Concurrent Sequential Writes Also Considered Harmful in Solid State Drives

Soyee Choi*, Dong Hyun Kang**, Sang-Won Lee*, Young Ik Eom*
* Sungkyunkwan University, Republic of Korea
** Dongguk University-Gyeongju, Republic of Korea
ithdli@skku.edu, dhkang@dongguk.ac.kr, {swlee, yieom}@skku.edu

In SSD devices, small random writes are considered harmful since they cause high write amplification during the garbage collection, thus lowering performance and shortening the lifespan. In contrast, large sequential writes are believed to be flash-friendly and thus to barely incur additional write amplification in SSD devices. This poster motivates, contradictory to this expectation, that large sequential writes made from data-intensive applications can amplify writes inside SSD devices, especially when issued concurrently. As a concrete example, we observe that the SSD-internal write amplification factor (WAF) reaches more than four when running YCSB benchmark using RocksDB on top of Ext4 file system with a commercial SSD from a market-leading vendor as its storage. Note that the write unit in RocksDB was set by default to 64MB. The main culprit for this counter-intuitive phenomenon is suspected to be the file system located between applications and storage, which need to be addressed by the file system community.

Writes in RocksDB RocksDB is a popular key-value (KV) store based on log-structured merge trees (LSM) data structure, aiming at taking advantage of high sequential bandwidth by only writing sequentially to storage such as SSD devices and hard disks. In addition to WAL, there are two main operations for data writes in RocksDB, `memtable flush` and `compaction`. When an in-memory memtable of 128MB is full of records, the memtable should be `flushed` to the storage as a new SSTable. Also, the costly compaction operation which is inevitably inherent in LSM-based KV stores creates multiple new SSTable files of 64MB. The patterns of the writes issued by both operations to the file system are mostly sequential. An important assumption behind LSM-based KV stores including RocksDB is that sequential writes to flash storage will minimize the in SSD write amplification; this is particularly true for SSD devices. If this assumption holds in reality, RocksDB will, though not resorting to multi-stream SSDs, show an ideal WAF of one.

Problem Definition Each file write from RocksDB (i.e., 64MB) logically contiguous in terms of file system space allocation . But, when sequential writes are concurrently issued via the underlying file systems, each write of 64MB will be physically split into smaller write chunks because of unknown reasons which have to be investigated later. To be worse, the split write chunks belonging to different files might be grouped into flash superblocks even though each file has a different lifespan. It causes high write amplification because some valid blocks belonging to the same superblock are copied to the new superblock during garbage collection operations. As a consequence, with RocksDB KV-store where sequential writes are dominant, the physical write amplification inside flash storage can reach up to five. This high WAF will deteriorate the write performance as well as lifespan of flash storage devices. The readers might suspect the file system fragmentation as the cause of write splits. That is, if an SSTable file of 64MB is scattered over multiple extents, then the writes to the storage might be split into each extent. However, most SSTables newly created during our experiment have a contiguous LBA address of 64MB. Thus, the file system aging is not the root cause of write splits in RocksDB.

Write Splits in RocksDB This section demonstrates the degree of write splits when running YCSB benchmark using RocksDB on top of Ext4 file system. Using YCSB, we ran 140 million insert (140GB) followed by 2 billion update (2TB) workload. We collect blktrace and SSD internal statistics using smartmonstool to check if issued writes are split and whether it really deteriorates SSD internal WAF; in all experiments, the WAL log was directed to a separate device to remove its interference. Although most SSTable file has contiguous LBA space allocation of 64MB in terms of Ext4 file system, the average length of write chunks at the block trace layer was merely 1.6MB; about 20% of write chunks has 64M size. Compared to the ever-increasing page, block, and erase-unit in SSDs, 1.6MB would be small. Therefore, taking into account that each SSTable has different lifetime, the overhead of garbage collection in FTL has continuously increased over time during the experiment and the final WAF eventually reached more than 4.8.