Design Document: ZooKeeper-Based Fault-Tolerant Replicated Database Server

1 Overview

The system ensures that all replicas process client operations in the same global order, yielding consistent state across servers without direct inter-replica communication.

Each client request is transformed into a sequential znode under the ZooKeeper path `/requests`. Because ZooKeeper guarantees monotonically increasing sequence numbers for `PERSISTENT_SEQUENTIAL` znodes, sorting the znodes by name yields a total order shared by all replicas. Every server watches the `/requests` parent znode and applies operations in sorted order using a single-threaded executor.

2 System Components

MyDBFaultTolerantServerZK

This class extends an existing single-server database implementation and adds fault tolerance through ZooKeeper. It encapsulates - Cassandra connection management (cluster + session), ZooKeeper connection and watch management, A total-ordering mechanism using sequential znodes, A single-thread executor that applies CQL statements deterministically.

ZooKeeper

ZooKeeper is used as a replicated metadata service. It provides A globally ordered sequence of client operations, Durable storage of operations (znodes under `/requests`), Watch notifications that inform replicas of newly added requests. Only one ZooKeeper root, `/requests`, is needed. All operations are created with prefix `/requests/op-`.

Cassandra

Each replica hosts a Cassandra session connected to the same keyspace. Replicas independently apply CQL operations but receive them in the same total order. No replica-to-replica messaging is required.

3 Operation Flow

Client Request Handling

When a client sends a command to the server:

1. The server receives raw bytes through the pre-existing NIO messaging layer.
2. The command string is written to ZooKeeper as a `PERSISTENT_SEQUENTIAL` znode under the prefix `op-`.
3. ZooKeeper assigns a monotonically increasing integer suffix, producing an ordered identifier such as `op-0000002314`.

No local execution is performed at the moment of receipt; the server waits until its watch handler processes the new znode.

Replica Ordering and Execution

Each replica watches the children of `/requests`. Whenever a `NodeChildrenChanged` event occurs:

1. The server retrieves the full list of znodes under `/requests`.
2. The list is sorted lexicographically, which corresponds to the sequence numbers.
3. For each znode not previously applied, the server:
   (a) Fetches the znode's data (the SQL/CQL command).
   (b) Marks the znode as applied to avoid duplicates.
   (c) Submits the command to a single-thread executor.
4. The executor applies each command to Cassandra using the same deterministic order at every replica.

This architecture guarantees linearizable state transitions across replicas as long as all replicas observe the same sequence of znodes.

4 Recovery and Fault Tolerance

ZooKeeper Guarantees

ZooKeeper provides durability for znodes and ensures that sequence numbers are persistent. After a server crash or restart, the server reconnects to ZooKeeper, It lists all existing znodes under `/requests`, It replays all operations not yet present in its local `appliedZnodes` set. Thus recovery is deterministic and requires no coordination with other replicas.

Replica Independence

Replicas do not communicate directly. They rely solely on ZooKeeper's ordering mechanism and Cassandra's ability to accept idempotent operations. A replica can fall behind, crash, restart, or temporarily disconnect from ZooKeeper without harming global correctness.

5 Concurrency and Consistency

Single-Threaded Executor

Each replica applies commands to Cassandra using a single-thread executor to ensure replica local ordering exactly matches ZooKeeper ordering, No race conditions during command execution, No need for local locking or transaction coordination.

Avoiding Duplicate Execution

The system keeps a concurrent set `appliedZnodes` containing names of znodes already scheduled for execution. Since watches may trigger multiple times and the system may re-list znodes, this structure ensures that no command is executed twice.

6 Design

The design follows a minimalist approach:

- ZooKeeper acts as the sole coordination mechanism.
- No leader election or quorum protocol is implemented in the server.
- Sequential znodes provide total order without extra logic.
- The design isolates failure domains: Cassandra stores data; ZooKeeper orders operations.