Distributed Key-Value Store with Multi-Paxos

Natanel Barazani
ID: 319032447

February 1, 2025

Abstract

This report details the implementation of a distributed key-value store using Multi-Paxos for consensus, built as part of the Distributed Systems course project. The system ensures consistency, fault tolerance, and high availability. The project involves leader election, log replication, and snapshot-based log compaction to maintain efficiency over time. This document covers the architecture, design choices, challenges faced, and testing results.

1 Introduction

The goal of this project is to implement a highly available, fault-tolerant key-value store using Multi-Paxos for achieving consensus across multiple distributed nodes. The system follows a state machine replication approach, ensuring linearizable consistency for updates and sequential consistency for reads.

2 System Architecture

The system consists of multiple server nodes running a Paxos-based consensus algorithm, an etcd-based membership management component, and a client API for interaction with the key-value store. The architecture includes:

- **Leader Election**: Nodes coordinate through etcd to elect a leader dynamically.
- **Multi-Paxos for Consensus**: The leader drives the decision-making process for log replication.
- **Replicated Log**: Ensures all nodes apply the same sequence of updates.
- **Log Compaction**: Periodic snapshotting in etcd for memory efficiency.
- **REST API**: Exposes HTTP endpoints for key-value operations.

3 Implementation Details

3.1 Leader Election

Nodes use etcd for leader election, and non-leader nodes forward requests to the leader.

3.2 Multi-Paxos Consensus

The consensus module maintains a log of decisions, applying updates in a strict order to maintain consistency.

3.3 State Machine Replication

The key-value store is modeled as a replicated state machine, applying updates from consensus decisions.

3.4 Snapshotting and Log Compaction

To reduce memory footprint, periodic snapshots of the state machine are stored in etcd, allowing nodes to discard old logs beyond the snapshot point.

4 Testing and Validation

A test suite was developed to validate:

- Leader election and failover.
- Correctness of key-value operations.
- Replication across nodes.
- Snapshot and recovery mechanism.

Automated scripts ensure correct behavior even under failure conditions.

5 Challenges and Solutions

- **Leader Forwarding Issues**: Some nodes were incorrectly forwarding requests without checking actual leader status. Solution: Validate the leader dynamically before forwarding.
- **Snapshotting Errors**: Initially, some nodes failed to retrieve snapshots on restart. Solution: Implemented etcd-based persistent storage for snapshots.

6 Conclusion

This project successfully implemented a fault-tolerant key-value store using Multi-Paxos. The system ensures consensus, replication, and efficiency through log compaction and snapshotting. Future improvements include optimizing leader election latency and implementing dynamic membership changes.