinator failures:
After receiving votes

Coordinator  Participant  Participant

Prepare  Prepare  Prepare

Yes  Yes  Yes

Prepare  Prepare  Prepare

Yes  Yes  Yes

Prepare  Prepare  Prepare

Yes  Yes  Yes

Commit  Commit  Commit
Coordinator failures: After sending decision
Do we need the coordinator?

Coordinator

Participant

Prepare

Yes

Prepare

Yes

Commit

Decision?

Commit
What if we do not have the coordinator’s decision?

Coordinator

Participant

Participant

Prepare

Prepare

Yes or No?

Yes

Decision?

Commit?
2PC is a *blocking* protocol

- A blocking protocol is one that cannot make progress if some of the participants are unavailable (either down or partitioned).

- They have fault-tolerance but not *availability*.

- Paxos does not have this limitation (but has a variant of it).

- This limitation is fundamental (2 generals problem).