Speeding up Sampling in Key-Value Stores

Xi Yang [1], Jiannan Wang [2], Jinglin Peng [2], Jing Yan [3]

[1]Zhejiang University [2]Simon Fraser University [3]Hongkong University
{xya81, jnwang, jlpeng}@sfu.ca jyan@cs.hku.hk

Motivition

- Key-Value(KV) store is a type of storage that retrives a value with a key.
- KV stores usually adopt log structure merge(LSM) tree to store the key-value pairs.
- LSM tree is optimized for write, but not for read-heavy analytic query.
- One approach to run analytic query fast is sampling.
- However, most existing KV stores sample very slow or can not sample correctly.

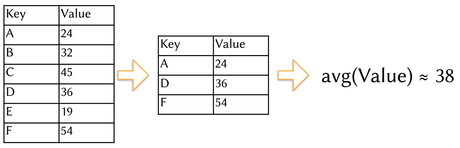


Figure 0. Key-Value data model, sampling and approximation. Imagine a scene where ages are stored as values, and user wants to query average age.

Log-Structured Merge-Tree (LSM-Tree)

- A Log-structured merge-tree[1] contains two or more tree-like data structures.
- Small C_0 tree is in memroy, large C_1 tree is on disk. The implementation of each tree can vary.
- C_0 tree is merged to disk tree on reaching threashold size.

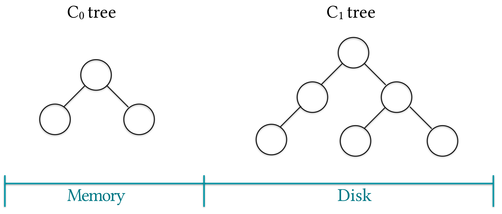


Figure 1. The LSM-Tree with two components

- New records are inserted to the memory resident C_0 tree.
- When querying a key, C_0 tree is firstly searched, then C_1 tree is searched if C_0 misses.

Multiple level LSM-Tree

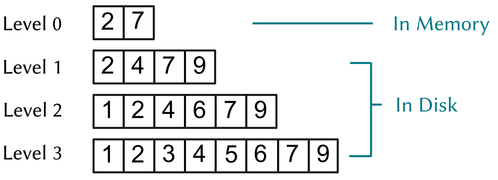


Figure 2. LSM-Tree with multiple levels

- Similar to two component LSM-Tree, but contains multiple disk resident sorted runs.
- A level is compacted to next level on reaching threashold size.
- A query searchs key from small to large levels. Records at smaller level are always newer.

Challanges

- Simple Bernouli sampling does not work due to duplicated keys.
- It gives a biased sampling, the duplicated keys has higher propobility to be sampled.

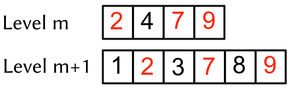


Figure 3. Example of duplications in LSM-Tree

- Iterators could scan all the data and eliminate dupliations, but too slow.

Skip Approach

- Records on the disk have variable length. Iterators parse every key-value pair, and sync different levels.

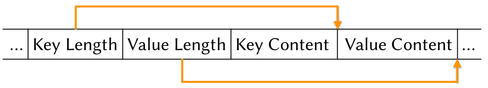


Figure 4. Typical record orgnization in LSM Tree

- Our method does not parse every record. We randomly skip some records.
- Only the small part of sampled records are parsed. For the skipped records, only read the key-value length.

Figure 5. Only parse the sampled records. Others are skipped.

Duplication Elimination: Bloom Filter

- No duplicated key within one level.
- Bloom filter is applied to check duplication in different levels.
- Check each sampled record at all levels. Accept with propobility 1/(#duplication).
- Time complexity is O(|S| |L|), where |S| is samples size, |L| is number of levels.

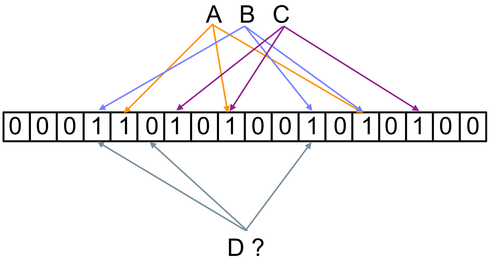
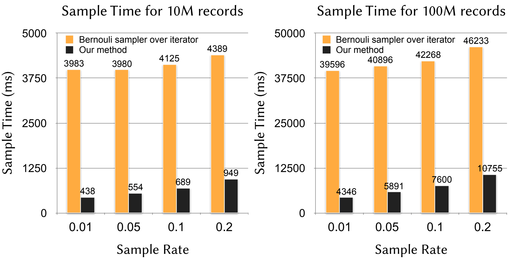


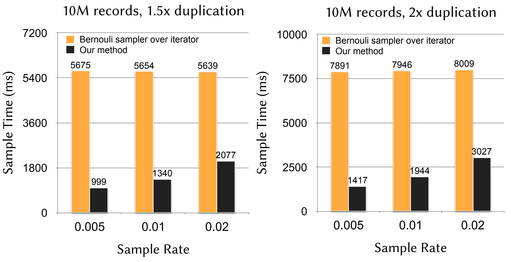
Figure 6. How Bloom filter works: apply multiple hash functions to a record, write the result to a bit array.

Experiments

- Implementation based on LevelDB.
- First group of experiments guarantee no duplication, thus bloom filter is not used.



- Second group contains averagely 1.5 or 2 duplication for each record.



Conclusion

- Our methods is 3 to 9 times faster than iterator based sampler.
- Unbiased sampling can be achieved by using Bloom filter

References

[1] Patrick O'Neil, Edward Cheng, Dieter Gawlick, and Elizabeth O'Neil. 1996. The log-structured merge-tree (LSM-tree). *Acta Inf.* 33, 4 (June 1996), 351-385.