Paper Review:
"*Towards Building a High-Performance, Scale-In Key-Value Storage System*"

Seri Lee
Seoul National University
sally20921@snu.ac.kr

***Abstract*—**

***Index Terms*—**

I. Challenges and Solutions for Conventional Key-Value Store

KV SSD offloads the key-value management to SSD, taking into account such application requirements. A dedicated key-value management layer efficiently harness the SSD internal system resources.

A. Multi-layer Interference

User read or write requests typically travels through multiple I/O layers before being submitted to the device.

Solution: Lightweight Request Translation

Since variable-size key-value pairs are supported, key-value to block translation is no longer required.

B. Resource Contention

Key-value stores need to balance between several foreground and background processes. In practice, for utilizing the device bandwidth, as the number of foreground processes increases, the workload on background processes also increases, which leads to slowing down or stalling of the system. More CPUs are dedicated to support smaller number of high performance storage devices.

Solution: Small Memory Footprint

Regardless of the number of key-value pairs in a device, the host memory consumption of KV SSD remains constant. Additionally, KV SSD allows key-value store applications to easily adopt a shared-nothing architecture as there is no main data structures between devices. By doing so, the scalability limitation from synchronization effects can be avoided.

C. Data Consistency

A write-ahead log WAL is generally used for data consistency. Though it provides consistency, WAL reduces user throughput by half as the total amount of data written doubles.

Solution: Consistency Without Journaling

It is not required to maintain metadata of key-value pairs at host side, eliminating the need for journaling or transactions to store metadata and data together. Consistency of key-value pairs is now managed inside KV-SSD, using its battery-backed in-memory request buffers. Each key-value request is guaranteed to have all or nothing consistency in a device. Therefore, for independent key-value pairs, write-ahead logging (WAL) mechanism is not necessary on host-side.

D. Read and Write Amplification

Key-value stores introduce processes like garbage collection, compaction, and defragmentation. These

significantly increases the read and write amplification as they need to read the written data back and process it. Further, there is internal read and write amplification caused by the SSD's garbage collection process.

Solution: Read and Write Amplification Reduction

The host-side read and write amplification factors of KV SSD remain to be an optimal value of 1, because it does not require additional information to be stored for key-value pairs.

II. In-Storage Key-Value Management

A. Command Processing

Large Variable-sized Key Support

The support for variable-size keys provide the opportunity for applications to encode useful information such as name and type, and more importantly eliminates the requirement of maintaining additional logs and indices for name resolution.

Iterator Support

Internally, all keys matching MSB 4B key keys are containerized into iterate bucket. This bucket is updated in a log-structured manner whenever put or delete operation is processes and periodically, cleaned up by GC.

B. FTL and Indexing

Block-based FTL is extended to support a variable-size key-value pairs. Traditional page-based or block-based mapping technique using LBA as a key cannot be used as a main index structure in KV SSD as the range of keys is not fixed.

C. Garbage Collection

The garbage collector (GC) in an SSD is changed to recognize the key-value pairs stored in a flash page and updated keys in the iterator buckets. The victim selection and cleaning policy is the same as block firmware, with addition to managing the global hash table and iterator buckets.

III. Conclusion

References

[1]
[2]