Replicated state machine and Paxos

Jinyang Li (NYU) and Frans Kaashoek
Fault tolerance => replication

• How to recover a single node from power failure?
  – Wait for reboot
    • Data is durable, but service is unavailable temporarily
  – Use multiple nodes to provide service
    • Another node takes over to provide service
    • How to make sure nodes respond in the same way?
Replicated state machine (RSM)

• RSM is a general replication method
  – Lab 8: apply RSM to lock service

• RSM Rules:
  – All replicas start in the same initial state
  – Every replica apply operations in the same order
  – All operations must be deterministic

• All replicas end up in the same state
• How to maintain a single order in the face of concurrent client requests?
RSM using primary/backup

- Primary/backup: ensure a single order of ops:
  - Primary orders operations
  - Backups execute operations in order
When does primary respond?

- After majority of backups have commit to op
  - Run two-phase commit
  - Lab 8: no persistent state; can avoid messages 2 and 3
Caveats in Hypervisor RSM

- Hypervisor assumes failure detection is perfect
- What if the network between primary/backup fails?
  - Primary is still running
  - Backup becomes a new primary
  - Two primaries at the same time!
- Can timeouts detect failures correctly?
  - Pings from backup to primary are lost
  - Pings from backup to primary are delayed
Paxos: fault tolerant agreement

- Paxos lets all nodes agree on the same value despite node failures, network failures and delays
- Extremely useful:
  - e.g. Nodes agree that X is the primary
  - e.g. Nodes agree that Y is the last operation executed
Paxos: general approach

- One (or more) node decides to be the leader
- Leader proposes a value and solicits acceptance from others
- Leader announces result or try again
Paxos requirement

• Correctness (safety):
  – All nodes agree on the same value
  – The agreed value $X$ has been proposed by some node

• Fault-tolerance:
  – If less than $N/2$ nodes fail, the rest nodes should reach agreement eventually w.h.p
  – Liveness is not guaranteed
Why is agreement hard?

- What if >1 nodes become leaders simultaneously?
- What if there is a network partition?
- What if a leader crashes in the middle of solicitation?
- What if a leader crashes after deciding but before announcing results?
- What if the new leader proposes different values than already decided value?
Paxos setup

- Each node runs as a *proposer*, *acceptor* and *learner*
- Proposer (leader) proposes a value and solicit acceptance from acceptors
- Leader announces the chosen value to learners
Strawman

• Designate a single node X as acceptor (e.g. one with smallest id)
  – Each proposer sends its value to X
  – X decides on one of the values
  – X announces its decision to all learners

• Problem?
  – Failure of the single acceptor halts decision
  – Need multiple acceptors!
Strawman 2: multiple acceptors

- Each proposer (leader) propose to all acceptors
- Each acceptor accepts the first proposal it receives and rejects the rest
- If the leader receives positive replies from a majority of acceptors, it chooses its own value
  - There is at most 1 majority, hence only a single value is chosen
- Leader sends chosen value to all learners

Problem:
- What if multiple leaders propose simultaneously so there is no majority accepting?
Paxos solution

• Proposals are ordered by proposal #
• Each acceptor may accept multiple proposals
  – If a proposal with value $v$ is chosen, all higher proposals have value $v$
Paxos state

• Acceptor maintains across reboots:
  – $n_a$, $v_a$: highest proposal # and its corresponding accepted value
  – $n_p$: highest proposal # seen

• Proposer maintains:
  – $myn$: my proposal # in current Paxos

• Each round of Paxos has an instance #
Proposer

• PROPOSE(v)
  choose my\(_n\) > np
  send PREPARE(my\(_n\)) to all nodes
  if PREPARE_OK(n\(_a\), v\(_a\)) from majority then
    v\(_a\) = v\(_a\) with highest n\(_a\), or choose own v otherwise
    send ACCEPT (n\(_a\), v\(_a\)) to all
  if ACCEPT_OK(n\(_a\)) from majority then
    send DECIDED(v\(_a\)) to all
Acceptor

• PREPARE(n)
  
  If \( n > n_p \)
  
  \( n_p = n \)
  
  reply <PREPARE_OK, n_a, v_a>

• ACCEPT(n, v)
  
  If \( n \geq n_p \)
  
  \( n_a = n \)
  
  \( v_a = v \)
  
  reply with <ACCEPT_OK>

This node will not accept any proposal lower than \( n \)
Paxos operation: 3 phase example

\[
\begin{align*}
\text{nh}=\text{N0}:0 & \quad \text{nh}=\text{N1}:0 & \quad \text{nh}=\text{N2}:0 \\
\text{na} = \text{va} = \text{null} & \quad \text{na} = \text{va} = \text{null} & \quad \text{na} = \text{va} = \text{null}
\end{align*}
\]

- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)
- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)
- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)

- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)
- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)
- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)

- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)
- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)
- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)

- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)
- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)
- \(\text{nh}=\text{N1}:1\), \(\text{na} = \text{null}\), \(\text{va} = \text{null}\)

- \text{Decide}, \text{val}1
- \text{Decide}, \text{val}1
- \text{Decide}, \text{val}1

\[
\begin{align*}
\text{N0} & \quad \text{N1} & \quad \text{N2}
\end{align*}
\]
Paxos properties

- When is the value V chosen?
  1. When leader receives a majority prepare-ok and proposes V
  2. When a majority nodes accept V
  3. When the leader receives a majority accept-ok for value V
Understanding Paxos

• What if more than one leader is active?
• Suppose two leaders use different proposal number, N0:10, N1:11
• Can both leaders see a majority of prepare-ok?
Understanding Paxos

- What if leader fails while sending accept?
- What if a node fails after receiving accept?
  - If it doesn’t restart …
  - If it reboots …
- What if a node fails after sending prepare-ok?
  - If it reboots …
Using Paxos for RSM

- Fault-tolerant RSM requires consistent replica membership
  - Membership: <primary, backups>
  - RSM goes through a series of membership changes
    <vid-0, primary, backups><vid-1, primary, backups> ..
- Use Paxos to agree on the <primary, backups> for a particular vid
  - vid == paxos instance #
Lab 8: Using Paxos for RSM

- All nodes start with static config vid1: N1
- N2 joins
- A majority in vid1: N1 accept vid2: N1, N2
- N3 joins
- A majority in vid2: N1, N2 accept vid3: N1, N2, N3
- N3 fails
- A majority in vid3: N1, N2, N3 accept vid4: N1, N2
Lab7: Using Paxos for RSM

vid1: N1
vid2: N1, N2

Prepare, vid2, N3:1
oldview, vid2=N1,N2

vid1: N1

N3 joins
Lab7: Using Paxos for RSM

vid1: N1
vid2: N1,N2

N1

Prepare, vid3, N3:1

N2

Prepare, vid3, N3:1

N3 joins

vid1: N1
vid2: N1,N2

N3

vid1: N1
vid2: N1,N2
Lab8: re-configurable RSM

• Use RSM to replicate lock_server
• Primary in each view assigns a viewstamp to each client requests
  – Viewstamp is a tuple (vid:seqno)
  – (0:0)(0:1)(0:2)(0:3)(1:0)(1:1)(1:2)(2:0)(2:1)
• All replicas execute client requests in viewstamp order
Lab8: Viewstamp replication

• To execute an op with viewstamp (vs), a replica must have executed all ops < vs
• A newly joined replica need to transfer state to ensure its state reflect executions of all ops < vs
Lab 8: Using Paxos for RSM

- Primary in new view is last primary, if alive
- Otherwise, backup lowest ID
- Resume responding to client after backups and primary are in sync