EECS 491
Introduction to Distributed Systems

Fall 2019

Harsha V. Madhyastha
Paxos: Fault-tolerant protocol for consensus

- Proposers propose to acceptors in 2 phases
- Can make progress if majority in same partition

Prepare (become leader and discover accepted values)

- Proposer sends unique proposal number $n$
- Acceptor promises to accept proposal if $n > n_p$ and informs proposer of last accepted proposal

Accept (leader gets majority to accept)

- Proposer proposes value with highest $n_a$, or own value
- Acceptor accepts if $n > n_p$
**Paxos: Sample Execution**

\( Pi = \text{Prepare message with proposal no. } i \)

\( Ai-V = \text{Accept message with proposal no. } i \text{ and value } V \)

- Acceptor1:  
  - P1  
  - A1-X  
  - P3  
  - A3-?  

- Acceptor2:  
  - P1  
  - P2  
  - A1-X  
  - A2-Y  
  - A3-?  

- Acceptor3:  
  - P2  
  - A2-Y  
  - P3

**Questions:**

- Which of the accept messages in black succeed?
- What is the value proposed with A3?
Paxos: Sample Execution

$Pi = \text{Prepare with proposal no. } i$

$Ai-V = \text{Accept with proposal no. } i \text{ and value } V$

- Acceptor1: P1 A1-X P3 A3-Y
- Acceptor2: P1 P2 A1-X A2-Y A3-Y
- Acceptor3: P2 A2-Y P3
Paxos: Sample Execution

- Acceptor1: P1  A1-X  P2
- Acceptor2: P1  A1-X  P2  P3
- Acceptor3: P1  A1-X  P3
Paxos

- So far …
  - Among concurrent proposals, how to pick one?

- Needed to implement RSM …
  - How to apply updates in same order at all replicas?
RSM with Paxos

- Log of updates at every replica
  - Replicas execute updates in order in log

- Use Paxos to come to consensus about each slot of the log
RSM with Paxos

Slots in log

Proposals

October 1, 2019
RSM with Paxos

- Banking example:
  - Client submits update to closest database replica

- Whenever an update is submitted:
  - Attempt to get update committed to last slot in log
  - If unsuccessful, retry proposing to higher slot

- Challenge: Must guess slot at end of log
RSM with Paxos

- Any replica may have stale log
  - t1: when an operation is chosen for $i^{th}$ slot in log
  - t2: when $i^{th}$ operation is executed at all replicas

- Arbitrarily large delay between t1 and t2

- Implication:
  - Cannot serve reads based on a replica’s local state
Serving reads

- How to preserve linearizability?
  - Read should reflect all completed writes

- Add reads too to the log
  - In the process, catch up on state of log
  - Execute operations in all prior slots before serving read
Project 2

- Embrace freedom in design space
- Think from first principles
- Move on to part B even if failing hidden test cases for part A
Comparing with P/B Replication
Comparing with P/B Replication

Need two rounds of inter-replica communication

How to skip Prepare phase?
Normal Execution of RSM

- R1 executes Prepare for slot = 1 with n = 2
- R2 executes Prepare for slot = 1 with n = 3
- R1 executes Accept for slot = 1 with n = 2
- R1 executes Prepare for slot = 1 with n = 5
- R1 executes Accept for slot = 1 with n = 5
- R1 executes Prepare for slot = 2 with n = 2
- R1 executes Accept for slot = 2 with n = 2
Incorrect Optimization

- R1 executes Prepare for slot = 1 with n = 2
- R2 executes Prepare for slot = 1 with n = 3
- R1 executes Accept for slot = 1 with n = 2
- R1 executes Prepare for slot = 1 with n = 5
- R1 executes Accept for slot = 1 with n = 5
- R3 executes Prepare for slot = 2 with n = 1
- R3 executes Accept for slot = 2 with n = 1
- **R1 executes Accept for slot = 2 with n = 2**
Safe Optimization

- R1 executes Prepare for slot = 1 with n = 2
- R2 executes Prepare for slot = 1 with n = 3
- R1 executes Accept for slot = 1 with n = 2
- R1 executes Prepare for slot = 1 with n = 5
- R1 executes Accept for slot = 1 with n = 5
- How to enable R1 to safely skip to Accept phase for slot 2?
Reuse leader across slots

- R1 executes Prepare for slot = 1 with n = 2
- R2 executes Prepare for slot = 1 with n = 3
- R1 executes Accept for slot = 1 with n = 2
- R1 executes Prepare for slot = 1 with n = 5
- R1 executes Accept for slot = 1 with n = 5
- R1 executes Accept for slot = 2 with n = 5
- R2 executes Prepare for slot = 2 with n = 6
- R2 executes Accept for slot = 2 with n = 6
Reuse leader across slots

- R1 executes Prepare for slot = 1 with n = 2
- R2 executes Prepare for slot = 1 with n = 3
- R1 executes Accept for slot = 1 with n = 2
- R1 executes Prepare for slot = 1 with n = 5
- R1 executes Accept for slot = 1 with n = 5
- R2 executes Prepare for slot = 2 with n = 6
- R2 executes Accept for slot = 2 with n = 6
- R1 executes Accept for slot = 2 with n = 5
Leader-based Paxos

- Pick one of the acceptors as stable leader
- All clients submit proposals to leader
- Leader can directly skip to Accept phase because no contention

How to pick a leader?
- Paxos!

Drawback compared to no acceptor is leader?
- Leader may be far from client
Leader-based Paxos
Leaderless Paxos
Raft

- Aim: Improve understandability of Paxos

- At all times, one of the acceptors is the leader
  - All replicas forward proposals to leader
  - Order decided by leader is replicated
Optimizing Reads

- How to serve reads with no inter-replica comm.?
- Leader can serve read from local state
  - All writes go through leader
  - Leader aware of which writes completed

- What if leader’s last proposal accepted by majority of Acceptors, but leader not yet aware?
  - Leader’s local state does not reflect this value
- Reads only need reflect writes ACKed by leader
Problems?

Increase in write latency
Replica failures!
How to fix?

Enabling All Replicas to Serve Reads Based on Local State

Lock service

Client

Replica 1

Replica 2

Replica 3
Lease

- Lock with timeout
- If lease holder fails, not a problem because lease will expire

How to pick lease timeout value?
- Short timeout $\rightarrow$ Lease renewal overhead
- Long timeout $\rightarrow$ Unnecessarily block operations
Discrepancy in Lease Validity

- R1 acquires lease
- R1 serves reads based on local state
  
  ... some time later ...

- R2 receives write request
- R2 checks and finds all read leases expired
- R2 executes write without contacting R1
  
  ... some time later ...

- R1 receives read and serves from local state believing lease still valid

How can this happen?
Discrepancy in Lease Validity

- Message that grants lease may have high delay
- Clock at lease holder and lease service may have different skew
- How to account for potential discrepancy?
Next time ...

- Changing the set of replicas in a Paxos RSM
- Applying Paxos in practice with Chubby
Scenario in which lease server and client differ about lease validity?