RAFT: Consensus Algorithm

Team Members

- Vedansh Mittal (2019101054)
- Shashwat Goel (2019111006)
- Nikhil Chandak (2019111040)
- Bhaskar Joshi (2019111002)

Table of Contents

How to run

```
cd src/raft
go test
```

Code Explanation

See Accompanying PDF for code explanation.

Test cases explanation

1. TestInitialElection

This creates a cluster and checks if the cluster elects any leader for the first term. It does not crashes any server and hence checks that there should not be any change in leader and term in such case.

2. TestReElection

This creates a cluster, waits for election, crashes the leader and then expects new leader to come up. It then crashes majority of nodes and then confirms that no leader is there due to no majority. It also confirm rejoining of old leader does not cause change in term or leader.

3. TestManyElections

Disconnects many servers just leaving majority alive and checks after a timeout that a leader is there or not. Repeats this process many times to robustly check leader election.

4. TestBasicAgree

Checks basic complete agreement of logs among the cluster.

5. TestRPCBytes

Checks that each command is sent to each server only once. No duplication.

6. TestFailAgree

Sends some logs, disconnects a server and then send some more logs. After some time the disconnected server gets re-elected and then sends some more commands. Eventually all servers including disconnected ones should have identical logs.

7. TestFailNoAgree

Checks that is majority of followers are disconnected then leader can not commit any log. Also when the majority is repaired then (they may elect new leader is they hold election in case heartbeats take time to reach time) and then if a new entry is logged then it can be easily committed by then leader.

8. TestConcurrentStarts

Sends many concurrent requests to the leader and waits for some time fr cluster to settle up and then checks the agreement of logs.

9. TestRejoin

Disconnects leader and then send it some commands, meanwhile send other commands to the new leader elected and then disconnect that new leader too. rejoin old leader and and again send the then leader new commands to agree. Finally join all nodes and then test that finally cluster is in an agreement on log entries.

10. TestBackup

Aim for this test is to make leader to correct large number of incorrect logs in followers. Firstly a majority of followers are disconnected and then many commands are sent to commit. Checks that they won’t commit and then the active nodes are also crashed and the followers crashed earlier should be made active. After a series of commands send the cluster which would be successfully committed, again some followers are disconnected leaving minority in active state. Eventually when all nodes are made alive many nodes will have very long lists of uncommitted logs and that too inconsistent with each other. Eventually all nodes would agree on same log entries.

There are some other tests which simulate which uses two different data structs simulating volatile and persistent memories in the server and checks that crash of a server (which erases volatile memory) does not cause the server to malfunction when repaired.