Raft Consensus Algorithm Implementation

Introduction

This repository contains an implementation of the Raft consensus algorithm in Python. Raft is a consensus algorithm designed for managing a replicated log in a distributed system. It provides strong consistency guarantees and fault tolerance, making it suitable for use in distributed databases, key-value stores, and other distributed systems.

Files

raft.py raft.py contains the implementation of the Raft consensus algorithm. The Raft class provides functionality for managing a Raft cluster, including leader election, log replication, and handling client requests. It also includes helper functions for sending and receiving messages between nodes in the cluster.

hashtable.py

hashtable.py contains a simple implementation of a hash table, which is used by the Raft nodes to store key-value pairs. The HashTable class provides methods for setting and getting values in the hash table, with support for concurrent access using locks.

consistent_hashing.py

consistent_hashing.py contains an implementation of consistent hashing, which is used by the Raft nodes to distribute keys evenly across the nodes in the cluster. The ConsistentHashing class provides methods for adding nodes to the hash ring and for determining which node a given key should be assigned to.

Getting Started

To use the Raft implementation, follow these steps:

Clone the repository to your local machine. Run the Raft cluster by executing python raft.py <ip_address> , where <ip_address> is the IP address of the node, is the port number to listen on, and is a string representing the partition configuration of the cluster. Repeat step 3 for each node in the cluster, ensuring that each node has a unique <ip_address> and . example for command:

```
python3 "raft.py" "127.0.0.1" "5001" "[['127.0.0.1:5001',
'127.0.0.1:5002', '127.0.0.1:5003', '127.0.0.1:5004', '127.0.0.1:5005']]"
0
```

Usage

Once the Raft cluster is running, clients can connect to any node in the cluster to perform operations on the distributed hash table. Clients can send SET and GET commands to set and retrieve values in the hash

table, respectively.