CS591A1 Spring 2019 - Research Project

Title: *Persistent deletes in LSM-trees*

Background: Log-structured merge-trees (LSM-trees) [1, 2] are one of the most commonly used data structures for persistent storage of key-value entries, and are widely adopted in the storage layers of several modern key-value stores incuding RocksDB at Facebook, LevelDB and BigTable at Google, bLSM and cLSM at Yahoo!, Cassandra and HBase at Apache, and so on. Principally, LSM-trees perform out-of-place updates to support faster rate of ingestion for key-value entries. At the cost of a nominal amount of disk space, this improves the update performance as well as the lookup performance. While, through this out-of-place updates, LSM-trees are able to support efficient deletes as well, the present state-of-the-art designs are not optimized to support *persistent deletes*. Rather persisting a delete in a LSM-tree can turn out to be highly expensive, in terms of the memory requirement, in certain cases.

Objective: This project is designed to be research-oriented, the workflow of which is as the following.

(a) Get familiarized with the read/write operations in the vanilla implementation of the LSM-tree
(b) Analyze the cost (in terms of I/O or main memory footprint) of persisting a delete in an LSM-tree
(c) Propose a mechanism, which offers efficient (in terms of I/O or/and memory) performance in persisting deletes in LSM-trees while ensuring competitive read/write performance

[1] 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)
[2] Niv Dayan, Manos Athanassoulis, Stratos Idreos. **Monkey: Optimal Navigable Key-Value Store**. SIGMOD Conference 2017: 79-94