Cloud Computing - Mini Project Report
Implementing Raft Logic in Go
April 2023

Submitted By:
Ayushi Soumya - PES1UG20CS097
Bhumika Nayak - PES1UG20CS104
Deepthi Dayanand - PES1UG20CS120
Vaishnavi V B - PES1UG20CS902

VI Semester Section B
PES University

Short Description and Scope of the Project:
Raft is a consensus algorithm for managing a replicated log in a distributed system. The Raft algorithm is divided into three main components: leader election, log replication, and safety. In this project we have implemented leader election and log replication in Go. The scope of the project includes understanding the Raft algorithm and its various components, such as leader election, log replication, and commit index, as well as implementing these components in Go. The project will require a good understanding of concurrent programming, network programming, and Go programming concepts. The project will also require testing the implementation for various failure scenarios, such as node failures, network partitions, and leader failures, to ensure that the system is fault-tolerant and resilient. The goal of the project is to create a reliable and fault-tolerant distributed system that can handle failures and network partitions while ensuring consistency and availability of data.

Methodology:
1. Understanding the Raft Consensus Algorithm: studying the Raft algorithm and its components. Steps involved in Leader election and log replication from “In Search of an Understandable Consensus Algorithm (Extended Version)” by, Diego Ongaro and John Ousterhout, Stanford University”
2. Understanding the Go programming language: Learning the syntax of functions, loops, if statements. Also understanding the importance of go routine and how to use it.
3. Understanding the Code Provided: Understanding structures, variables, and functions given in the code provided. Also, identifying the importance of the parts of code to be filled by us.

4. Implementing our part of code in go lang.
5. Testing the System with the two test cases provided to us.
6. Debugging the code and making it more efficient

Testing:
When implementing the Raft consensus algorithm in Go, testing is critical to ensure that the system works correctly and reliably under different scenarios and failure conditions.
We were given 2 test cases for this project:
Test case 1: The leader election is not successful hence the consensus algorithm fails.
Teat case 2: First a leader is elected and then the leader fails after committing 2 logs. Then a new leader is elected. Which then commits many logs. Then the original leader reconnects and then becomes a follower and updates its logs.

Results and Conclusions:
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.
The project demonstrated the value and importance of using a consensus algorithm like Raft in building reliable and resilient distributed systems.