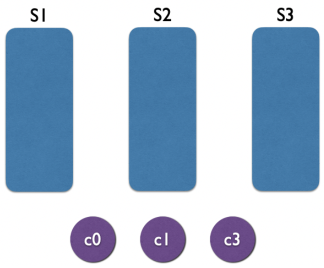
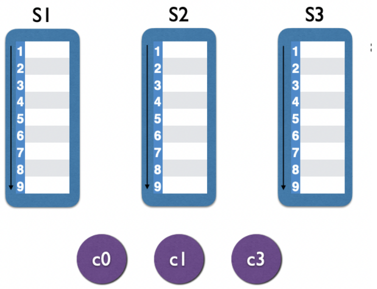
CAS CS 350
Lec 11

Raft

1. Two general problems
    a. Two generals need to coordinate an attack
    b. Must agree on the time to attack
    c. They win only if they attack simultaneously
    d. Communicate through messengers
    e. Messengers may be killed on their way
2. Solve the two generals problem
    a. G1 sends time of attack to G2
    b. Problem: how to ensure G2 received message?
        i. Solution: let G2 ack receipt of msg
    c. Problem: how to ensure G1 received ack
        i. Solution: let G1 ack the receipt of the ack
    d. This problem is impossible to solve
3. Two general problem
    a. Applicability to distributed systems
    b. Two nodes need to agree on a value
    c. Communicate by messages using an unreliable channel
    d. Consensus is a core problem
4. Raft
    a. Fault tolerance through replicated state machines (RSM)
        i. Fault tolerance
            1. Execution (reassign the task to other workers) → in MapReduce
            2. Primary backup
            3. Etc.
    b. RAFT: much more complete design than the PAXOS paper
    c. Fundamentally, Raft bakes in notion of a leader and log
5. Big picture
    a. Have bunch of clients and servers
    b. Servers are accumulating logs of client commands
    c. 

d. Goal is to keep the logs in the servers identical (when given failures → server and network)

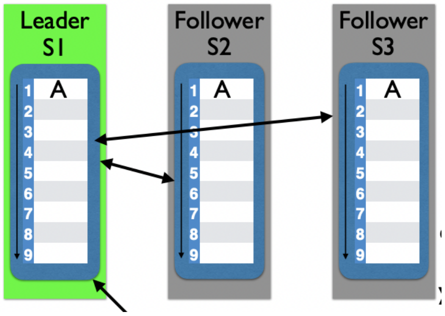
e.



f. At any given time, there is at most one leader (from the server) → example Server1

g. Other server (server2 and server3) become followers here

h.



i. Clients are supposed to send requests to the Leader

j. Client0 sends request A to leader (server1)

k. Leader updates the command to its log when it receives request A from client

l. Sends concurrent request to followers (server2 and server3)

m. Once the leader believes majority of the followers (including itself) have command in there log it announces commitment and that command can be executed safely → send back result to the client

6. What matters

a. NOT: is it more or less understandable that PAXOS

b. NOT: is it better, faster, more efficient

c. RATHER: it is a relatively complete tutorial on how to build a RSM system

7. PAXOS - lots of messages

a. Raft can replicate whole sequence of commands while PAXOS is not (reduces the communication between the protocols greatly)

b. Whole bunch of independent PAXOS instances - too slow & requires lots of messages → in practice optimize so that common case does not need this

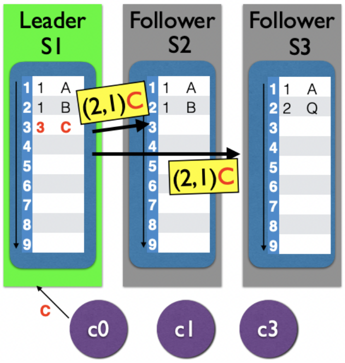
8. What are we hoping for?

a. Tolerate failure of a minority of followers (if majority fails, RAFT cannot do anything about it)

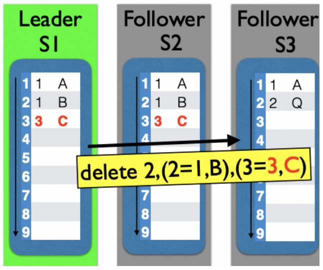
b. Convergence to one version of the log

    c. No undos of logs
    d. Only execute client requests (and thus reply to client) when it is “committed” – can’t go back and unexecuted or un-tell client a response

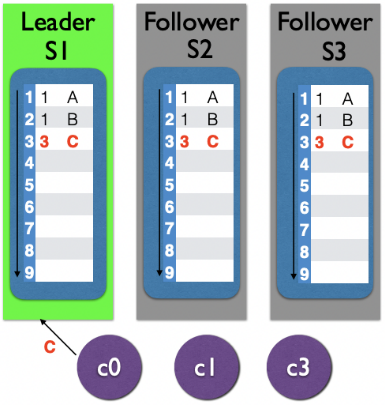
9. No Leader failure
    a. But: Followers’ crash, Network loses messages
    b.



    c. LOGS same by force:
        i. Leader
            1. Appends to its log
            2. In parallel Sends index (sequence number that is command in the log → 1,2,3,4,etc in log) and “term” (sequence number that indicates a phase in the RAFT protocol → each time there is a new election, the term increases) of prior entry as part of AppendEntries RPC (concurrent requests)
        ii. Followers
            1. If prior index and term not in log REJECT if match then log is consistent
    d. Here, S3 will not append the command while S2 will append the command
    e. LOGS same by force: If a follower is found inconsistent then leader rolls back follower and then rolls it forward
    f.

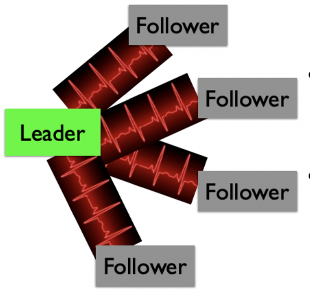


    g. Leader does not care what is in your log; it forces its view on all followers! (when leader sends the current command, it also sends the previous index and term number and the previous command → if the command is not the same, the follower replies back to the leader that the numbers/commands do not match → leader sends the previous command/numbers to that follower so that the follower can have the same command/number)

h. 

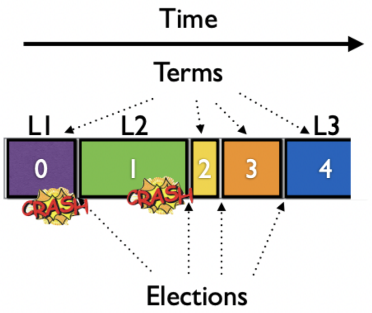
   i. When follower rejects the command, it sends its previous command/numbers to the leader → leader jumps to that command/numbers

10. 2 common divergence
    a. Follower crashed for a while (disconnected)
       i. The leader simply rolls follower forward
    b. A prior leader crashed while sending out some AppendEntries messages
       i. Then leader both rolls follower back and forward
    c. Consistency through leader driven reconciliation
11. Summary
    a. Tolerate failure of a minority of followers
       i. Various times states “wait for majority”, “when majority” (majority matters → if majority fails, then we are done)
    b. Convergence on one version of the log
       i. By reconciliation to leader’s log
    c. Only execute “committed” requests
       i. Not a problem if leader does not crash and waits for a majority to respond that append is durable - then sends out a commit point (piggy backed or AppendEntries RPC)
12. HeartBeat to followers

    a. 

    b. appendEntries done repeatedly by leader to all followers (null if necessary)
    c. Followers initiate an “ELection” if heartbeat timeout expires
    d. The purpose is for the followers to acknowledge that there is a leader
    e. Each server has a timeout → followers immediately initiate a new leader election

13. Time broken into Terms



    a.
    b. Detect that a leader has “crashed”
    c. Terms are distinct phases of the protocol
    d. Start an election to initiate a new term with a new leader (term cannot go backwards)
    e. New leader has to pick up the pieces (when there is no leader, in the outside world, the system appears dead → cannot take any request)
    f. Remember a leader can crash in the middle of anything

14. Leader election
    a. Fewer than 2 leaders any time (0 and 1 is ok)
    b. Ideally greater than 0 (so the most ideal is one leader)
    c. Raft has separate mechanisms for these

15. Fewer than 2 (at most one) leader elected per term
    a. When a server gets a vote request it is allowed to vote yes or no BUT it is only allowed One vote in any given term - it can only say yes to one candidate
    b. Candidate must collect yes votes from a majority to become the leader
    c. Therefore, only one candidate become the leader for a term

16. Reason to use the majority
    a. Only requiring majority (not everyone) means that you can tolerate failure of a minority!
    b. Avoids split brain
    c. Any two majorities must overlap with one server - prior majority must share a server with current majority - ensures continuity
    d. New leader will be sure to know about all previous decisions (committed entries)