BIG DATA MANAGEMENT

CZ/CE4123

Tutorial 10
Key-Value Stores

QUESTION 1

Consider a leveling LSM-tree with a size ratio 4. The memory buffer (Level 0) can store 5000 key-value pairs. Initially the LSM-tree is empty. After inserting 70000 key-value pairs with distinct keys continuously, how many levels are formed?

QUESTION 1-SOLUTION

This question is related to understanding the capacity.
The capacity of a level is defined as “the maximum number of key-value pairs that can be stored in the level”.

Size Ratio = 4
Level 0 capacity = 5000
Level 1 capacity = 5000x4=20000
Level 2 capacity = 5000x4x4=80000

70000 > Level 0 capacity + Level 1 capacity (=25000)
70000 < Level 0 capacity + Level 1 capacity + Level 2 capacity (= 105000)
So there are at least 3 levels (Level 0, Level 1, Level 2)

QUESTION 1-SOLUTION

We further verify that Level 3 is not created. (why?)

If level 2's actual size is larger than Level 2 capacity – Level 1 capacity, it can already trigger the merge

Before creating Level 3, it must trigger the sort-merge of Level 2. Since the 70000 keys are distinct, the actual size of Level 2 must be a multiple of the capacity of Level 1 (i.e., 20000). So only when the actual size of Level 2 reaches 80000 it can trigger a merge, (note that 0, 20000, 40000, 60000 are not larger than {Level 2 capacity – Level 1 capacity}, and hence will not trigger a merge). However, since there are only 70000 keys, it is impossible for Level 2 to reach a size of 80000. Hence, Level 3 will not be created. So finally there are 3 levels (Levels 0, 1, 2).

QUESTION 2

Consider a leveling LSM-tree with a size ratio 4. The LSM-tree has 5 levels (excluding the memory buffer level), and it is incorporated with **both** fence pointers and Bloom Filters. Assume that a key-value pair is always entirely stored within a disk page. Consider the procedure of Get(*K*) for a key *K*, which of the followings sequence are possible to be the I/O costs from Level-0 to Level-5? (can select multiple answers)

(a) 1, 1, 1, 1, 1, 1

(b) 0, 1, 1, 1, 1, 1

(c) 0, 0, 1, 0, 0, 1

(d) 0, 0, 0, 0, 0, 1

(e) 1, 0, 0, 0, 0, 0

SOLUTION

(a) and (e) are not possible because Level 0 is memory buffer, and hence no I/O costs.

(b) (c) (d) are possible because if *K* is not in the LSM-tree, then each level's I/O cost fully depends on the accuracy of the Bloom filter (We consider Fence Pointer will incur 1 I/O when Bloom filter returns true).

- (b) is possible: BFs in Levels 1-5 all generate false-positives
- (c) is possible: BFs in Levels 2 and 5 generate false-positives
- (d) is possible: BF in Level 5 generates a false-positive

QUESTION 3

Consider a leveling LSM-tree with a size ratio 4. The LSM-tree has L levels (excluding the memory buffer level), and it is **only** incorporated with fence pointers (without Bloom Filters). Assume that a key-value pair is always entirely stored within a disk page. Consider the procedure of Get(K) for a key K,

(1) What is the possible I/O cost of accessing Level-i (i is in [1, L])?

(2) If K exists in the LSM-tree, what is the expected I/O cost at Level-i (i is in [1, L])? (hint: divide the cases based on the first-appearing location of the key K)

SOLUTION FOR Q3(1)

The possible I/O cost at Level-*i* can be 0 or 1.

0 is possible:

if *K* first appears in a level smaller than Level-*i*, then the search is terminated before Level-*i*. Hence, no I/O cost at Level-*i*.

1 is possible:

if *K* first appears in Level-*i* or larger levels, then fence pointer is used and can incur 1 page read, or 1 I/O. (Note: We assume that using fence pointers always incurs 1 I/O.)

SOLUTION FOR Q3(2)

Let j be the first level that contains key K.

Divided into two cases:

- If $i>j$, then the (expected) I/O cost is 0, because the search ends at Level j.

- If $i<=j$, then the (expected) I/O cost is 1, because at level i fence pointer is used and incurs 1 I/O.

QUESTION 4

Consider a leveling LSM-tree with a size ratio 4. The LSM-tree has L levels (excluding the memory buffer level), and it is incorporated with **both** fence pointers and Bloom Filters. Assume that a key-value pair is always entirely stored within a disk page. Consider the procedure of Get(K) for a key K,

(1) What is the possible I/O cost of accessing Level-i (i is in [1, L])?

(2) If K exists in the LSM-tree and the FPR of the Bloom filter at Level-i is P (P is in [0, 1]), what is the expected I/O cost at Level-i (i is in [1, L])? (hint: divide the cases based on the first-appearing location of the key K)

SOLUTION FOR Q4(1)

The possible I/O cost at Level-*i* can be 0 or 1.

0 is possible: if *K* is not in the LSM-tree, and the BF in level-*i* returns FALSE. Then, the search within the disk for this level is skipped.

1 is possible: if *K* is not in the LSM-tree, and the BF in level-*i* returns TRUE. Then, fence pointer is used and incurs 1 page read, or 1 I/O. (Note: We assume that when using fence pointers always incurs 1 I/O.)

SOLUTION FOR Q4(2)

Since K exists in the LSM-tree. The cases is divided based on the first-appearing level of key K.

Let Level-j be the level that first contains K.

Case 1: If $i>j$, the search is up to Level-j, and terminates before reaching Level-i, and hence the I/O cost at Level-i is 0;

Case 2: If $j=i$, the I/O cost at level-i must be 1 because the key first appears at Level-i.

Case 3: If $i<j$, the I/O cost depends on the FPR of the Bloom filter:

- Since the key K does not exist in Level-i, then with probability P the Bloom filter returns TRUE, and later the fence pointer incurs 1 I/O.
- Since the key K does not exist in Level-i, then with probability 1-P the Bloom filter returns FALSE, and no I/O cost is incurred.
- To summarize, the expected I/O cost is : $P*1+(1-P)*0=P$.