15-746/18-746 Project 1 myFTL Checkpoint 3 Handout: Wear Leveling and Improving FTL Design

September 23, 2016

Contents

1 Checkpoint 3 - Wear Leveling

Checkpoint 3 includes adding wear leveling and open-ended FTL design to your FTL, and it also allows you to re-design your FTL to be better on an overall metric that combines endurance, write amplification, and memory usage.

As explained earlier, SSDs blocks have limited number of erases. In this checkpoint, you may completely redesign the mapping and / or the garbage collection policies you developed in checkpoint 1 and 2 as you incorporate wear leveling and improve the metrics. When an SSD wears out, based on the wear leveling policy used, it may not have used up all the allowed erases for each block. Let us call the writes that non-worn out blocks could have accomodated as "opportunities". One high-level objective of checkpoint 3 is to **minimize lost opportunities**. Similar to segment cleaning, wear leveling is also a heavily policy-driven feature and you are advised to keep your code modular to experiment with multiple policies if need be.

1.1 Guidelines for designing your wear leveling scheme

Following are some of the factors you may want to consider when implementing your wear leveling policy. Note that the factors mentioned below are by no means a complete list. You can choose to wear level using any / none of the following factors, but you must describe your wear leveling scheme in detail in the report and defend your choice of parameters for having done so.

1.1.1 Remaining life of a block

Given the number of erases an SSD block is capable of withstanding, the remaining life of a block is the remaining number of erases that can be performed before we can declare the block dead. In general, the remaining life of an SSD can be considered as the remaining life of its most worn out block. A good wear leveling policy should try and reduce the number of erases being performed on blocks close to their death.

1.1.2 Deviation from average remaining life

Since cleaning policies may influence wear leveling significantly, you could envision wear-based choosing of blocks for the purpose of cleaning. A possible metric to choose a block can be based on the deviation of the remaining life of that block from the average remaining life of the SSD (which is simply the average of the remaining life of all blocks of the SSD). Ideally, we want to try and avoid the SSD declaring itself defunct because a few blocks are significantly more worn out than others, i.e., we want to wear the blocks as evenly as possible.

1.1.3 Blocks containing cold data

One way of classifying data is by temperature. Hot data is data that is frequently or recently written, while cold data is data that has been written relatively long ago. You performed a similar exercise in the cost-benefit segment cleaning policy in checkpoint 2, where you identified the age of a block as the timestamp of the latest page written to that particular block. From the wear leveling point of view, the blocks occupied by cold data are usually the blocks that have seen very little wear. And cold data that will never be deleted or rewritten, "freezes out" opportunities for additional erase/write cycles the LBAs containing that cold data could provide. Choosing to shuffle cold data blocks may be a possible strategy of wear leveling, but one that might complicate the mapping of LBAs to physical addresses.

1.1.4 Overprovisioned space wear leveling

Throughout checkpoint 1 and 2, you have divided the flash capacity space into data blocks, and overprovisioned blocks, subdivided into log-reservation blocks and cleaning-reservation blocks. The overprovisioned space itself can get more worn out than the data blocks. Relocating the overprovisioned blocks to a less worn out area can be a possible wear leveling strategy. Also, within the set of overprovisioned blocks, the log-reservation blocks might wear at different rates than the cleaning-reservation blocks. Shuffling blocks inside the overprovisioned space is another way to reduce wear. Checkpoint 1 ensured that you only needed mapping tables for the overprovisioned space. In the attempt to do sophisticated wear leveling, you should try and avoid introducing huge mapping tables. But you can change the designation of blocks as data, log and cleaning as you see fit.

1.1.5 Selection of segment cleaning policy

Keep in mind that the chosen segment cleaning policy will result in different wear statistics for the SSD. As a result, the blocks chosen by your wear leveling scheme may differ for different segment cleaning policies. You get to choose which cleaning policy to use for checkpoint 3, and you can design a new one, if you want.

1.1.6 TRIM

We introduced TRIM in checkpoint 2 to get you thinking about the ways in which it can be used for checkpoint 3, and we deliberately did not have tests that tried to exploit TRIM in checkpoint 2. But now, for this checkpoint, we will be issing TRIMs along with WRITEs and READs, and it is in your best interest to handle them well. Think about how handling TRIM intelligently can affect the metrics described in the following section.

1.2 Evaluation

1.2.1 Metrics

The evaluation of your FTL's mapping, cleaning and wear leveling policy is going to be broadly based on the following metrics:

- **Write amplification:**
  Write amplification is the ratio of the number of page writes performed by the SSD to the number of LBAs written by a workload. The ideal write amplification is 1. But, since we perform segment cleaning and wear leveling, write amplification is usually >1. We will run various workloads against your SSD to understand the write amplification factor. The closer to 1 it is, the better.
- **Maximum writes observed:**
  Having wear leveling, the SSD should ideally be capable of performing more writes than in checkpoint 1 and 2. We will run various workloads on your SSD until it declares wear our failure and measure the number of writes accepted by the SSD. Usually, the larger the number of writes, the more effective your wear leveling algorithm.
- **Even wear:**
  The block in the entire SSD with the minimum number of erases left is a pretty good indication of the wear in your scheme. Ideally, the block with minimum erases left at the end of a test should have as high a number of remaining erases as possible, to indicate that you have tried to wear your SSD evenly.
- **Memory usage:**
  As mentioned throughout this handout, there is usually very little memory available on an SSD. A small memory footprint is another very important aspect you should keep in mind while designing your wear leveling scheme. Our tests will measure the total amount of memory used by your data structures while running a variety of workloads. In general, the lower the memory usage, the better.

Our tests will combine these metrics into an overall formula which will be an overall measure of your implementation.

1.2.2 Tests

The tests are divided into two categories for checkpoint 3:

- **Finite tests:**
  These tests perform a finite number of operations and end with either a success or a failure. They will also show the memory usage, write amplification, and number of writes sustained.
- **Stress tests:**
  These are essentially infinitely long running tests, i.e. tests that will keep writing data until the SSD can no longer sustain writes. They will try to extract as many writes from your SSD until they can. These tests will also show the memory usage, write amplification, and number of writes sustained.

1.2.3 Code Review

Along with the the above mentioned metrics, we will also be reading your code. Make sure that you describe your scheme in detail mentioning the important points and defend the decisions you have made.

2 Submission

Your code should be submitted to the **myFTL Checkpoint 3** project in Autolab in the same way you submitted your code for checkpoints 1 and 2. You can only modify the *myFTL.cpp* file. When you want to submit the code on Autolab, you should only submit this file.

3 Final Report

After checkpoint 3 is over, you will write a final report explaining your FTL design. We will release a separate writeup detailing our expectations from your report. This will be due 2 days after checkpoint 3 is due.