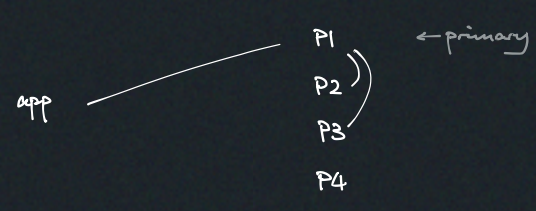
Lec 23 OLTP, Distributed

OLTP : short lived txns
small footprints
low latency

* Coordination



Want : all nodes agree, commits are safe
handle these :
- node failure
- lost message
- message coming late

Assume : - node under control and well-behaved
(no malicious things, etc.)

Replication

Design decisions :

* Replica config
  → Primary - Replica
    └ write to primary only
    └ primary propagate to replica
    └ leader selection if primary down
  → Multi - Primary
    └ update in any replica
    └ sync them
  * K - safety — fault tolerance measure
    └ K = num of replica needed to be online for each DB object

* Propagation Scheme
  → Sync (Strong consistency)
    └ Flushed to replica before commit succeed
      Send change log, other node recover the log
  → Async (Eventual ..)
    └ Return commit after sending out log
    └ (need recovery if other node crash)
* Prop timing
  → Continuous log streaming
    └ Keep sending log
    - Abort need propagation
  → On commit
    └ Send log only at commit
    + Less waste in case of abort
* Active vs passive
  → Active - Passive
    └ One send one receive
  → Active - Active
    └ Run query on all replica, if all agree then return
    └ Ship queries btwn nodes

Atomic Txn Protocol

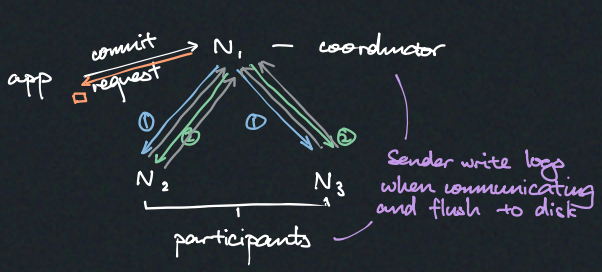
Coordinate txn commit order
└ state machine

▷ Two - Phase Commit (1970)

* Resource Managers (RMs)

* Properties
  - Consistency — all end up in same state
  - Stability — no change after decision
  - Liveness — no hang

* Success case



* Failure : if any participant says abort
coordinator decides abort
coordinator tells everyone abort

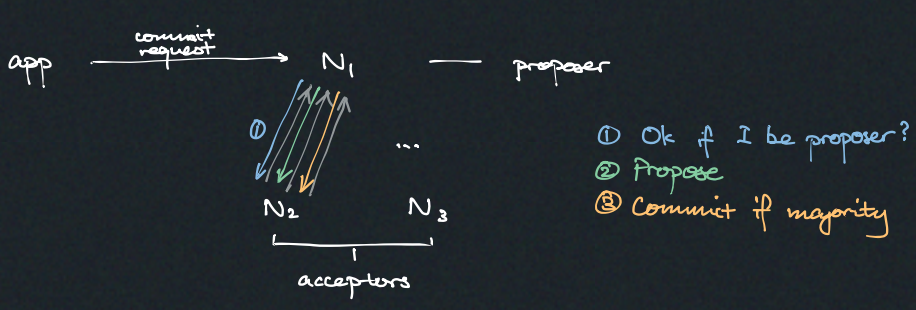
* Recovery :
  - In in prepare state, tell coordinator
  - Not prepared, just say abort
  - If coordinator and is committing, tell participants commit

* Issue :
  - If coordinator down after first phase ... no one else know the phase of that txn — then blocked waiting for coordinator to come back

* Optimizations :
  → If last query, say it's last and ask if prepared
  → Coordinator say commit success early, before phase 2

▷ Paxos (1989)

Problem: keeping replica in sync
Want : very fault tolerant
good if majority of participants up



Proposer race — handled by proposer number
monotonically increasing
→ Slide 28

▷ Multi - Paxos
└ Select leader to lead for a while
If leader down, some protocol

▷ Raft
└ Only node with most up-to-date log becomes leader

Consistency Issues (CAP / PACELC)

Want : - Consistency
- Always available
- Network partition tolerant

CAP Thm : you can only get 2 of those

But for certain data struct, like those CRDT stuff, we can get all 3.

Google Spanner

Global lock : want monotonic increasing counter for all nodes.
→ Use GPS and atomic clock
Each txn gets time range

Use Paxos, 2PL, MVCC
Strict Serializable