COMPSCI 578 - PA4

GigaPaxos Implementation

This document describes the design, implementation of the Replicated state machine(GigaPaxos) for a fault-tolerant application. Gigapaxos guarantees total ordering of all client requests, and this implementation executes these requests deterministically on Cassandra.

Design Overview:

Implementation handles:

- Deterministic updates to Cassandra
- Handling of duplicates
- Periodic checkpointing of full state
- Ability to restore from a checkpoint

Execution Pipeline & Duplicate Handling

Gigapaxos delivers all requests in a strict total order, through the implementation of execute(). It also ensures idempotency by preventing duplicate execution.

- Each time the execute() is called, it first converts the request into a SQL string, then extracts the unique request ID from the RequestPacket.
- Before executing the SQL, the request ID is checked against the in-memory executedReqs set and requestLog. If the ID has already been executed, the request is suppressed, preventing multiple executions and non-deterministic state changes.
- If the request is new, the SQL command is executed on the Cassandra database.
- After successful execution, the request ID is added to the executedReqs set and requestLog. The log maintains a (MAX_LOG_SIZE) of recent requests to bound memory usage while providing duplicate detection.
- The LastSequence method updates the latestExecutedSeq by parsing the sequence number from SQL statements for checkpointing.

Checkpointing & Restoration

Checkpointing is required so that replicas don't have to replay their entire operation history to recover.

Checkpoint Handling:

- Checkpoint is stored as a single JSON string that contains all the information needed to restore the full state of the application.
- The process captures both the application's metadata and the database state.
- The state is obtained by executing a full table scan: `SELECT id, events FROM grade;`
- The resulting rows, along with the application's metadata and the duplicate suppression state, are converted into a structured JSON format.

Restoration:

- The application simply reconstructs the exact state saved in the checkpoint instead of replaying old commands. It begins restoration by parsing the JSON checkpoint.
- Next, it restores the in-memory structures used for duplicate suppression, including both the `executedReqs` set and the `requestLog`.
- The existing Cassandra table is cleared deterministically using the TRUNCATE grade; command.
- It then iterates through the JSON-encoded database snapshot and recreates every row using `INSERT` statements, fully restoring the state exactly as it was at checkpoint time.

Testing Results:

Number of test cases passed : All test cases have passed.