CapsuleDB: A DataCapsule Based Key-Value Store

Project 18: William Mullen, Nivedha Krishnakumar, and Brian Wu

Problem

- Paranoid Stateful Lambdas (PSL) is a distributed Function-as-a-service (FaaS) framework. It leverages the Global Data Plane's (GDP's) architecture to enable large-scale parallel, secure, and stateful computation in the cloud and on the edge.
- PSL is limited by enclave memory size restrictions, poor bulk data latency, and slow crash recovery.
- CapsuleDB is a level-tree based key-value store that seeks to solve these problems by acting as persistent storage for PSL.

DataCapsule Background

- A DataCapsule (DC) is an append-only cryptographically hardened bundle of data. It's structure resembles a "blockchain in a box", with hash pointers taking the role of backlinks.
- DC has a standardized format which provides strong integrity, authenticity, privacy, and provenance guarantees.
- When combined with the GDP, they become a highly available and flexible storage system.

Paranoid Stateful Lambda Background

- The PSL project utilizes DCs and the GDP to create a federated and stateful FaaS environment that preserves security on the edge and in the cloud.
- PSL workers leverage the Secure Concurrency Layer (SCL) to communicate and loosely synchronize their internal memtables.
- The PSL interface resembles a standard key-value store with put and get functions. This way, developers can use a familiar interface while PSL leverages enclaves to securely add data to a DC.
- PSL manages worker authentication using a Common Access API (CAAPI).

Solution

- CapsuleDB is a DataCapsule backed key-value store that follows a level-tree structure similar to RocksDB or SplinterDB.
- When a client reads from the database, CapsuleDB checks the memtable for recent data. If key is present it either multicasts or sends key to the requested enclave. If key is not found, the search continues to check the index as shown in Fig 1.
- The index lists all the active data blocks associated with the database instance. It checks each level in sequence before eventually either returning the value or determining the key does not exist.
- Writes always go to CapsuleDB's memtable first. Once the memtable fills, it is written out to level 0. Over time, older data will be moved down into the larger levels, which have slower search speeds due to their size, in a process called compaction.
- All CapsuleDB data is still written to the DataCapsule along with individual KV pairs as in the original version of PSL for durability.

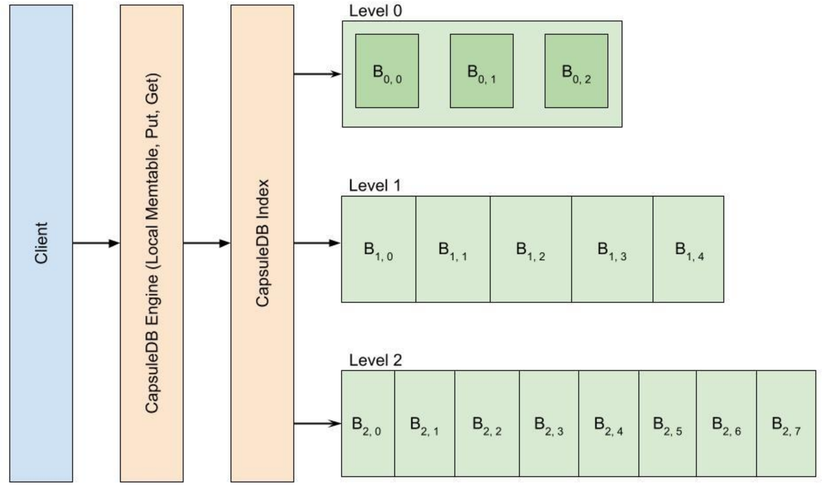

Fig. 1 CapsuleDB's structure with memtables, levels(L0-L2) indexed by CapsuleDB Index

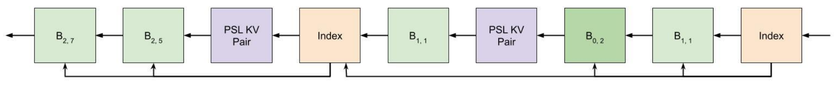

Fig. 2 DataCapsule representation of the Database

Results

- Base results show optimized SCL outperforms Redis+GrapheneSGX on almost every workload.
- We expect this to further improve with the addition of CapsuleDB.
- Furthermore, as a standalone unit, we expect CapsuleDB to be competitive with RocksDB and SplinterDB.
- We expect the write amplification to be similar to SplinterDB, approximately a 2x improvement over RocksDB.
- Finally, we expect to see substantial recovery time speed up since a full scan of the DataCapsule is unneeded.