CAS CS 660: Grad. Intro to Database Systems
Data-intensive Systems and Computing Lab
Department of Computer Science
College of Arts and Sciences, Boston University
http://bu-disc.github.io/CS660/

CS660 Fall 2024 – Written Assignment

Title: LSM Trees

Q1:

a) You are tasked with designing a real-time analytics platform. For the current implementation, we will focus on the following requirements:

1. High write throughput
2. Balanced read performance
3. Low latency: Both read and write operations need to be executed with minimal delay.
4. Efficient storage utilization

Given these requirements, would you prefer to implement the storage using an LSM tree, or a B-tree?

b) Suppose the system must now handle a workload with highly skewed access patterns, i.e., a small subset of data (hot/recent data) is accessed much more frequently than the rest (cold/older data). How would this new requirement influence your choice of data structure?

Q2. Consider that we are constructing LSM trees by inserting the following sequence of key-value (k,v) pairs.
(8, “I”), (2, “K”), (1, “O”), (10, “L”), (42, “G”), (8, “D”), (23, “Y”), (31, “M”) , (6, “B”), (23, “X”), (15, “V”), (3, “N”), (26, “C”), (36, “R”), (20, “Z”)

1. Assuming the write buffer can store up to 3 entries and we are using the basic LSM-tree structure, show the LSM tree structure after every 3 inserts. If a compaction occurs, show the tree structure before and after compaction.
2. Assuming the same write buffer size, show the tree structure (after every 3 inserts) if we use leveling LSM-tree with T=3
3. Assuming the same write buffer size, show the tree structure (after every 3 inserts) if we use tiering LSM-tree with T=3

Q3. Consider the following key set (20, 38, 70, 13, 5)

1. Please build the bloom filter using two hash functions $h_1(x) = (2x + 3) mod\ 17$ and $h_2(x) = (5x + 4)\ mod\ 17$ with a 17-bit vector
2. What is the bloom filter query result for key 21? How about key 22, key 25?
3. Rebuild the bloom filter using the hash functions $h_1(x) = (2x + 3)\ mod\ 29$ and $h_2(x) = (5x + 4)\ mod\ 29$ with a 29-bit vector. What are the query results for keys 21, 22, and 25? What conclusion do you have by comparing the accuracy of the two Bloom Filters?

Q4.

a) You are managing the database for a popular social media platform like Twitter or Instagram, which experiences heavy write loads due to users continuously posting updates, comments, and likes. The system primarily performs point queries to fetch individual posts or user profiles. Would you prefer a leveled or tiered compaction strategy for your LSM tree?
b) If the read queries shift to predominantly range queries over recent data—such as displaying a user's newsfeed or retrieving a list of recent comments—how would this affect your choice between leveled and tiered compaction?

Q5. You are designing a LSM tree (different from the one we discussed in class) with the following characteristics:

1. Leveled compaction strategy
2. 4 SSTables in level 0, i.e. level 0 contains multiple SSTables with **overlapping key ranges** due to recent flushes from the memtable.
3. 5 levels in the LSM tree, i.e. level 1-level 4 contain SSTables with **non-overlapping key ranges**. Each key appears in at most one SSTable per level.
4. Each level is 10 times larger than the previous one

a) Estimate the number of disk reads required to perform a point lookup query without Bloom filters in the worst-case scenario.
b) Repeat the calculation assuming a tiered compaction strategy.
c) If Bloom filters are used with a 1% false positive rate, how would this affect the query cost in both leveled and tiered compaction strategies?
d) Write Amplification is the ratio of the total amount of data written to the LSM tree to the amount initially written by the user. It occurs because data is rewritten multiple times during compaction processes as it moves from lower(L_0) to higher levels(L_5). Estimate the write amplification factor due to compaction processes in leveled compaction.

Q6. You are managing the backend storage for a popular social media platform like Twitter, which uses an LSM tree to store and retrieve tweets. The platform handles a massive dataset with 100 million unique tweets (keys) and experiences a high volume of both read and write operations. Users frequently query for tweets, some of which may not exist due to deletion or incorrect URLs.

i) Option X: Bloom filter with a false positive rate of 0.1%
ii) Option Y: Bloom filter with a false positive rate of 1%

1. Which option would you choose to optimize overall system performance if I have a memory budget of 200MB?
2. If the memory budget is changed to 500KB, what would be the best option?