Day 14: Replication revisited

Raft Guarantee
Passive

- Code can be non-deterministic

Active

- All RM's are equal
- Required code to be deterministic

Key issues:
- Lots of messages & high overhead

Wide area network (Internet)
- Network latency => 40 - 100ms
- Geo-distributed database / storage
- Will use active replication

Can you tolerate the latency overhead of leader failures / leader election?

Passive => Data Centers / Clusters
- Network latency => 4ms

Key issues:
- Under failure it takes time to find new leaders & commit
Consensus

Consensus = all nodes in the algorithm agree on some value

Chandy-Lamport: agreement on what is in a snapshot

Examples of consensus protocols:
- Chandy-Lamport: agree on snapshot
- Passive replication: agree on leader state (in log)
- Active replication: agree on ordering of events
Raft (Passive Replication)

"Raft cluster" $\Rightarrow N$ nodes that participate in protocol
\hspace{1cm} $\Rightarrow$ 1 leader
\hspace{1.5cm} $\Rightarrow N-1$ followers

"Node" $\Rightarrow \log +$ state machine + raft protocol code
\hspace{1cm} $\Rightarrow$ list of client requests
Raft Election Properties

1. Liveness \(\Rightarrow\) eventually you will have a leader
   \(\Rightarrow\) random timeouts

2. Safety \(\Rightarrow\) at most one leader
   \(\Rightarrow\) each node only votes once
   \(\Rightarrow\) only the candidate with majority votes wins

When there are partitions \(\Rightarrow\)

n/w partition

these nodes can communicate with each other but not across partitions
How to ensure that leader election terminates (liveness)

Goal: avoid a split election

No leader is elected because no node has enough votes

3 node Raft cluster
Leader requires: 2 votes

If each node votes for itself, no one wins. This can happen if multiple nodes timeout at the same time.

Add randomness to avoid multiple nodes becoming candidates at same time!!
Each node randomly set heartbeat timeout

\[ \text{Timeout} = \text{Constant Timeout} + \text{Random Value} \]
Raft Log Safety

- The log is an array
- Each element in the log includes:
  1. The term of the leader which inserted the entry
  2. The index of the array
  3. The client request

On item is committed when the leader responds to a client.

Diagram:

- FE
- Term 1 → Term 2 → Term 3
- 1
- Regs
- Node 1
- Node 2
- Node 3
- Leader
- Statement
- Done
- Followed
- 1
- 3
- 4
- 5
Log Safety

if a leader claims that an entry is committed then all
future leaders should agree that it’s committed

* leader election \[\Rightarrow\text{both}\] ensure
* log commit \[\Rightarrow\text{log safety}\]
log commit

as new leader, your job is to

1. repair logs of followers

2. commit a "noop" event to a majority of nodes before responding to clients

help repair
log & bring this follower up to date
Leader Election

To ensure that the node with the most complete log wins, each follower only votes for a candidate with an equivalent or more complete log.

Based on comparison of index/term of the last element in candidate's log & follower's log.

Old leader

4343
1 2 3 4
now failed

Request vote

Candidate

343
1 2

This node can never become a leader.
FE

provides at most once semantic

leaders

cache for duplicate suppression & response replay
Size of a Raft Cluster ($N$)

Size is based on expected number of maximum failures ($F$)

$$N = 2F + 1$$

If $F$ nodes fail then $F + 1$ nodes are available $\Rightarrow$ majority of nodes are available