Pebble Db – Key Value Store Using Fragmented Log Structured Merge Trees (Part 1)

Presented by –
Deepak Varghese
Pebble DB Overview

- High performance write-optimized key-value store
- Built using new data structure Fragmented Log-Structured Merge Tree
- Fragmentation is done using guards
- Helps in reducing the write amplification
- Range search can be performed
Question 1 -

“Figure 2 illustrates compaction in a LSM key-value store.” Please use the example to explain why compaction operation can be very expensive.

- Multiple rewrites occurring when compaction is done on the LSM key value store.
- This leads to high write amplification.
Question 2 - “Instead of rewriting the sstable, FLSM’s compaction simply appends a new sstable fragment to the next level.” Compared to the LSM-tree in-place rewriting, this appending is more efficient. However, what’s the tradeoff (any negative impact of the appending)?

- Multiple sstables can have the same key and can have overlapping key ranges on the same level.
- This would affect the read performance.
- It would also lead to false positive cases.
Question 3 - “FLSM performance is significantly impacted by how guards are selected.” Could you give a criterion of being good guards?

- Guards should be able to separate ranges efficiently so that it doesn’t have multiple sstables.
- The guards selected should be based on the higher density of keys.
- Guards are selected based on guard probability value.
- Guard probability is lowest at lower levels and increases as level number increases.
Question 4 - “Guard probability $gp(key,i)$ is the probability that key becomes a guard at level $i$.” Why is the probability a function of level number?

- Choosing guards based on level number helps in having better key distribution between guards.
- The intervals are better defined as levels increase.
- As level number increases the number of keys and sstables increase. Hence we need more guards.
Question 5 - Guards are continuously generated with key insertions. And “We note that in many of the workloads that were tested, guard deletion was not required.” Could you solve the contradiction? What’s the consequence of not conducting guard-deletion operations in a store keeping admitting new keys?

- Not performing guard deletion does not affect read and range search performance as get and range query operations skip over empty guards.
- We can not have large number of guards when we have lower number of keys.
- Deleting guards help in even distribution of the keys among them.
- Thus consolidating data among fewer guards help in improving performance.
Questions ?