SILT: A Memory-Efficient, High-Performance Key-Value Store

Presented by:
Ramya Danappa
Seesaw Game?

How can we improve?

Memory efficiency

High performance

SkimpyStash

FAWN-DS
FlashStore
HashCache
BufferHash
(1) “The memory overhead and lookup performance of SILT and the recent key-value stores. For both axes, smaller is better.” Explain the positions of FAWN-DS, SkimpyStash, BufferHash, and SILT on the graph.
Describe SILT’s structure (Architecture of SILT). Compared with LevelDB, SILT has only three levels. What’s concern with a multi-level KV store when it has too few levels?
“SILT uses a memory-efficient, high-performance hash table based upon cuckoo hashing.”. Explain what the cuckoo hashing is and why it is used.

Design of LogStore: an in-memory cuckoo hash table (index and filter) to describe how a PUT request and a GET request is served in a LogStore. In particular, explain how the tag is used in a LogStore for cuckoo hashing.
Explain how a LogStore is converted into a HashStore?

Figure 4: Convert a LogStore to a HashStore. Four keys K1, K2, K3, and K4 are inserted into the LogStore, so the layout of the log file is the insert order; the in-memory index keeps the offset of each key on flash. In HashStore, the on-flash data forms a hash table where keys are in the same order as the in-memory filter.
“Once a LogStore fills up (e.g., the insertion algorithm terminates without finding any vacant slot after a maximum number of displacements in the hash table), SILT freezes the LogStore and converts it into a more memory-efficient data structure.” Compared to LogStore, what’s the advantage of HashStore? Why doesn’t SILT create HashStore at the beginning (without first creating LogStore)?

Advantage of Hashstore over logstore:
• Hashstore saves memory over Logstore by eliminating the index and reordering the on-flash (key,value) pairs from insertion to hash order.

SILT create Logstore first because so every data ideally should store on flash.
“When fixed-length key-value entries are sorted by key on flash, a trie for the shortest unique prefixes of the keys serves as an index for these sorted data.” While a SortedStore is fully sorted, could you comment on the cost of merging a HashStore with a SortedStore? Compare this cost to the major compaction cost for LevelDB?

- Cost is proportional to the size of Database.
- SILT is worse because single sorted store leads to bad write amplification.
Questions?
Thank you