A review of Monkey: Optimal Navigable Key-Value Store

Mahim Mahbub - 1505022

September 3, 2020

1. What is the problem the paper deals with?

   The problem the paper deals with is that existing LSM-tree based designs assign the same false positive rate to every Bloom filter regardless of the run size. This leads to larger runs having proportionally larger size for Bloom filters but due to fence pointers the I/O cost of probing a run remains the same regardless. So, the same memory can have a higher impact in reducing the false positive rates of smaller runs.

2. Why is the problem important?

   The problem is important because most modern key-value stores use LSM-tree structure. The existing systems do not optimally assign main memory for Bloom filters. This leads to sub-optimal lookup costs. Hence, these existing systems are not tuned along the optimal Pareto curve i.e. the curve beyond which lookup cost cannot be improved without harming update cost and vice versa. Hence, it was important to make a model which could navigate the Pareto curve to find the optimal balance between the lookup cost and update cost for a fixed main memory budget and an application workload.

3. What is the proposed solution?

   The proposed solution for this paper is to allocate main memory to Bloom Filters across different levels such that the sum of the false positive rates is minimized i.e. setting the false positive rates of each Bloom filter to be proportional to the number of entries in the run that it corresponds to. The intution behind it is that for a fixed amount of main memory allocated to Bloom Filters for all runs, larger size runs only bring about minor benefits compared to smaller size runs i.e. the same amount of memory can have a higher impact in reducing false positive rates of smaller size runs. This leads to the insight that the lookup cost is proportionally related to the sum of false positive rates of all the Bloom filters.

4. Why is their solution unique i.e. Strengths of the solution.

   The solution is unique because none of the related previous works were on Bloom filters. Monkey optimizes lookup cost by optimally allocating main memory for the Bloom fiters. Monkey has the ablility to predict how changing one design decision can impact the worst-case cost performance. And, also the ability to holistically auto-tune to maximize the worst-case throughput. Scales are better and easier to tune. Experimental analyses show that Monkey scales better with bigger data volume. Monkey can match the same performance as existing systems with smaller main memory. Monkey improves lookup latency for different variety of temporal locality. Also, Monkey can utilize the cache and show its superior performance due to the better allocation of false positive rates across the Bloom filters.

5. Limitations and future work:

   A possible future work could be experimenting for distributed systems to evaluate the performance metrics of this model. Another possible future work could be to design cache for key-value pairs. Also, to run further experiments to tune the cache and to reduce I/O for true positives.