Raft consensus algorithm

Brno Distributed Systems Meetup Group

Agenda

- Raft algorithm
- Naive TLA+ spec
- Published TLA+ spec

Consensus in distributed systems

- Agreement - all ***correct*** processes decide on the **same value**
- Validity - value proposed by on of the processes
- Termination - all correct processes eventually reach the decision

- Impossible to guarantee in fully asynchronous system
  - FLP Impossibility paper by Fischer, Lynch, Paterson

Consensus algorithms

- Viewstamped replication (1988)
- Paxos and it’s variants (1989)
- ZAB - Zookeeper Atomic Broadcast (2008-2011?)
- Raft - reaction to difficulties with Paxos (2013)

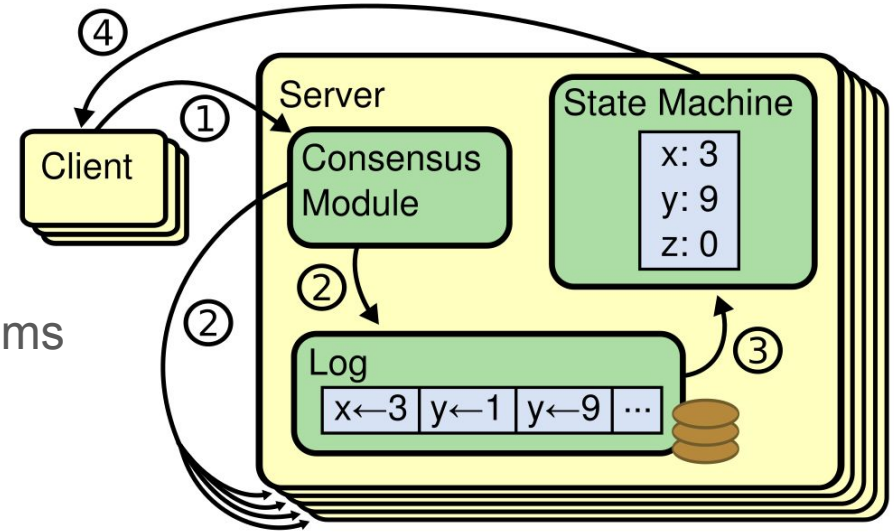
Raft

- Goals
  - more understandable
  - suitable for practical systems
  - result equivalent to (multi) Paxos

- **R**e{liable|plicated|dundant} **A**nd **F**ault-**T**olerant

  As a plus, we were using the randomly generated name Cheesomi in the paper before we came up with the name Raft in September 2012. The name appeared just over 100 times in our paper submission back then, so switching to the shorter name actually helped shrink the paper down quite a bit.
  https://groups.google.com/forum/#!msg/raft-dev/95rZqptGpmU/cfH4N7reBQAJ

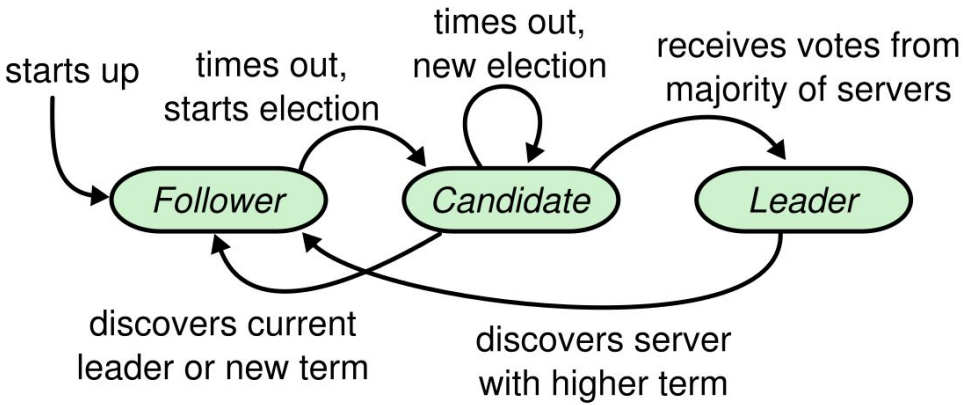
Raft



- Replicated state machine
- Algorithm decomposed into subproblems
  - Leader election
  - Log replication
  - Safety
- Either RPC (paper) or asynchronous messages (TLA+ spec)
  - RequestVote RPC, AppendEntries RPC
  - RequestVoteRequest, RequestVoteResponse, AppendEntriesRequest, AppendEntriesResponse
- 3 possible states
  - Follower, Candidate and Leader
- Time divided into **terms**, at most one leader per term

Leader election

- All servers start as followers
- Randomized election timeout starts an election
  - reset by heartbeats - AppendEntries
- Election
  - increment current term
  - vote for itself
  - reset election timer
  - send RequestVote to all other nodes
- Result
  - wins election
  - other node wins election
  - no node wins election



Log replication

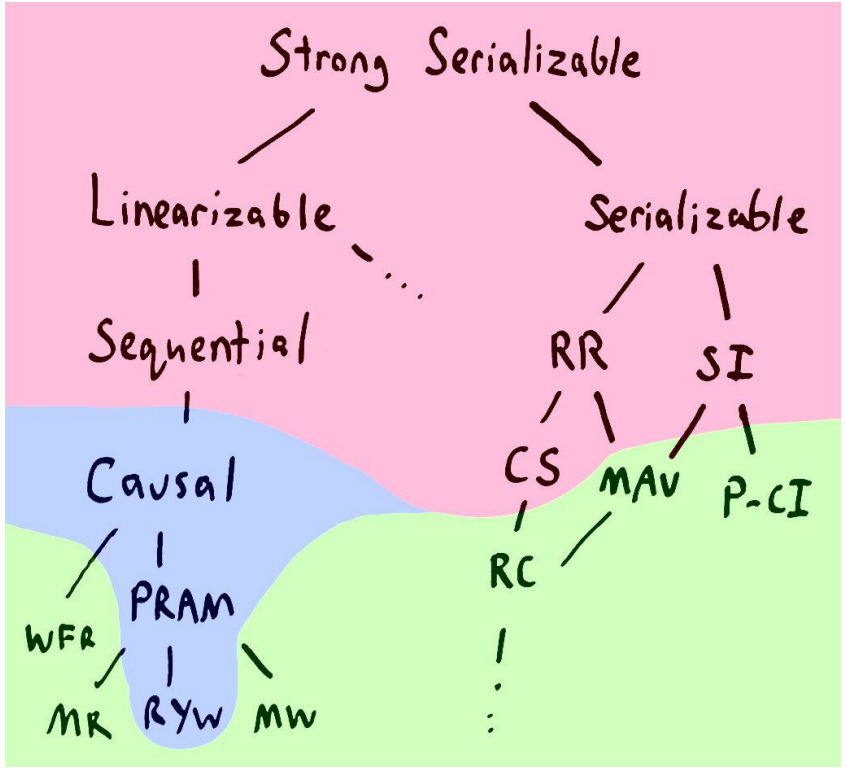
- Leader receives command from a client
- Leader appends to its log
- Leader issues AppendEntries
  - current term, entries, leaderId, prevLogIndex, prevLogTerm, leaderCommit
- Follower adds to its log (but does not commit)
- When leader receives confirmation from a majority it commits its log and replies to the client

Safety

- Which servers can become leaders
  - only server with highest position in the log and newest term
- What entries can be safely committed and when
  - For current term
    - Leader commits when receives majority of the votes
    - Follower when it receives commitIndex from the leader
  - For previous term
    - Leader commits entries from previous terms only when a there is a successful commit from current term

Client interaction

- Client connects to random server, followers redirect to leader
- Client adds serial number to the request
- If leader already processed the request it responds immediatelly


Strong Serializable
Linearizable
Serializable
Sequential
RR
SI
Causal
CS
MAV
P-CI
RC
PRAM
WFR
MR
RYW
MW

Implementations

- LogCabin - reference Raft implementation
- Kafka (WIP) - KIP-500: Replace ZooKeeper with a Self-Managed Metadata Quorum
  - https://cwiki.apache.org/confluence/display/KAFKA/KIP-500
- Neo4j
- Hazelcast

Aeron Cluster

- Efficient reliable UDP unicast, UDP multicast, and IPC message transport.
- Adaptation of the Raft algorithm