Distributed Systems – Fault Tolerance

Lab 2

2021/2022

Quorum replication

Use quorums for consistent replication with the `lin-kv` service. Store data as $key \rightarrow (value, timestamp)$, a set of locked keys, and implement the following operations:

- **Read** Collect $(value, timestamp)$ from a read quorum, disregarding locking state, and return the $value$ with the highest $timestamp$.
- **Write** In two steps:
  - Step 1: Pick a write quorum and collect $timestamp$ from them, acquiring locks and returning an error if the lock is already taken.
  - If the quorum is not available, return error 11 to client and give up, informing replicas to release the locks.
  - Step 2: Select the highest $timestamp$ and send $(value, timestamp + 1)$ to the write quorum.
  - In each server, update $(value, timesamp)$ and return an acknowledgment.
  - Wait for all acknowledgments and reply to client.
- **CAS** Similar to write, but use $read \cup write$ quorum; collect values in step 1 to validate that $from$ matches previous value.

Steps

1. Implement the quorum protocol.
2. Test with different quorum combinations (ROWA, majority, ...).
3. Retest with increasing request rate (`--rate`) and network latency (`--latency`).
4. Discussion topics: Do reads need to respect locking? Why do writes need to be acknowledged? Does the protocol tolerate crashes?

Learning Outcomes Apply quorum replication for build a concurrent linearizable register. Recognize the relevance of quorums for fault tolerance.