COSI 167A - Fall 2024 - Research Project

Title: *Self-Designing LSM Memory Buffer*

Background: The log-structured merge (LSM) tree is a disk-based data structure that is highly write-optimized, and thus, widely used in modern NoSQL-based key-value stores [1,2]. While most of the data in an LSM-based storage engine resides on secondary storage, the data structure (such as vector, skip-list, hash-linked list, etc.) used to implement the memory buffer and its tuning affect the performance of the storage engine significantly. Choosing the correct memory buffer implementation and data structure configuration is difficult, especially, if the workload and performance goals shift.

Objective: In this work, we will take RocksDB as the LSM-engine of our interest and study its implementation of range deletion.

(a) Step 1: Explore the design space of buffer data structures in LSM-trees and understand their performance tradeoff. (optional: Implement a lightweight updatable data structure, such as a trie, and benchmark its performance.)
(b) Step 2: Design an ML-powered optimizer that can automatically suggest the optimal buffer implementation and tuning for a given workload and performance target.
(c) Step 3: Integrate the optimizer with a commercial LSM-engine, such as RocksDB, to experimentally verify its correctness.
(d) Step 4: Update the engine's codebase to switch to automatically the optimal buffer implementation (and tuning) if there is a shift in the workload and/or a change in the target performance.

References

[1] O'Neil et al., "The Log-Structured Merge-Tree (LSM-Tree)", Acta Informatica, 1996.
[2] Luo and Carey, "LSM-based Storage Techniques: A Survey", The VLDB Journal, 2020.
[3] Kaushik and Sarkar, "Anatomy of the LSM Memory Buffer: Insights & Implications", DBTest, 2024.