CAS CS 561: Data Systems Architectures
Data-intensive Systems and Computing Lab
Department of Computer Science
College of Arts and Sciences, Boston University
http://bu-disc.github.io/CS561/

CS561 Spring 2023 - Systems Project

Title: *Implementation of an LSM-Tree based key-value store*

Background: Log-structured merge-tree (LSM-trees) [1, 2, 3] are one of the most commonly used data structures for persistent storage of key-value entries. LSM-tree-based storages are in use in several modern key-value stores including RocksDB at Facebook, LevelDB and BigTable at Google, bLSM and cLSM at Yahoo!, Cassandra and HBase at Apache, and so on. LSM-trees store data in the disk as immutable logs (also known as sorted sequence tables (SSTs)), which are maintained in hierarchical levels of increasing capacity. To bound the number of logs that a lookup has to probe, LSM-trees merge logs of similar sizes. The two possible merging strategies are (i) leveling (optimized for lookups) and (ii) tiering (optimized for updates).

Objective: The objective of the project is to implement an LSM-tree using only a single process thread. The workflow for this is as the following.

(a) Review the LSM-tree literature to understand the principles and operations supported in an LSM-tree.
(b) Implement an LSM-tree (vanilla implementation) using only a single process thread for both merging strategies — leveling and tiering.
(c) Develop by cloning the API available to you at: https://github.com/BU-DiSC/cs561_templatedb. Note that you are expected to build a more extensive testing infrastructure.
(d) Provide the source code. The performance of your implementation will be regularly measured, and the best performer will be announced on the website.

Note: There are two more restrictions when implementing range query and deletes.
(1) The output of range query should be sorted and you shall not use a large vector to store intemediate results from all the levels and sort them together to output, this means that you need to implement the merge process of several iterators in different runs to avoid unnecessary space amplification.
(2) The range delete implementation should have similar logic to the point tombstone, as you learned in class. You shall not implement range query/delete by iterating every element between min and max then issue point query/deletes.

[1] Chen Luo, Michael J. Carey. **LSM-based storage techniques: a survey.** VLDB J. 29(1): 393-418 (2020)

[2] Niv Dayan, Manos Athanassoulis, Stratos Idreos. **Monkey: Optimal Navigable Key-Value Store**. SIGMOD Conference 2017: 79-94

[3] Patrick E. O'Neil, Edward Cheng, Dieter Gawlick, Elizabeth J. O'Neil. **The Log-Structured Merge-Tree (LSM-Tree)**. Acta Inf. 33(4): 351-385 (1996)

CAS CS 561: Data Systems Architectures
Data-intensive Systems and Computing Lab
Department of Computer Science
College of Arts and Sciences, Boston University
http://bu-disc.github.io/CS561/