COSI 167A - Fall 2024 - Research Project

Title: *Implementing Compaction Policies on RocksDB*

Background:
Log-structured merge-trees (LSM-trees) [1,2] are widely used in the storage layer of state-of-the-art key-value stores as they offer fast ingestion performance and competitive reads, a common requirement for many modern applications. LSM-trees organize the data in the form of a hierarchical collection of sorted runs which are periodically merged to improve query performance. This process of merging multiple sorted runs to create a single longer sorted run is known as compaction [3,4]. Compactions affect the performance of an LSM-based storage engine across multiple performance axes.

Objective:

The objective of this development project is to implement a new set of compaction strategies on RocksDB (a widely used, open-source, commercial LSM-based key-value store) and compare their performance.

(a) Implement compaction policies that write data in the following data layouts: (i) leveling, (ii) tiering, and (iii) i-leveling.
(b) Implement compaction policies that compact data based on the following data movement policies: (i) full-level compaction, (ii) partial-level compaction, and (iii) tiered compaction.
(c) For partial-level compaction, implement the following data movement policies: (i) pick the oldest file in a level, (ii) pick the least recently accessed file, and (iii) pick the file with the most number of tombstones.
(d) Compare the different compaction policies in terms of performance.

References:
[1] O'Neil et al. The Log- Structured Merge-Tree (LSM-Tree). Acta Informatica. 1996

[2] Facebook. RocksDB. http://github.com/facebook/rocksdb.

[3] Sarkar et al. Constructing and Analyzing the LSM Compaction Design Space. VLDB 2021.

[4] Sarkar et al. Compactionary: A Dictionary for LSM Compactions. SIGMOD 2022.