CLOUD COMPUTING - UE20CS351

Project: Implementing Raft Logic in Go

Team:

1. PES1UG20CS565 – Mrunalini Thamankar D
2. PES1UG20CS567 – N V Bharath Ithal
3. PES1UG20CS606 – Thejas N U
4. PES1UG20CS609 – Veeresh R G

Introduction:

The goal of this project was to implement the Raft consensus algorithm in the Go programming language. Raft is a distributed consensus algorithm used to achieve consensus among a group of nodes in a distributed system. It ensures that all nodes agree on a single value or state even in the presence of failures. The Raft algorithm is widely used in distributed databases, distributed file systems, and other distributed systems to ensure consistency and reliability.

Scope:

- Leader Election: The ability to elect a leader among a group of nodes in a distributed system.
- Log Replication: The ability to replicate logs across all nodes in the system, ensuring consistency.
- Commitment of Logs: The ability to commit logs once they are replicated to a majority of nodes, ensuring durability.
- Membership Changes: The ability to handle changes in the membership of the distributed system, such as adding or removing nodes dynamically.

Implementation:

1. Implement the becomeFollower function entirely.
2. Implement the logic for a follower node to handle a received RequestVote function from a candidate.
3. Implement the logic for a candidate to handle a reply to the above RequestVote it sent out to its peers, be it successfully or unsuccessfully.
4. Implement the logic for the leader to commit its log successfully in the event of majority confirmation, or fail in case majority confirmation is not received.

Testing:

We were given 2 test cases for this project:

Test case 1: The leader election is not successful, hence the consensus algorithm fails.

Test case 2: First a leader is elected and then the leader fails after committing 2 logs. Then a new leader is elected which commits many logs. The original leader reconnects and then becomes a follower and updates its logs.

Results:

The implementation successfully replicated the Raft algorithm and provided reliable consensus and fault tolerance in a distributed system. The implementation was able to recover from node failure and leader failure without losing data or compromising safety.