Final Report for Storage Systems Project 1: myFTL

Author: Danyang Ren   AndrewID: danyangr

The goal of this project is to design and implement a Flash Translation Layer, which is called myFTL and used by SSD.

There are mainly three parts of the FTL for this project: one is address translation components that maps Logical Block Address (LBA) to Physical Page Address (PBA) in the SSD; a garbage collection component that removes invalid data and make space for new data; a wear leveling component that ensures that all the blocks in the SSD wear out at a near-uniform rate.

We need to improve three basic metrics for those three parts: write amplification, maximum writes observed and memory usage. Since once a page is written then it can’t be written again before the block being erase, we need to read the valid pages in this block, copy it to somewhere else, erase this block and copy back, then write data in this page. As a result, write amplification is caused. We may also need to remap certain blocks due to wear leveling, which will also cause write amplification. Maximum writes observed can be improved by good design in wear leveling. The reason we need it is because SSD can only tolerate certain number of erase operations for one block. Then those blocks behave badly and we shouldn’t use them anymore. Some blocks can wear out quickly due to the characteristics of the data it saved, and certain blocks like log reservation blocks and cleaning reservation blocks are more likely to wear out. Remapping may be adopted to change the PBA of those blocks. Memory usage are caused by keeping those mapping relations from LBA to PBA, from data blocks to over-provisioning blocks, block erase times record, etc.

Due to the limited time, I only adopted a few strategies to improve those metrics. I will talk about the strategies I used and the strategies I want to use but not finished in next paragraphs.

To improve memory usage, I changed the bad design used before as one map use LBA as key and PBA as value. But then I found that I only need to compute the block index of LBA, then use it as the key; correspondingly, the value is also the PBA block index. Given a LBA, we can find the corresponding physical page block and use mod operation to get the specific PBA.

I also tried to improve the number of maximum writes observed by changing mapping of log reservation block and cleaning reservation block. When a log reservation block or cleaning reservation block reaches its dead, I tried to find another block in over-provisioning blocks to replace original ones. But this strategy only improves the performance a little due to the small percentage of over-provisioning blocks.

Strategies I want to use but not implemented yet are listed as follows.

The most important one is to map the location of log reservation block and cleaning reservation block to all the blocks of SSD, not limited to certain percentage of all the blocks as before. Due to the small amount of over-provisioning blocks and the high frequency usage of those block, those blocks are more likely to wear out. So it’s important of the wear leveling algorithms to change the location of those blocks, rather than just use static mapping for them.

The second one is the TRIM function, which is used by file system to inform SSD that certain LBAs are outdated. The workload of Garbage Collection can be greatly reduced with the help of this function.

The last one is the Garbage Collection algorithm. I adopted Round Robin algorithm due to it’s simplicity, but I think LFS Cost-Benefit policy may be a better choice as it take the advantage of both Least Recently Used Block and Greedy by Minimum Effort algorithms. The efficiency of this algorithm can also be greatly improved by the TRIM.

Up to now, my own FTL can only works well in situations that the workload and erase time are evenly distributed across all those blocks. Due to the static mapping of over-provisioning blocks, if there are many erase in this area, those blocks will wear out quickly.

The code structure for my myFTL is pretty simple. Helper function performErase() is used by cleaningForFullLogReservationBlock() and roundRobinGarbageCollection() to perform erase in certain data block and overprovision block. cleaningForFullLogReservationBlock() is triggered when one block is full but has corresponding log-reservation block, while roundRobinGarbageCollection() is triggered when one block need one new log-reservation block but all the log-reservation blocks have been assigned to other data blocks. changeMappingForCleaningForFullLogReservationBlock() is trigged within cleaningForFullLogReservationBlock() when this data block or over-provisioning block reach their death.