COSI 167A - Fall 2024 - Research Project

Title: *Finding the optimal caching policy in LSM-trees*

Background:
Log-structured merge-trees (LSM-trees) [1] 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. To enhance query performance, LSM-based storage engines employ lightweight auxiliary data structures, such as Bloom filters and zone maps (also known as fence pointers). In addition to this, commercial LSM-engines often use block-based caching for the index, filter, and data blocks. Since memory is a limited resource, the decisions of which blocks to cache and for how long are crucial for lookup performance. However, commercial systems treat all blocks equally evicting the index, filter, and data blocks with equal priority.

Objective:

The objective of this project is to find out the optimal granularity of the caching policy in LSM-trees at the run time. The workflow for this project is as follows.

(a) Get familiarized with the query procedure using block cache in the vanilla implementation of the LSM-trees [2].
(b) Design the cost model of the lookup that can find the optimal granularity of index blocks based on the workload statistics.
(c) Benchmark the performance of RocksDB with the block cache disabled and enabled with different sizes of the block cache.
(d) Quantify the impact of pinning the index and filter blocks in memory.
(e) Implement the proposed solutions on top of RocksDB, and analyze their performance with respect to the state-of-the-art.

References:
[1] Patrick E. O'Neil, Edward Cheng, Dieter Gawlick, Elizabeth J. O'Neil. The Log- Structured Merge-Tree (LSM-Tree). Acta Inf. 33(4), (1996)

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