Distributed Systems
601.417
Distributed Transactions

Department of Computer Science
The Johns Hopkins University

Distributed Transactions
Lecture 6
Basic Definition

Transaction - a collection of operations on the physical and abstract application state, with the following properties:

- Atomicity
- Consistency
- Isolation
- Durability

The ACID properties of a transaction
Atomicity

Changes to the state are atomic:

- A jump from the initial state to the result state without any observable intermediate state
- All or nothing (Commit / Abort) semantics
- Changes include:
  - Database changes
  - Messages to the outside world
  - Actions on transducers (testable / untestable)

Consistency

- The transaction is a correct transformation of the state

This means that the transaction is a correct program
Isolation

Even though transactions execute concurrently, it appears to the outside observer as if they execute in some serial order.

Isolation is required to guarantee consistent input, which is needed for a consistent program to provide consistent output.

Durability

- Once a transaction completes successfully (commits), its changes to the state survive failures (what is the failure model?)

- The only way to get rid of what a committed transaction has done is to execute a compensating transaction (which may be impossible sometimes)
A Distributed Transaction

- A distributed transaction is composed of several sub-transactions, each running on a different site.
- Each database manager (DM) can decide to abort (the veto property).
- An Atomic Commitment Protocol (ACP) is run by each of the DMs to ensure that all the subtransactions are consistently committed or aborted.
Atomic Commitment Protocol

A correct ACP guarantees that:

- All the DM that reach a decision, reach the same decision
- Decisions are not reversible
- A Commit decision can only be reached if all the DMs vote to commit (veto property)
- If there are no failures and all the DMs vote to commit, the decision will be Commit
- At any point, if all failures are repaired, and no new failures are introduced, then all the DMs eventually reach a decision

Two Phase Commit

Send “prepare to commit”

- Forced disk write
- Lazy disk write
Two Phase Commit

Forced disk write
Lazy disk write

Return vote (ready or abort)

Coordinator
Participant

Send decision (commit or abort)

Forced write
Lazy write
State Diagram for 2PC

Coordinator
- Init
- Wait
  - All voted “Ready”
  - Timeout or “abort”
- Commit
- Abort

Send “prepare”

Participant
- Init
- Wait
- Abort
- Commit

Ready vote
Commit message
Abort message

Presumed Abort 2PC

When the recovery mechanism has no information about a transaction, it presumes that the transaction has been aborted.

Forced write if commit
Lazy write if abort
Presumed Commit 2PC

When the recovery mechanism has no information about a transaction, it presumes that the transaction has been committed.

Coordinator

participant

Forced write

Forced write if abort
Lazy write if commit

Forced write if abort
Lazy write if commit

(On recovery-needs to talk to all the participants!!!)

Summary So Far

- Basic approach: Two Phase Commit:
  - It works
  - It pays in forced disk writes
  - It is vulnerable to coordinator failure
- Presumed Abort 2PC:
  - Saves forced disk writes by invoking lazy writes on abort
- Presumed Commit 2PC:
  - Saves forced disk writes by invoking lazy writes on commit but pays a price at recovery
Non Blocking ACPs

- An ACP is called **blocking** if the occurrence of some failures forces the DMs to wait until failures are repaired before terminating the transaction.
- When a transaction is blocked at the DM, its locks cannot be released. This may lead to system-wide blocking.
- What can we say about network partitions and blocking?

Every protocol that tolerates network partitions is bound to be blocking.
Quorum Based Protocols

- Every DM has to agree locally
- A majority of the DMs must agree to abort or commit after all the DMs agreed locally
- Simple majority can be generalized to weighted majority
- Majority can be generalized to a quorum
- Instead of one quorum, there can be an abort quorum and a commit quorum

3PC State Diagram (no faults)
3PC Decision Rule for Recovery

Collected States:

- If at least one DM aborted - decide to abort
- If at least one DM committed - decide to commit
- Otherwise if at least one DM in Pre-Commit and a quorum of DMs in (Pre-Commit and Wait) - move to Pre-Commit and send “prepare commit”
- Otherwise if there is a quorum of DMs in (Wait and Pre-Abort) move to Pre-Abort and send “prepare abort”
- Otherwise - Block

3PC Recovery Procedure

- Send state and id
- The new coordinator collects the states from all the connected DMs, it computes its next step according to the decision rule
- Upon receiving a Prepare-Commit/Prepare-abort, each DM sends an OK message
- Upon receiving an OK message from a quorum, the coordinator commits/aborts and sends the decision
3PC Recovery State Diagram

- Simple majority, 3 DMs, smallest-id connected DM is the coordinator

DM1 & DM3 are blocked
Enhanced 3PC Highlights

**E3PC:**

- Uses identical state diagrams as 3PC
- Uses similar communication to 3PC (with different message contents)
- Maintains two additional counters:
  - Last_elected - the index of the last election this DM participated in
  - Last_attempt - the election number in the last attempt this DM made to commit or abort
- Uses a different decision rule and recover procedure

E3PC Decision Rule

**IMAC**: a predicate that is true iff all the connected members with max Last_attempt are in the PC state

Is Max Attempt Committable?

- If at least one DM aborted - decide **abort**
- If at least one DM committed - decide **commit**
- If IMAC and there is a quorum - move to **Prepare-Commit**
- If not IMAC and there is a quorum - move to **Prepare-Abort**
- Otherwise (i.e. no quorum) - Block
E3PC Recovery Procedure

- Elect a coordinator - send state and 2 counters
- Upon getting the Max_elected from the coordinator, set Last_elected = Max_elected + 1
- If the coordinator decision is not to block
  - It sets Last_attempt = Last_elected
  - It moves to the calculated state & multicast a decision
- Upon receiving Prepare-Commit/Prepare-Abort, the DM:
  - Sets Last_attempt = Last_elected
  - Changes state to PC or PA and sends OK
- If a fault happens, restart the recovery procedure, otherwise termination is guaranteed

3EPC Never Blocks a Quorum

- Simple majority, 3 DMs, smallest connected DM is the coordinator

(last elected, last attempt)
Summary

- **Basic approach: Two Phase Commit:**
  - It works
  - It pays in forced disk writes
  - It is vulnerable to coordinator failure

- **Presumed Abort 2PC:**
  - Saves forced disk writes by invoking lazy writes on abort

- **Presumed Commit 2PC:**
  - Saves forced disk writes by invoking lazy writes on commit but pays a price at recovery

Summary (cont.)

- **Basic approach: Two Phase Commit:**
  - It works
  - It pays in forced disk writes
  - It is vulnerable to coordinator failure

- **Three Phase Commit:**
  - It pays even more in messages & forced disk writes
  - Most of the time, it solves the vulnerability problem of 2PC when a quorum exists

- **Enhanced Three Phase Commit:**
  - It costs exactly as 3PC, but with a better logic
  - Always solves the vulnerability problem of 2PC when a quorum exists