

1 Flash-Based SSDs

Vocabularies

1. Flash Solid-State Storage

- Is a type of non-volatile computer storage that stores and retrieves digital information using only electronic circuits, without any involvement of moving mechanical parts

2. NAND-Based Flash

- Is an electronic non-volatile computer memory storage medium using NAND-gate that can be electrically erased and reprogrammed.

3. Flash Page

- Is the smallest unit that can be programmed into flash

4. Flash Block

- Is a group of pages and the smallest unit that can be erased.

Physical Block Addresses											
Block 0						Block 1					
Page n	Page 1	Page 0	Page n	Page 1	Page 0	Page n	Page 1	Page 0	Page n	Page 1	Page 0
Sector n	Sector 1	Sector 0	Sector n	Sector 1	Sector 0	Sector n	Sector 1	Sector 0	Sector n	Sector 1	Sector 0



5. Wear Out

- Is similar to going past **expiration date**
- Means it has exceeded their endurance rating

6. Single-Level Cell

- Is a type of cell in solid-state storage that stores one bit of data per transistor (0 or 1)

7. Multi-Level Cell

- Is a type of cell in solid-state storage that stores two bits of data (i.e 00, 01, 10, 11) per cell using two different levels of charge

8. Triple-Level Cell

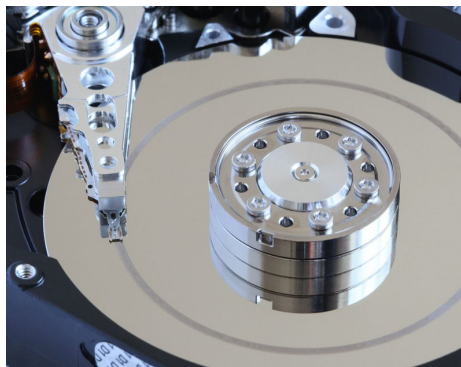
- Is a type of cell in solid-state storage that stores three bits of data per cell (i.e 000, 001, 010, 011, 100, 101, 110, 111)

9. Bank / Plane

- Is a group of large number of cells

10. Head Crash

- Is a condition where the drive head makes contact with the recording surface

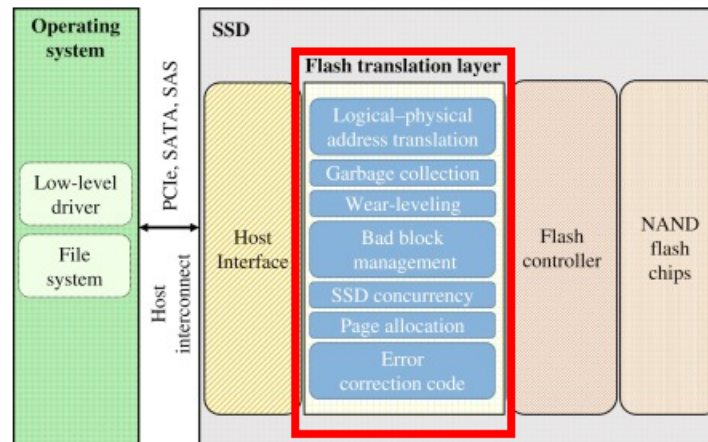


11. Disturbance

- Is also known as **read disturbance** or **program disturbance**
- Is a condition where accessing a bit in a page causes some bits to get flipped in neighboring pages

12. Flash Transition Layer

- Is an intermediate system made up software and hardware that manages SSD operations



13. Wear Leveling

- Is a technique for prolonging the service life of some kinds of erasable computer storage media, such as flash memory, which is used in solid-state drives (SSDs)

14. Direct Mapped

- Is a simplest organization of an **Flash Transition Layer** that maps read of logical page N directly to read of physical page N .

15. Logging

- Is a concept in **log-structured file system** that buffer all writes (data + metadata) using an in-memory segment; once the segment is full, write the segment to a log

16. Mapping Table

- Is a table that stores the physical address of each logical block in the system

17. Logical Block Address

- Is a common scheme used for specifying the location of blocks of data stored on computer storage devices, generally in secondary storage system



18. In-Memory Mapping Table

- Is a table inside the memory of the secondary storage device (is persistent in some form) that stores the physical address of each logical block in the system

19. Garbage Block

- Is also called **Dead Block**
- Is old version of block in secondary storage, such as solid state drive

20. Garbage Collection

- Is the process of finding garbage blocks and reclaiming them for future use

21. Cache Flush

- Is the process of clearing out sections of memory to ensure writes have actually been persisted in solid state drive

22. Trim

- Is an operation that takes an address (and possibly a length) and informs the device that the block(s) specified by the address (and length) have been deleted



23. Overprovision

- Is an extra amount of flash space used to reduce the cost of **garbage collection**, increase the longevity of flash drive, and prevents the device from slowing down



24. Page-Level FTL

- Is an intermediate system made of software and hardware that manages SSD operations at page-level.
 - It does not write a full block
 - Only writes the necessary page(s) of data along with the FTL metadata that must be written to track of the new position of the data

25. Hybrid Mapping

- Is a mapping technique used in **Flash Transition Layer** that utilizes both block-based mapping and page-based mapping to enable flexible writing but also reduce mapping costs

26. Log Blocks

- Are blocks in solid state storage where contents are erased and all writes are directed

27. Switch Merge

- This will be revisited when reading related section

28. Partial Merge

- This will be revisited when reading related section

29. Full Merge

- This will be revisited when reading related section

1.1 Flash-Based SSDs

- Has two interesting problems to overcome
 1. To write a small chunk (called **flash page**), a bigger chunk (**flash block**) must be erased first
 2. Writing too often would cause a page to **wear out**

1.2 Storing a Single Bit

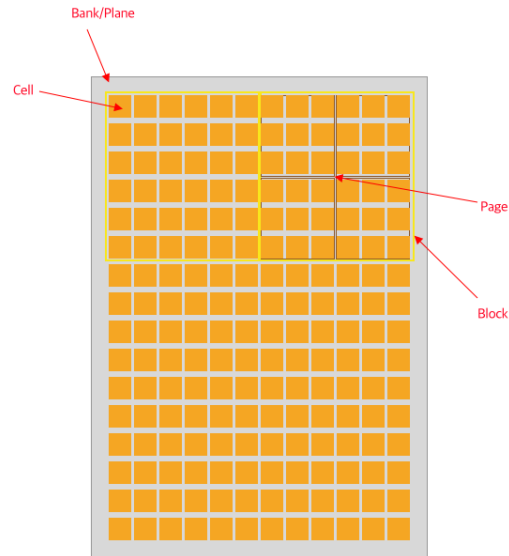
- **Single-level cell** → 1 bit per cell
- **Multi-level cell** → 2 bits per cell
- **Triple-level cell** → 3 bits per cell
- **Single-level cell** has higher performance and are more expensive
 - More 촌촌하다
- How SLC, MLC, TLC works → Physics!!

1.3 From Bits to Banks / Planes

Question Is content in a flash chip a cell? How many bits can be stored per content?

Question I should ask clarification from professor about why Samsung and other tech giants are producing higher level cells if SLC is better in performance

- In each plane/bank, there are large number of blocks
- In each block, there are a large number of pages



1.4 Basic Flash Operations

- **Read (a page):**
 - Is fast ($10 \mu s$)
 - Can access any location uniformly
 - * flash-based SSD is a **random access device**
- **Erase (a block):**
 - Is most expensive
 - **block** must be erased before erasing a **page**



- **Program (a page):**
 - Is used to change some of the 1's within a page to 0's and vice versa
 - Is less expensive than erasing a block
 - Is more costly than reading a page
 - Before overwriting any page within a block, we must first move any data we care about to another location
 - * Frequent repetitions of program/erase cycle cause flash chips to **wear out**
- Page starts in INVALID state
- Erasing block results in pages with ERASED state
 - resets contents in page
 - makes contents re-programmable
- Programming a page results in VALID state
 - Contents are set and can be read
 - Only way to change it's content is to erase the entire block

	iiii	Initial: pages in block are invalid (i)
Erase()	→ EEEE	State of pages in block set to erased (E)
Program(0)	→ VEEE	Program page 0; state set to valid (v)
Program(0)	→ error	Cannot re-program page after programming
Program(1)	→ VVEE	Program page 1
Erase()	→ EEEE	Contents erased; all pages programmable

1.5 Flash Performance and Reliability

- Performance

	Device	Read (μ s)	Program (μ s)	Erase (μ s)
Single-layer cell	SLC	25	200-300	1500-2000
Multi-layer cell	MLC	50	600-900	~3000
Triple-layer cell	TLC	~75	~900-1350	~4500

Slower
as more bits
packed per
cell

- Reliability Concerns

- Wear Out

- * Accrues a bit of extra charge when flash block is erased and programmed
- * Over time, 0 and 1 become difficult to distinguish
- * Becomes unusable at the point where it becomes impossible to distinguish

- Disturbance

- * Access a page within a flash may cause some bits to get flipped in neighboring pages
- * **Disturbance** while programming → **Program Disturbance**
- * **Disturbance** while reading → **Read Disturbance**

1.6 From Raw Flash to Flash-Based SSDs

- Addresses the question "How to turn a basic set of flash chips into something that looks like a typical storage device"

- Hardware Requirements

- Flash chips
- Some volatile memory for caching (e.g. SRAM)
- Control logic to orchestrate device operation

- Other Requirements

- Flash Transition Layer

- * Is made up of both hardware and software
- * Takes read and write requests on logical blocks
- * Turns contents in logical blocks into low-level read, erase, and program commands on the underlying *physical page* and *physical block*
- * FTL should accomplish this task with the goal of excellent performance and high reliability

1. Performance

- Running in **parallel** - using multiple chips internally boosts performance

Is similar to using multiple disk arm on HDD

2. Reliability

- **Wear Leveling** - Spreading writes across the blocks of flash as evenly as possible
 - Address **wear out**
 - Slows the buildup of charge
- Writing page in order from low page to high page
 - Minimize **disturbance**

1.7 FTL Organization: A Bad Approach

- **Direct Mapped**

- How it works
 - * Read
 - Directly translates the read of logical page N to a read of physical page N
 - * Write
 - Read in the entire block page N is contained within
 - Erase the block
 - Program the old page as well as the new one
- Creates **write amplification** (performance problem)
 - * Write is proportional to number of pages in a block
 - Reading entire block is costly, Erasing it is costly, and programming it is costly
 - * Results in terrible write performance
 - * Is slower than typical hard drives
- Doesn't follow **wear leveling** (reliability problem)
 - * User file data is repeatedly over written
 - * Results in **wear out** and potentially loses data

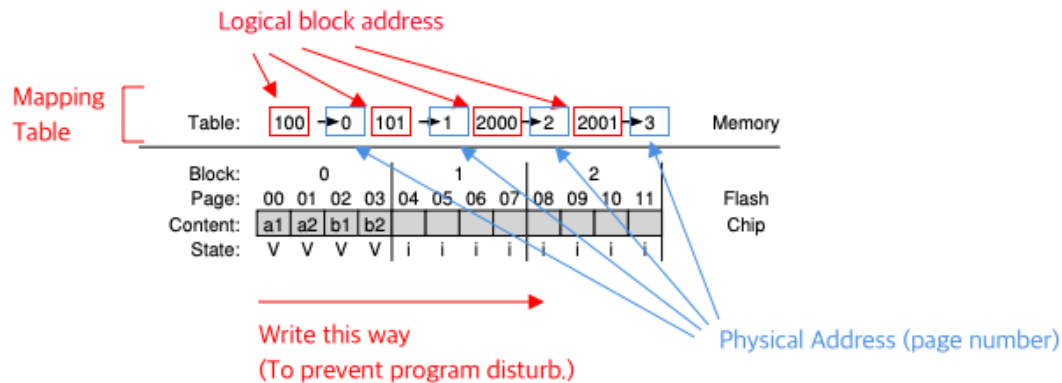
1.8 A Log Structured FTL

- Works similarly as **log structured file system**
- Is useful in storage device and file system
- How it works

- Keeps a **logical block** N
 - * To append the write to the next free spot in the currently-being-written block
- Keeps a **mapping table**
- Translate contents from logical address to physical address
 - * Is done when logical block is full

Example

- Write logical address (100) with content a1
- Write logical address (101) with content a2
- Write logical address (2000) with content b1
- Write logical address (2001) with content b2



- Advantages
 - Improves performance
 - Greatly enhances reliability

1.9 Garbage Collection

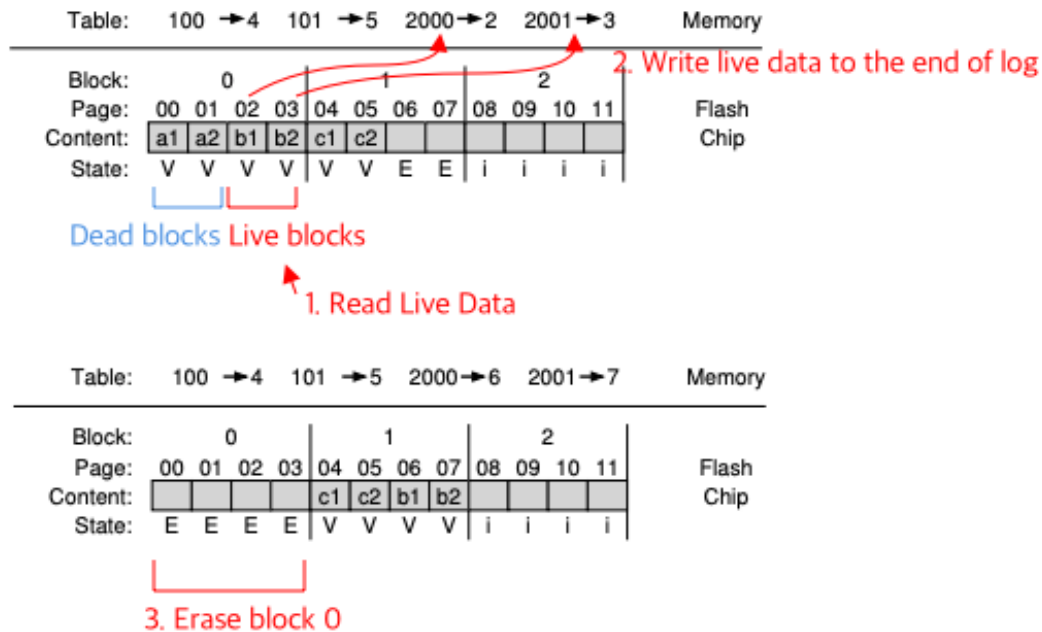
Note Mapping not on **mapping table** is considered a garbage

Question I should ask professor if durability is the issue, why can't we clean the build of charge by moving block to overprovisioned space and then discharging the block?

- How it works
 - Find a block with one or more garbage pages
 - Read in the live (non-garbage) pages from the block
 - Write out those live pages to the log

- Reclaim the entire block for use in writing

Example



- Disadvantage
 - Is expensive
 - * Reading (25 - 75 μ s / operation) + Re-writing live blocks (200 - 1360 μ s / operation)
 - * Cost is reduced via **overprovision**
 - Moves to-be-deleted items to this background space
 - **Garbage Collection** is done when the device is less busy

1.10 Mapping Table Size

- Addresses the second problem of log structuring - a huge mapping table
 - 4 TB of space with 4KB page with 4 byte per entry requires 1 GB of memory
 - Page-level FTL is impractical
- Solutions
 1. **Block-based mapping**
 - Has a pointer per block (not one pointer per page)
 - Reduces mapping amount

- Does not work very well

1. Small write

- * Must read a large amount of live data from old block before moving to a new one
- * Increases **write amplification**
- * Decreases performance

2. Hybrid mapping

- Is the most used mapping method today
- Enables flexible writing but reduces mapping costs
- How it works

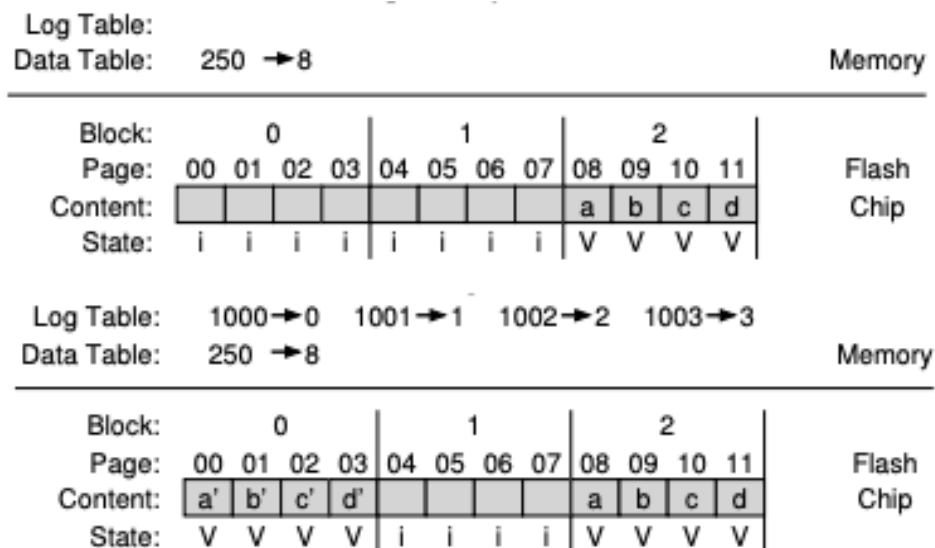
- * Keep pointers for both page and block
 - Has small set of per-page mapping in **log table**

Note Small is the key to hybrid mapping

- Has larger set of per-block mapping in **data table**
- * Periodically examine **log blocks**
- * When sufficient, switch them into blocks that can be pointed to by only a single block pointer

Example

- Contents of the writes to 1000,1001,1002,1003 are a, b, c, d respectively
- Client overwrites the contents a, b, c, d to a', b', c', d'



Log Table:													
Data Table:		250 → 0											
Block:		0				1				2			
Page:		00	01	02	03	04	05	06	07	08	09	10	11
Content:		a'	b'	c'	d'								
State:		V	V	V	V	i	i	i	i	i	i	i	i

Flash
Chip

1.11 Wear Leveling

1.12 SSD Performance And Cost