design doc - raft

Shri Venkatakrishnan Vasudevan and Abhijit Chunduru

December 2025

1 Design and Architecture

- The system follows the **replicated state machine (RSM)** model, using Apache Ratis to provide Raft-based consensus and ensure a single global order of database operations.
- Client SQL commands are encapsulated with a unique operation identifier and submitted as Raft log entries; commands are executed only after they are committed, guaranteeing strong consistency and fault tolerance.
- A custom Raft state machine applies log entries sequentially and enforces idempotency by tracking completed log indices, preventing duplicate execution after crashes or leader changes.
- Persistent state is stored in Apache Cassandra, with each server maintaining an isolated keyspace for application data and Raft metadata such as the last applied log index.
- To limit log growth and speed up recovery, the system periodically creates binary checkpoints of the database state using a custom snapshot format stored on the local filesystem.
- On recovery, the server restores the latest checkpoint and resumes Raft log replay from the recorded position, ensuring correct and efficient crash recovery.
- Concurrency is managed using a combination of execution locks and dedicated thread pools, separating client request handling from background maintenance tasks such as snapshotting and cleanup.