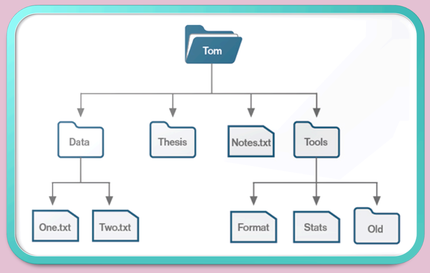
Implementing a log-structured file system for NAND memory

Quan Do ‘19 & Tiffany Zheng ’20
Advisor: Tom Murtagh

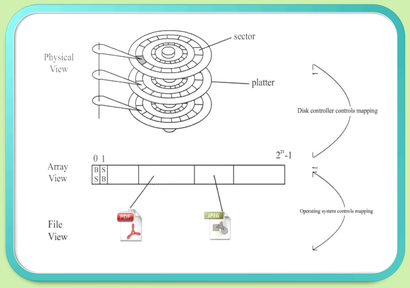
What is a file system?

A file system refers to the methods and structures that manages how and where the operating system stores data in various forms of persistent memory such as disks and NAND memory. It is responsible for keeping track of and retrieving data by storing metadata (information about the data).

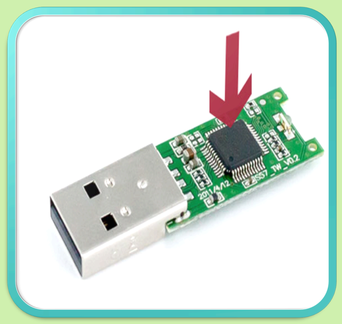


Disk Drives vs. NAND Memory

Hard Disk Drive: For decades, data have been stored into sectors on hard disk drives (HDD). Each sector contains 512 to 2048 bytes worth of data. To access a sector, a magnetic head has to **(1)** move to the right cylinder of the disk *(seek time)* and **(2)** wait for the sector on that track to rotate into position *(rotational delay)*. Data access time is thus determined by these two processes.

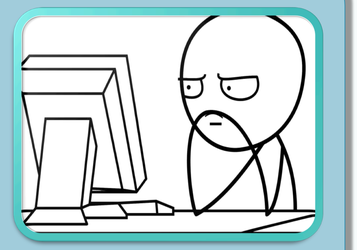


NAND Memory: More recently, NAND-based memory emerged as an alternative to HDD. Pages in NAND memory can be randomly accessed in constant time. This advantage in data access however comes with a few caveats: **(1)** NAND pages have to be erased before being rewritten. To amortize the erase cost, multiple pages are organized into erase blocks, which is the smallest unit one can erase. **(2)** erasing is time-consuming and **(3)** multiple erases gradually wear out NAND cells.



The Problem with the Flash Translation Layer

Because disk drives were more widely used in the past, most file systems are tailored for hard disk drives. NAND memory is accompanied with a Flash Translation Layer (FTL) that simulates the interface of disk drives, thus allowing us to use it without any changes to the operating system. The FTL **(1)** writes updated data to a new empty page and remembers this current address through block mapping, **(2)** ensures that erases are done evenly across blocks so that they wear evenly, and **(3)** keeps track of inactive pages to be erased and recycled later. However, it is somewhat inefficient in that it does not exploit some characteristics of NAND memory to its full potential. Unnecessary features to reduce rotational delay and costs of seeking are still implemented. Additionally, the FTL lacks the ability to access essential information to better reduce block erasure costs.



Log-structured File System (LFS)

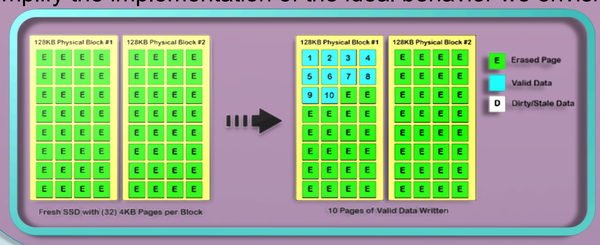
A log-structured file system aims to perform all writes *sequentially* by maintaining a log – an area on a storage device in which new pages are allocated sequentially. Any new update of data or metadata will be written to the end of the log as new copies, instead of written in-place into existing data locations. This greatly reduces write time in HDD.

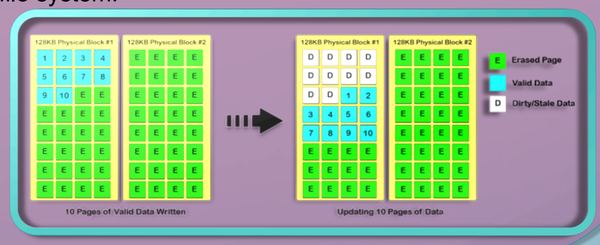
Our Solution

Aim: a file system tailored to the physical properties of NAND memory that **(1)** minimizes number of erases and **(2)** distribute erases throughout all blocks.
This file system is based on principles of log-structured file systems but uses multiple active logs simultaneously so that each log’s contents can be limited to data from a very small group of files. Features include:

- With large files, a log will *exclusively* hold all the data from a *single file* so when we delete the file, we can simply erase all the blocks of the log.
- A log can still be shared between *multiple small files*.
- A list of partially used free blocks is maintained so that we do not recycle a block until it is sufficiently full of obsolete data.

Our focus this summer was developing the foundation and required infrastructure for such a log-structured file system. This includes a library that provides a virtual NAND memory, and a tool to initialize our file system. We also implemented key features to open, create, close, and remove an empty file in such a way that mimics the behavior of NAND. To do so, we utilized FUSE (Filesystem in Userspace) which acts as an intermediary between application programs and our file system without directly modifying the kernel (program that controls the computer’s operating system). By laying out the necessary groundwork, we hope to simplify the implementation of the ideal behavior we envision for our file system.





References

-Arpaci-Dusseau, R. H., & Arpaci-Dusseau, A. C. (2014).*Operating systems, three easy pieces*. Madison, WI: Arpaci-Dusseau Books.
-Rosenblum, M., & Ousterhout, J. K. (1992). The design and implementation of a log-structured file system.*ACM Transactions on Computer Systems,10*(1), 26-52.
-Understanding Flash: The Flash Translation Layer. (2015, September 14). Retrieved August 03, 2017, from https://flashdba.com/2014/09/17/understanding-flash-the-flash-translation-layer/

Acknowledgements

Special thanks to Alexander Majercik ‘17, Austin Paul ‘16, Julia Goldman ’18, and Rehaan Vij ‘18 for their previous work on this project.