PebblesDB: Building Key-Value Stores using Fragmented Log-Structured Merge Trees (I)

NOTE: Your slides/presentation only need to cover background information necessary to answer the given questions (in a clear and well-organized manner). You are allowed to borrow contents from other resources, such as online slides, as long as you acknowledge them. The presentation should be mostly question-focused and proceed mostly in a Q&A format. Please include the questions in your slides. Don’t write detailed answers in the slides and read them to the class. Instead, use bullet points, graphs, or animations to explain your answers to the class.

In your Q&A report, use text to more thoroughly answer the questions. Include a short paragraph at the beginning of the report to summarize the paper.

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

(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)?

(3) “FLSM performance is significantly impacted by how guards are selected.” Could you give a criterion of being good guards?

(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?

(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?