Jin-Soo Kim
(*jinsoo.kim@snu.ac.kr*)

Systems Software &
Architecture Lab.

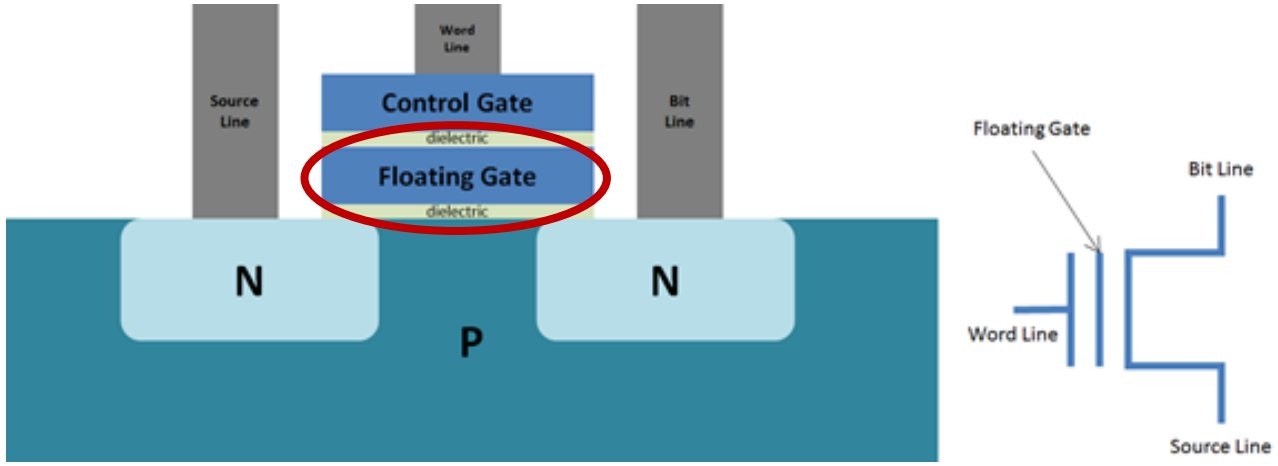
Seoul National University

Fall 2024

# Solid State Drives (SSDs)

# Flash Memory Cell

- Transistor with floating gate
  - The floating gate is insulated all around with an oxide layer
  - Electrons trapped in the floating gate can remain for up to years



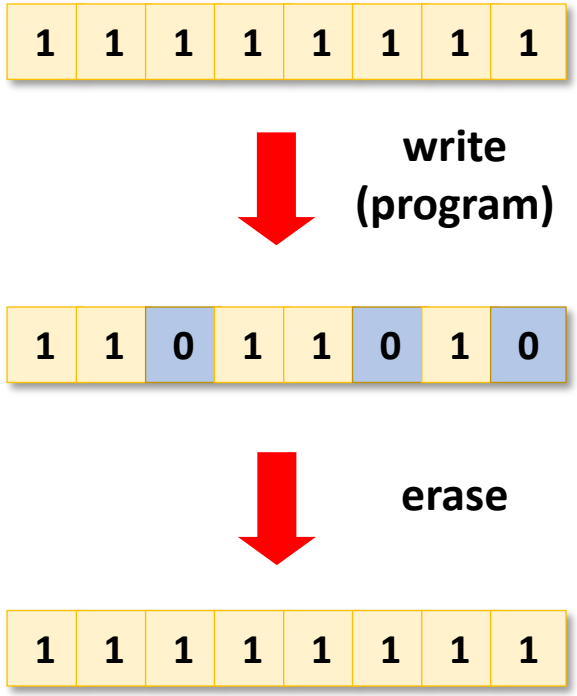



http://www.thenandflash.com

# Flash Memory Characteristics

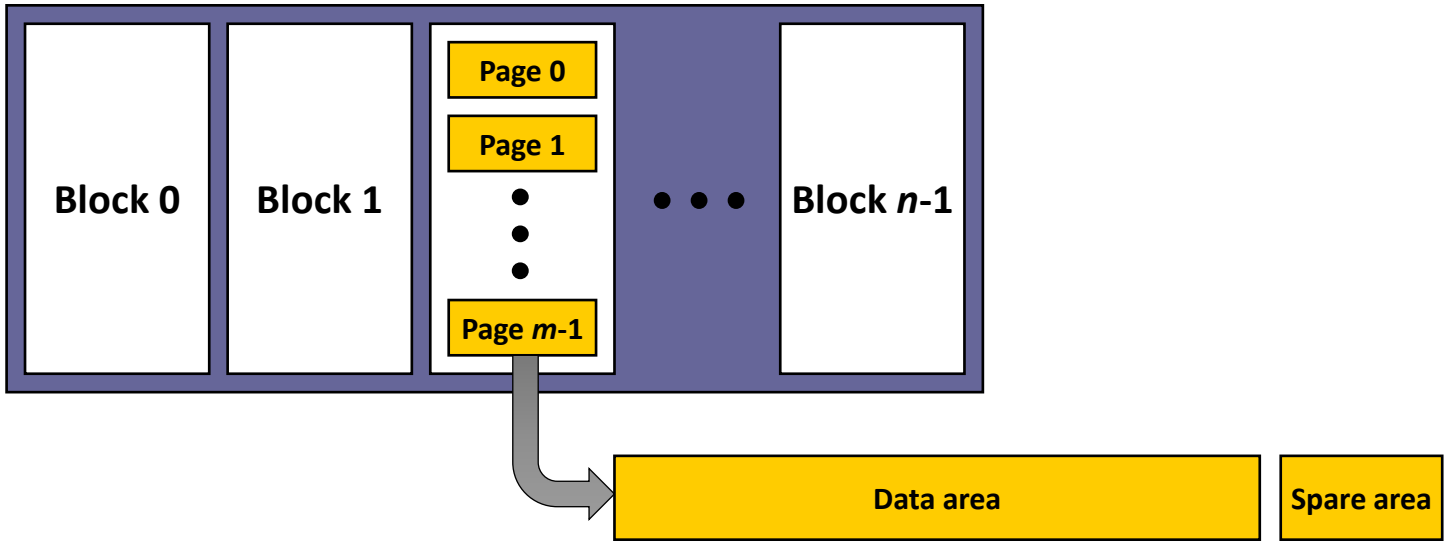
- Erase-before-write
  - Read
  - Write or Program: 1 → 0
  - Erase: 0 → 1

- Bulk erase
  - Program unit:
    - NOR: byte or word
    - NAND: page
  - Erase unit: _____



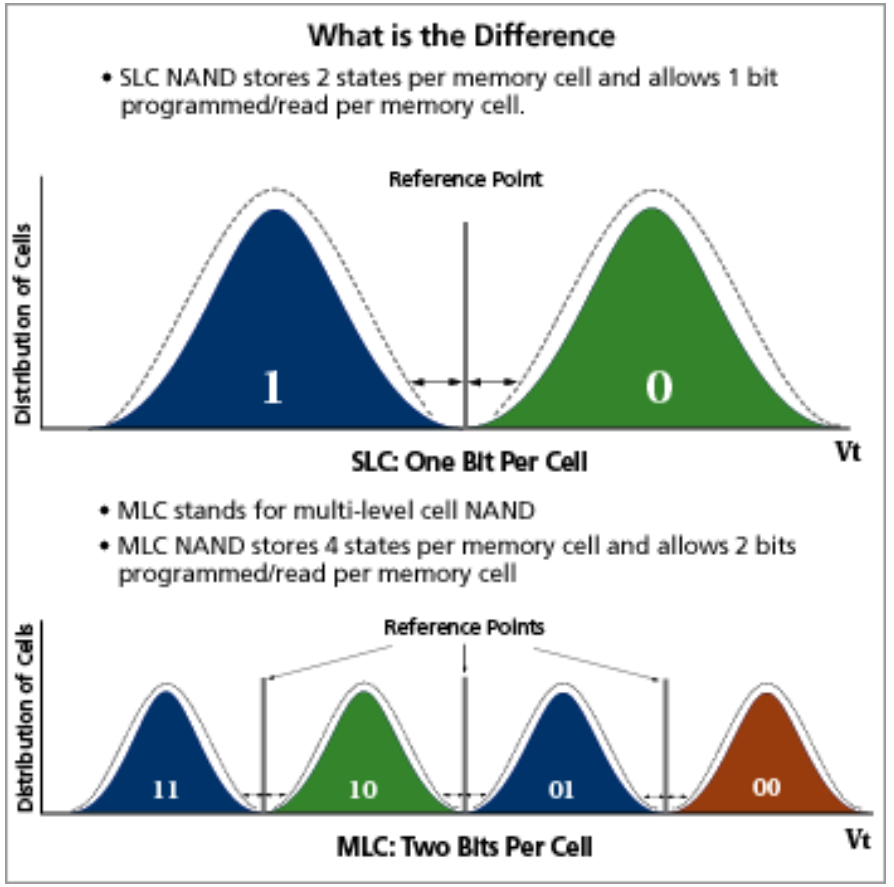
# Logical View of NAND Flash

- A collection of blocks
- Each block has a number of pages
- The size of a block or a page depends on the technology (but, it's getting larger)



# NAND Flash Types

- SLC NAND
  - Single Level Cell
  - 1 bit/cell
- MLC NAND
  - Multi Level Cell (misnomer)
  - 2 bits/cell
- TLC NAND
  - Triple Level Cell
  - 3 bits/cell
- 3D NAND



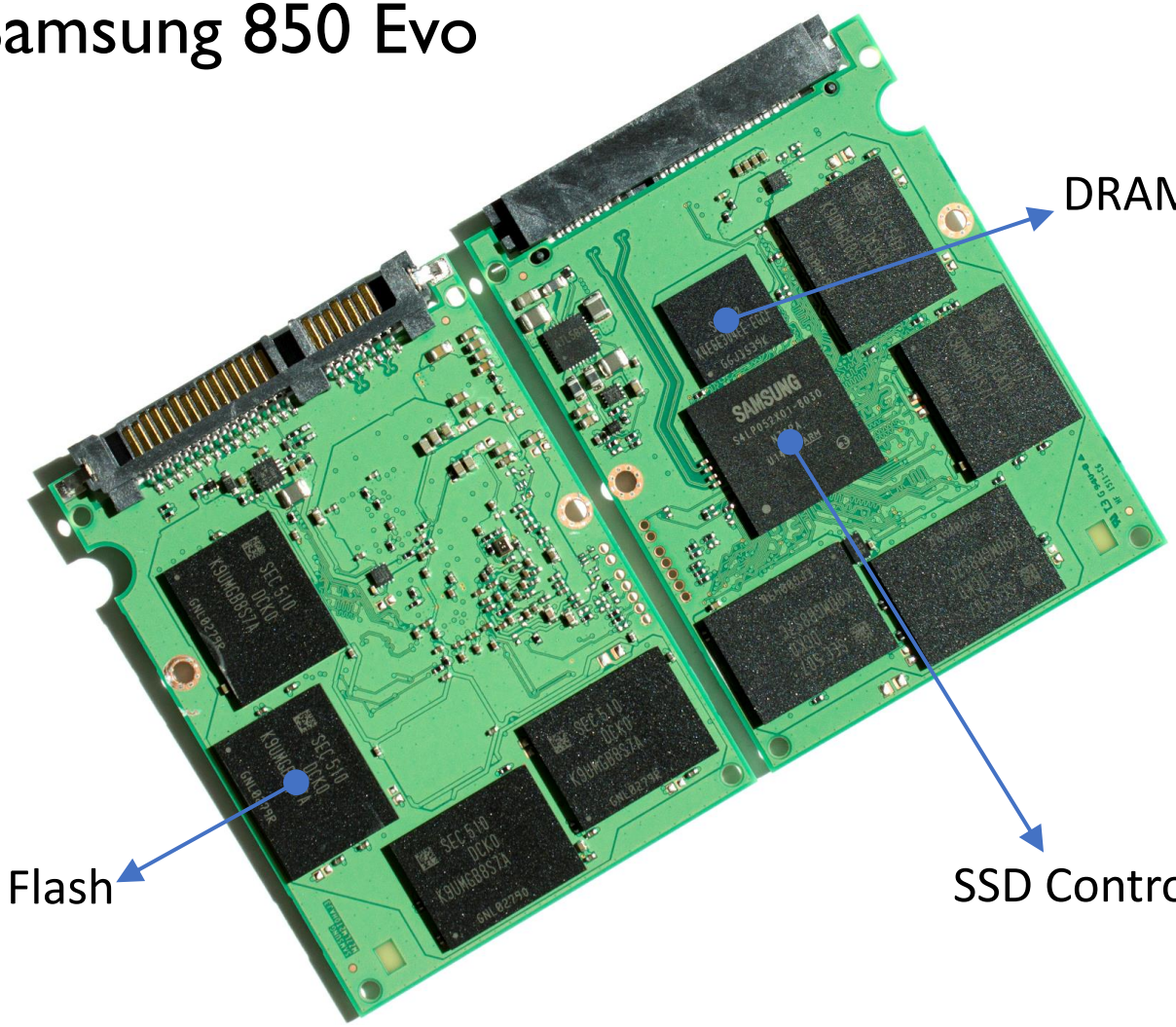

*Source: Micron Technology, Inc.*

# NAND Applications

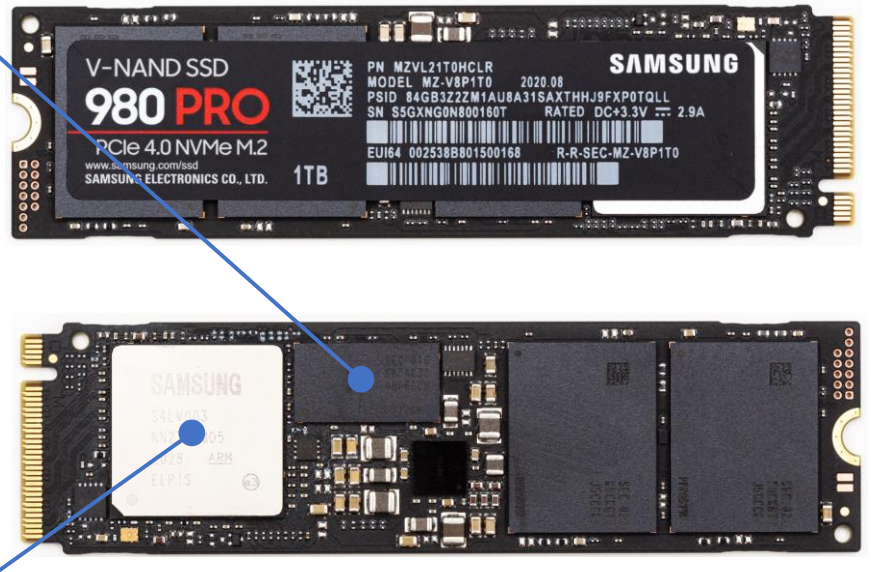
- Universal Flash Drives (UFDs)
- Flash cards
  - CompactFlash, MMC, SD, Memory stick, …
- Smartphones
  - eMMC (Embedded MMC)
  - UFS (Universal Flash Storage)
- SSDs (Solid State Drives)
- Other embedded devices
  - MP3 players, Digital TVs, Set-top boxes, Car navigators, …


# Anatomy of an SSD

- Samsung 850 Evo

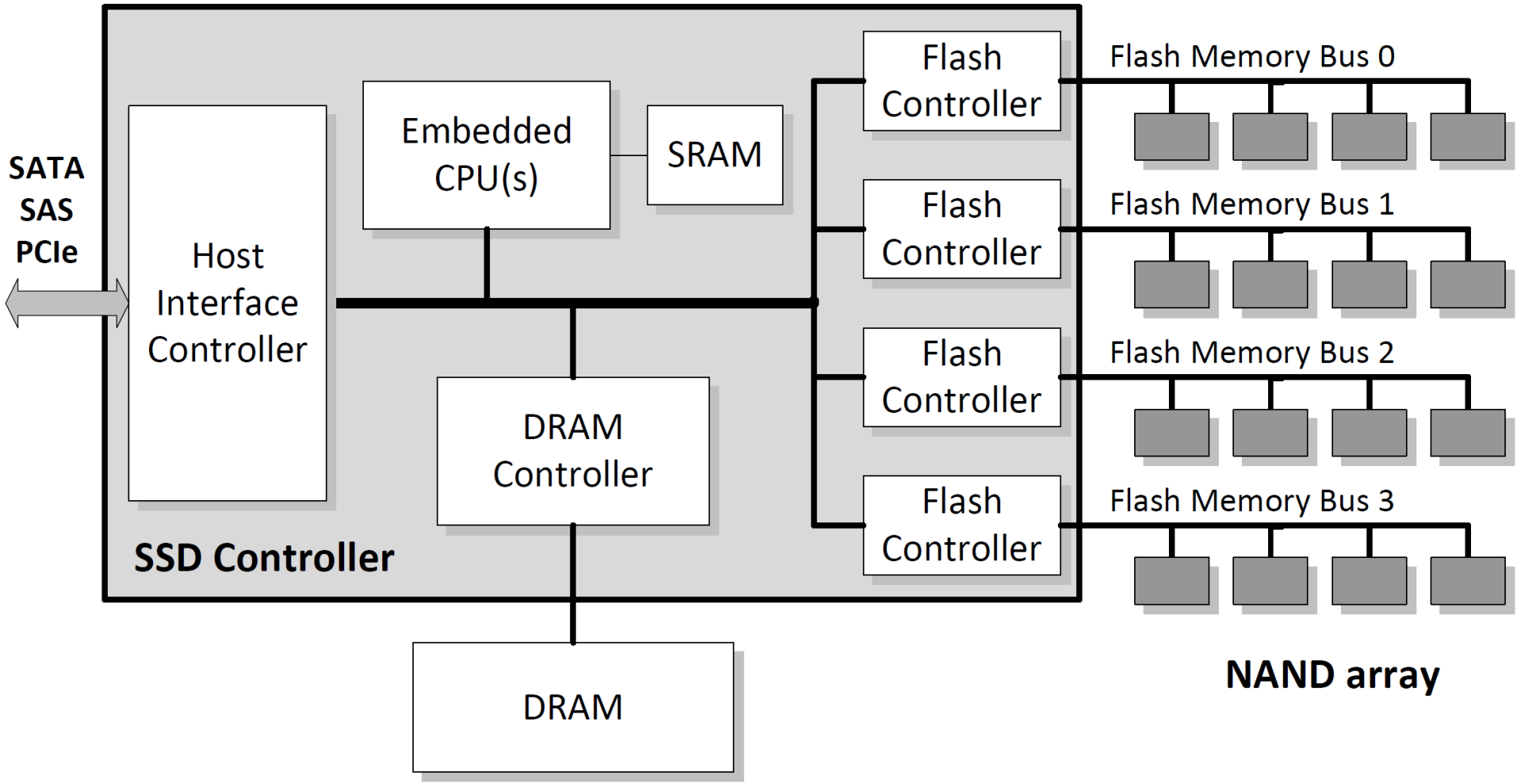



- Samsung 980 Pro



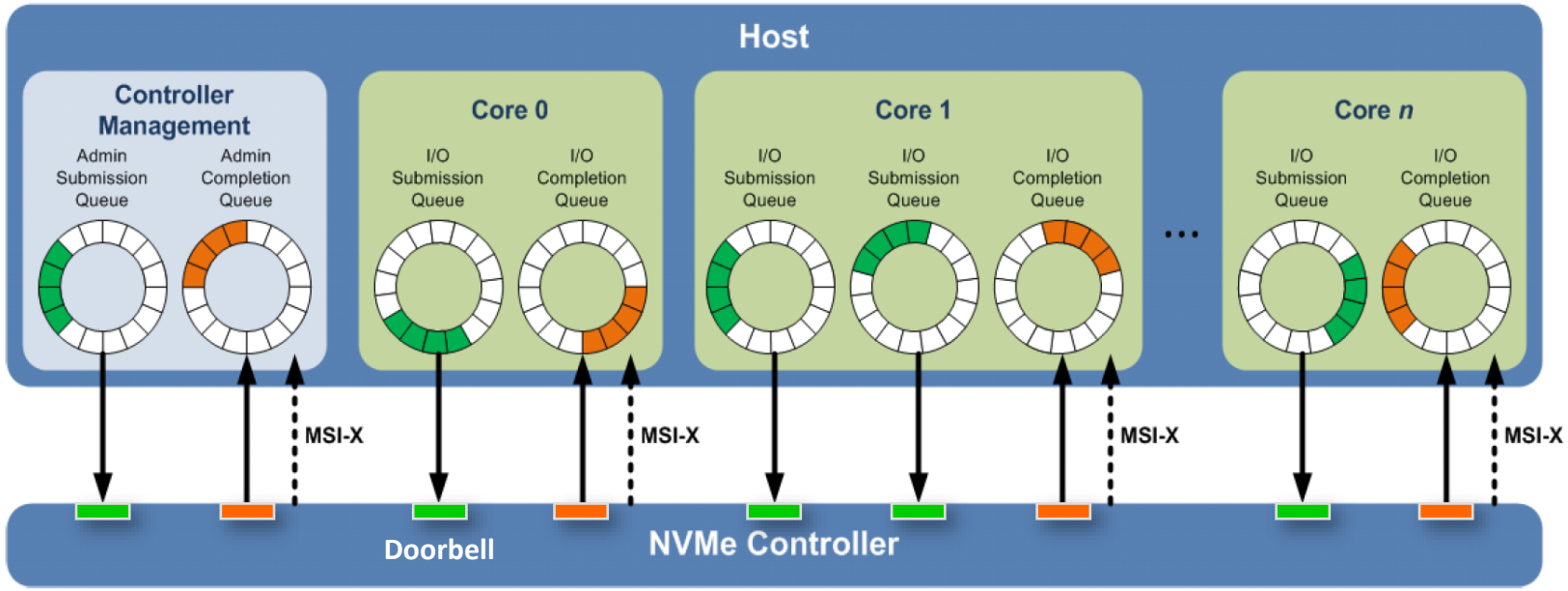

*http://www.anandtech.com/show/9451/the-2tb-samsung-850-pro-evo-ssd-review*
*https://www.tomshardware.com/reviews/samsung-980-pro-m-2-nvme-ssd-review*

# SSD Internals



# NVMe SSD

- PCIe-based (PCIe Gen. 3: 1GB/s per lane, up to 32 lanes)
- Deep queue: 64K commands per queue, up to 64K queues
- Streamlined command set: only 13 required commands
- One register write to issue a command ("doorbell")



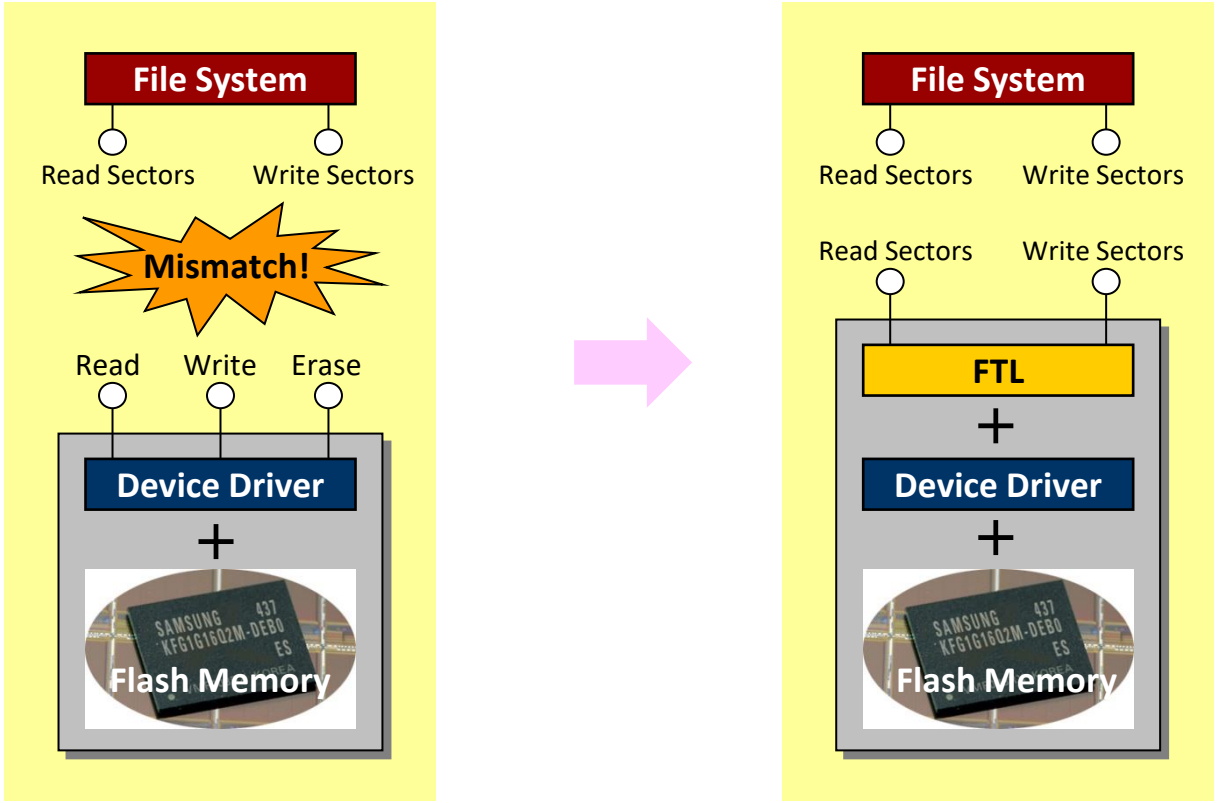
# HDDs vs. SSDs

| Feature | SSD (Samsung) | HDD (Seagate) |
|---|---|---|
| Model | MZ-V8P2T0 (980 Pro) | ST2000LM003 (SpinPoint M9T) |
| Capacity | 2TB<br>(512Gb 128-Layer 3D V-NAND TLC<br>x 16 dies/chip x 2 chips) | 2TB<br>(3 Discs, 6 Heads, 5400 RPM) |
| Form factor | M.2 (2280), 55g | 2.5”, 130g |
| DRAM | 2 GB | 32 MB |
| Host interface | NVMe (PCIe 4.0 x 4, 8GB/s) | SATA-3 (600 MB/s) |
| Power consumption<br>(Active / Idle / Sleep) | 6.1 W/ 0.035 W / 0.005 W | 2.3 W / 0.7 W / 0.18 W |
| Performance | Sequential read: 7000 MB/s<br>Sequential write: 5100 MB/s<br>Random read: 1,000K IOPS (QD32)<br>Random write: 1,000K IOPS (QD32)<br><br>Random read: 22,000 IOPS (QD1)<br>Random write: 60,000 IOPS (QD1) | Sequential read: 124 MB/s<br>Sequential write: 124 MB/s<br>Random read: 56 IOPS<br>Random write: 98 IOPS<br><br>Power-on to ready: 3.5 sec<br>Average seek: 12/14 ms<br>Average latency: 5.6 ms |
| Price[1] | 243,050 won (123won/GB) | 66,000 won (33won/GB) |

[1] *http://www.danawa.com (As of Nov. 29, 2023)*

# Flash Translation Layer (FTL)

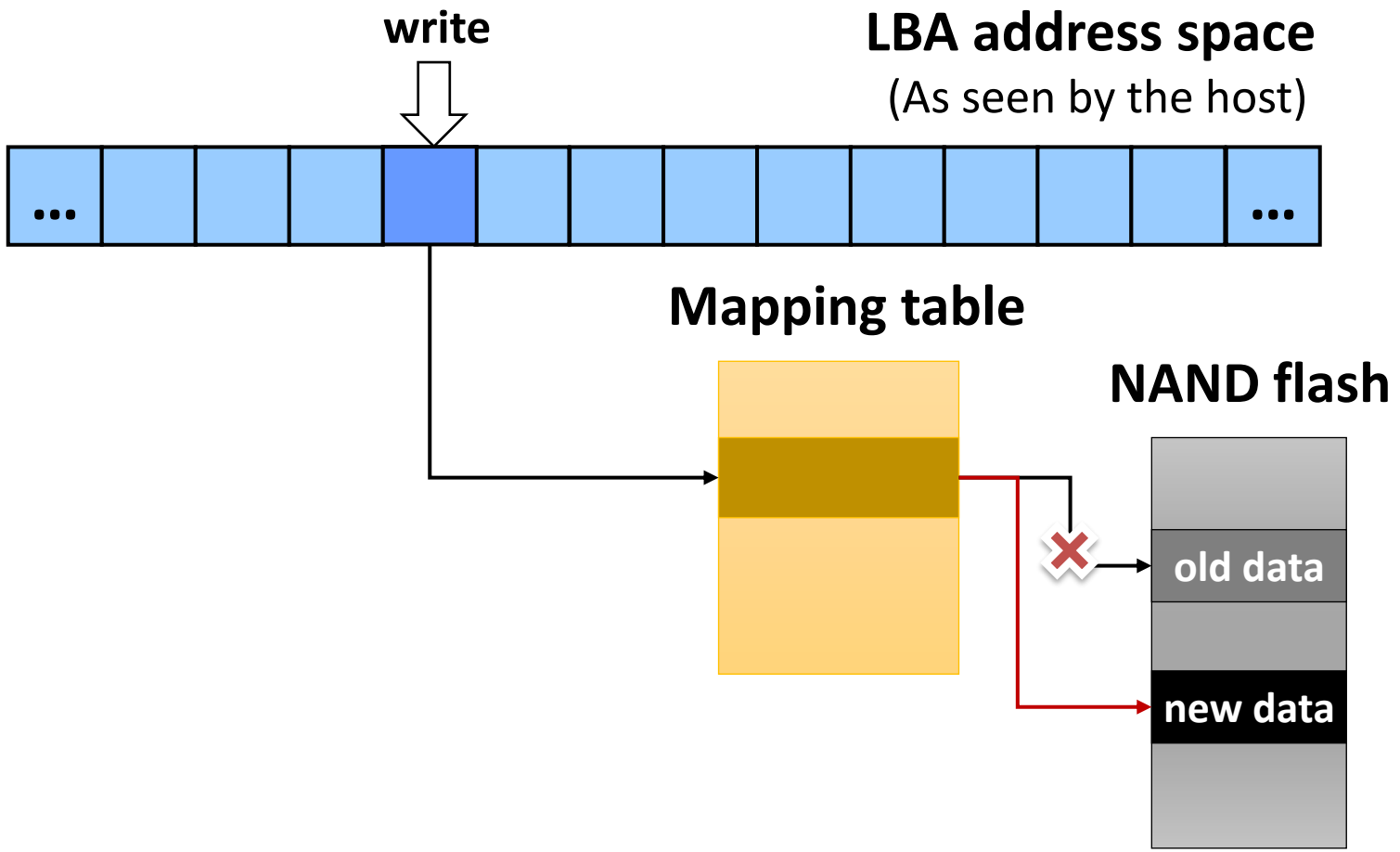
- A software layer to make NAND flash fully emulate traditional block devices (e.g., disks)




Source: Zeen Info. Tech.

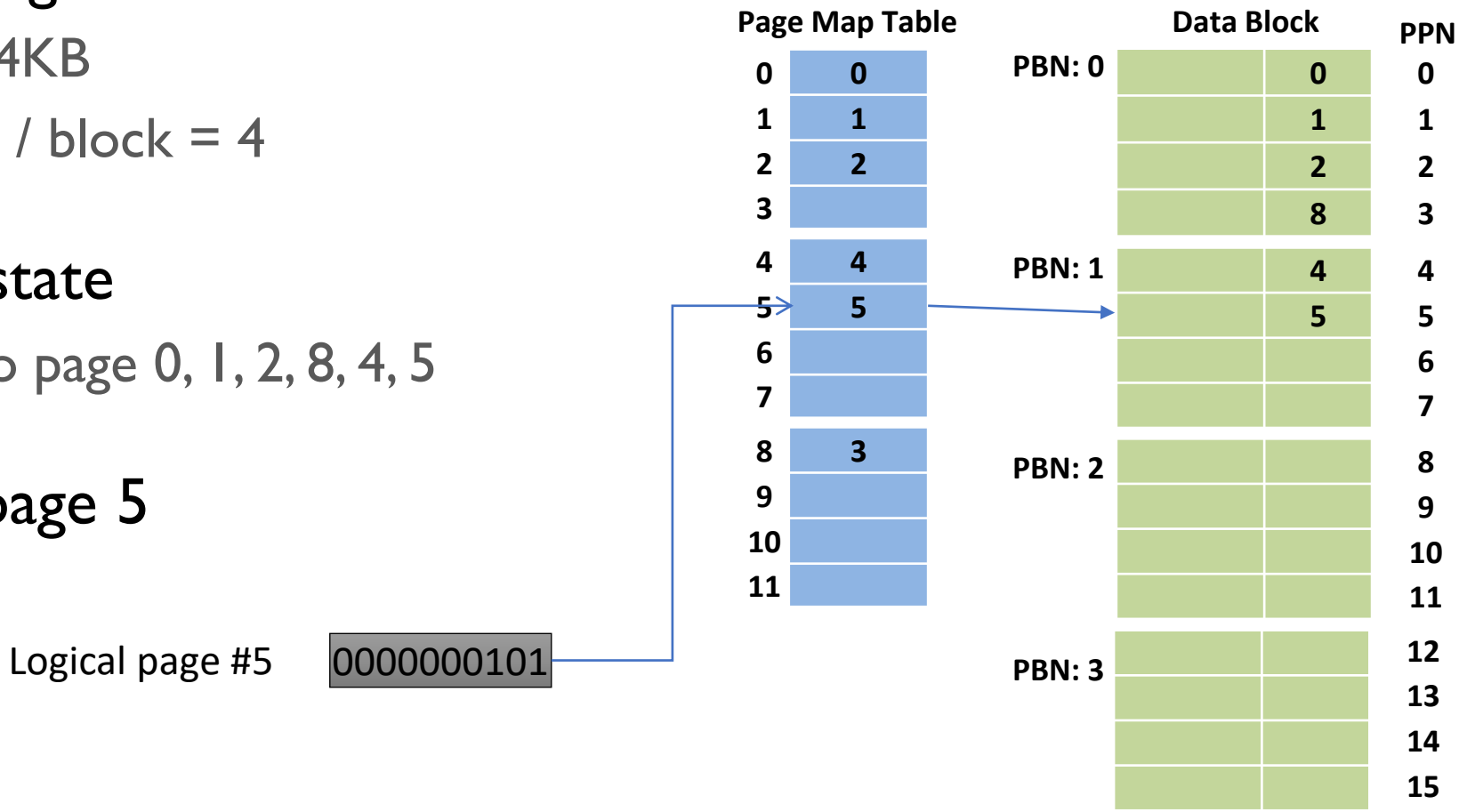
# Address Mapping

- Required since flash pages cannot be overwritten



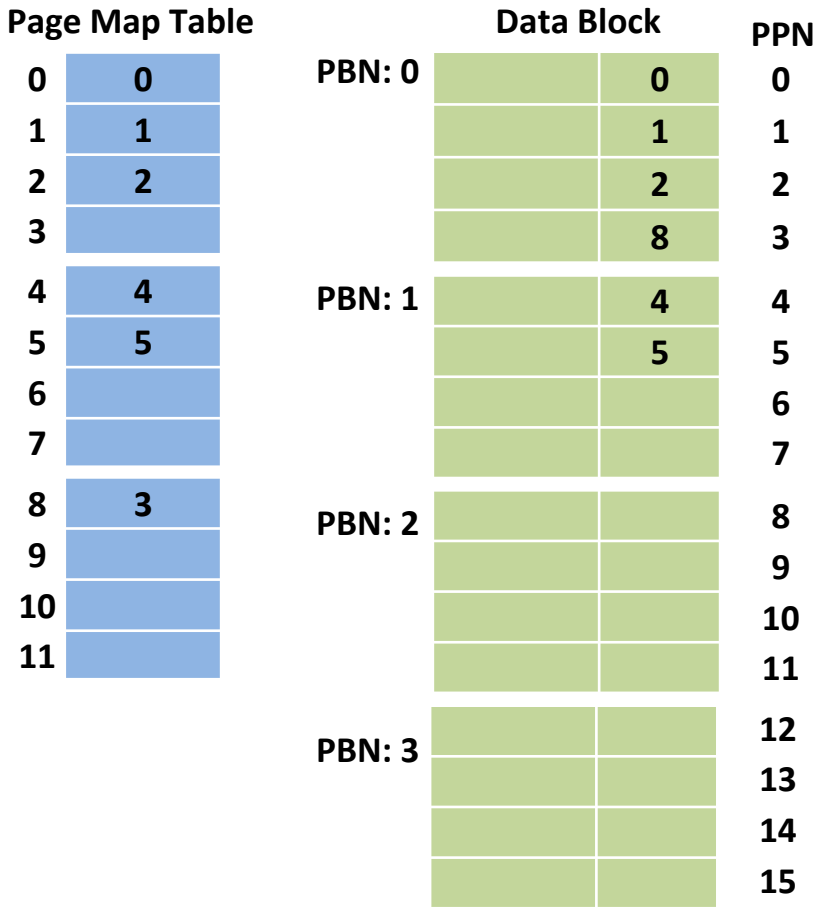
# Example: Page Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
- Current state
  - Written to page 0, 1, 2, 8, 4, 5
- Reading page 5



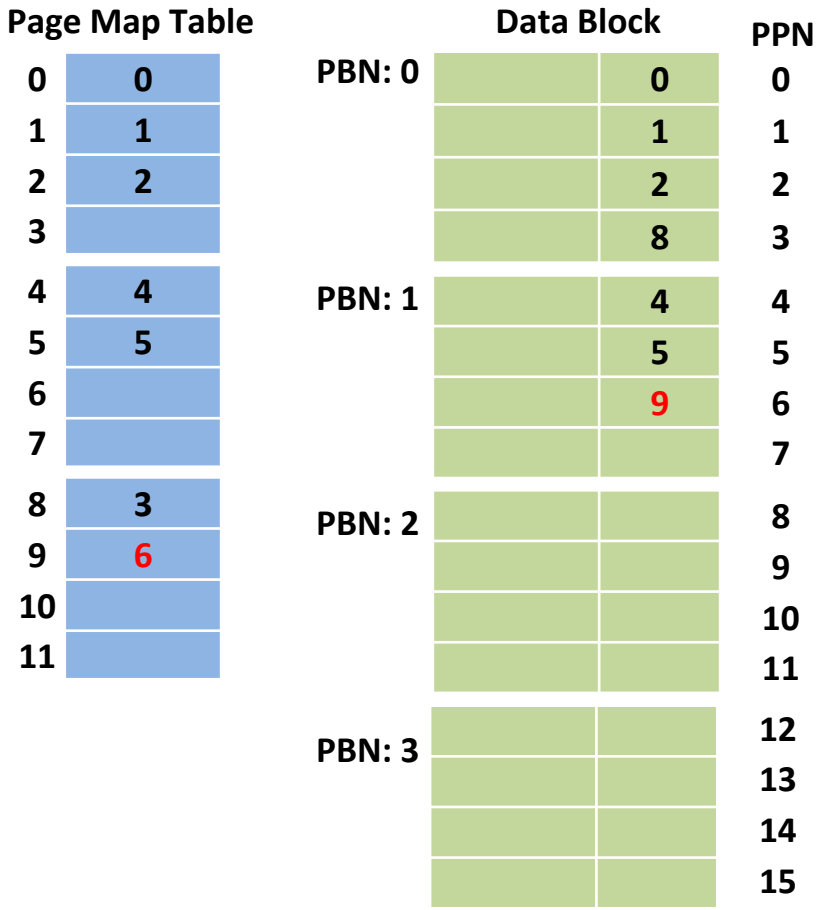
# Example: Page Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
- Current state
  - Written to page 0, 1, 2, 8, 4, 5
- New requests (in order)
  - Write to page 9
  - Write to page 3
  - Write to page 5



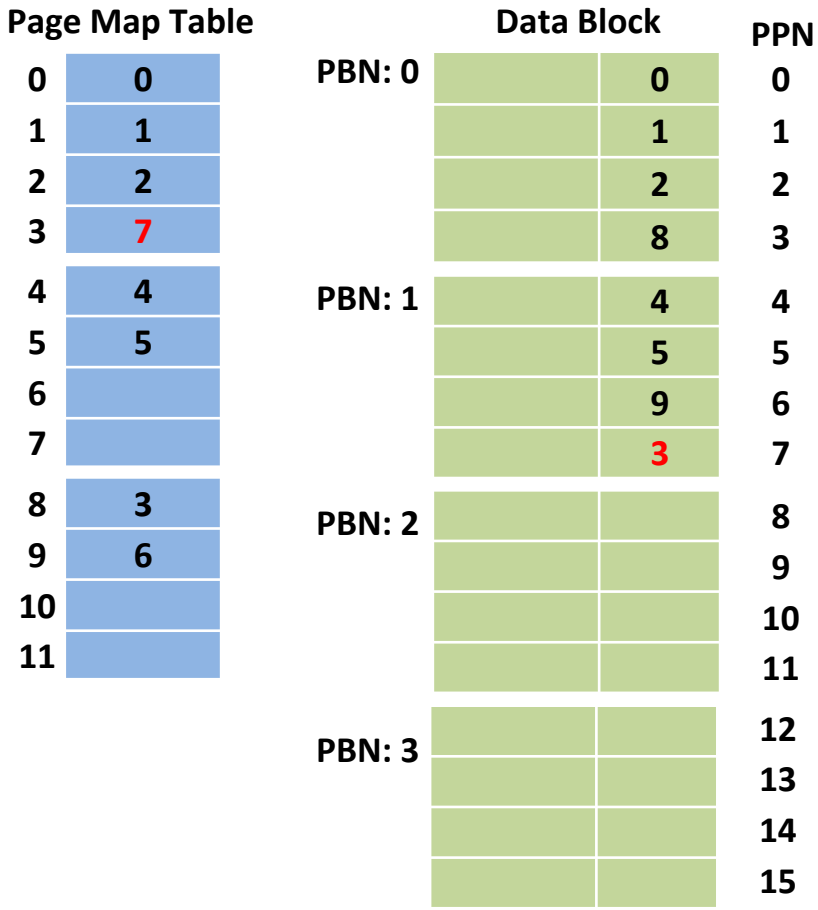
# Example: Page Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
- Current state
  - Written to page 0, 1, 2, 8, 4, 5
- New requests (in order)
  - Write to page 9
  - Write to page 3
  - Write to page 5



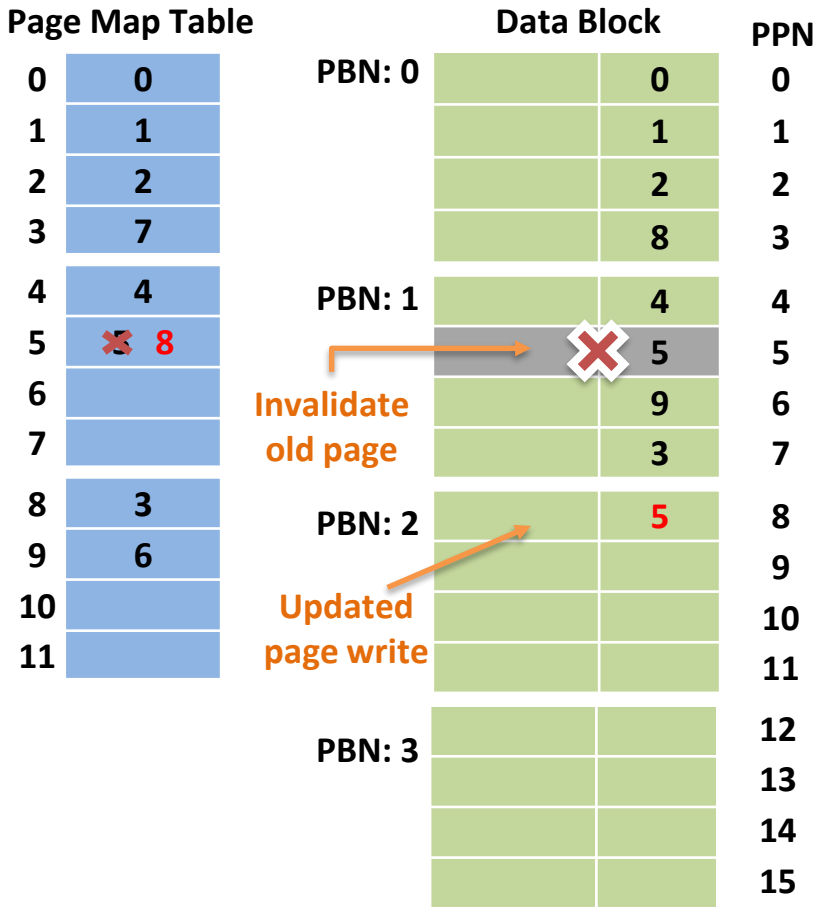
# Example: Page Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
- Current state
  - Written to page 0, 1, 2, 8, 4, 5
- New requests (in order)
  - Write to page 9
  - Write to page 3
  - Write to page 5



# Example: Page Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
- Current state
  - Written to page 0, 1, 2, 8, 4, 5
- New requests (in order)
  - Write to page 9
  - Write to page 3
  - Write to page 5



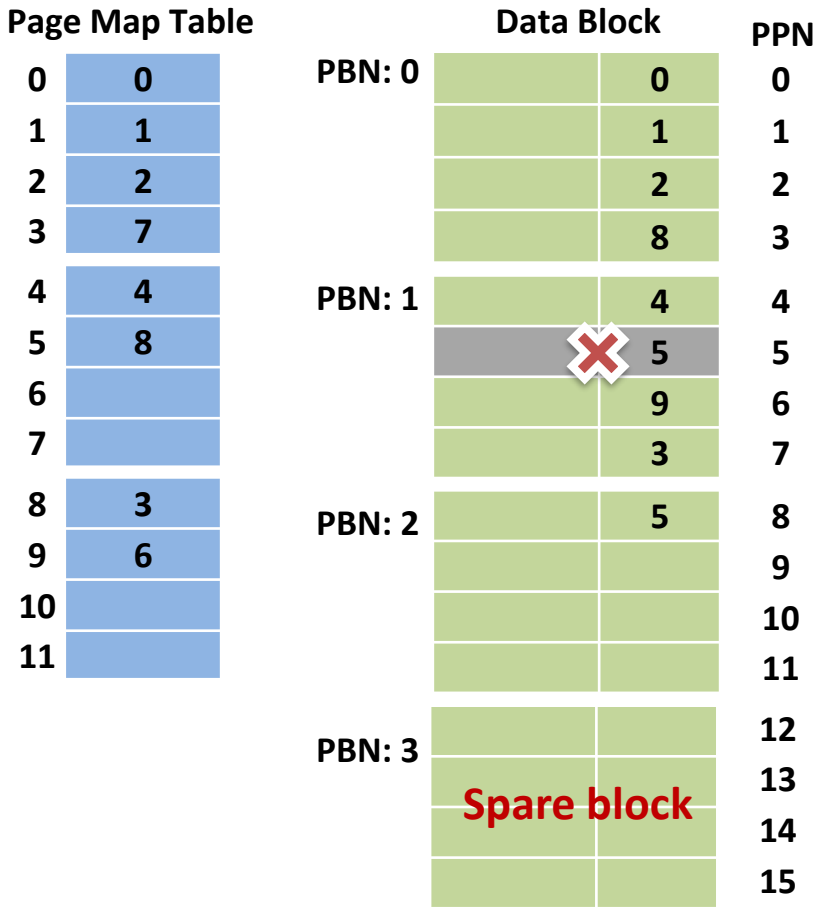
# Garbage Collection

- Garbage collection (GC)
  - Eventually, FTL will run out of blocks to write to
  - GC must be performed to reclaim free space
  - Actual GC procedure depends on the mapping scheme

- GC in page-mapping FTL
  - Select victim block(s)
  - Copy all valid pages of victim block(s) to free block
  - Erase victim block(s)
  - Note: At least one free block should be reserved for GC

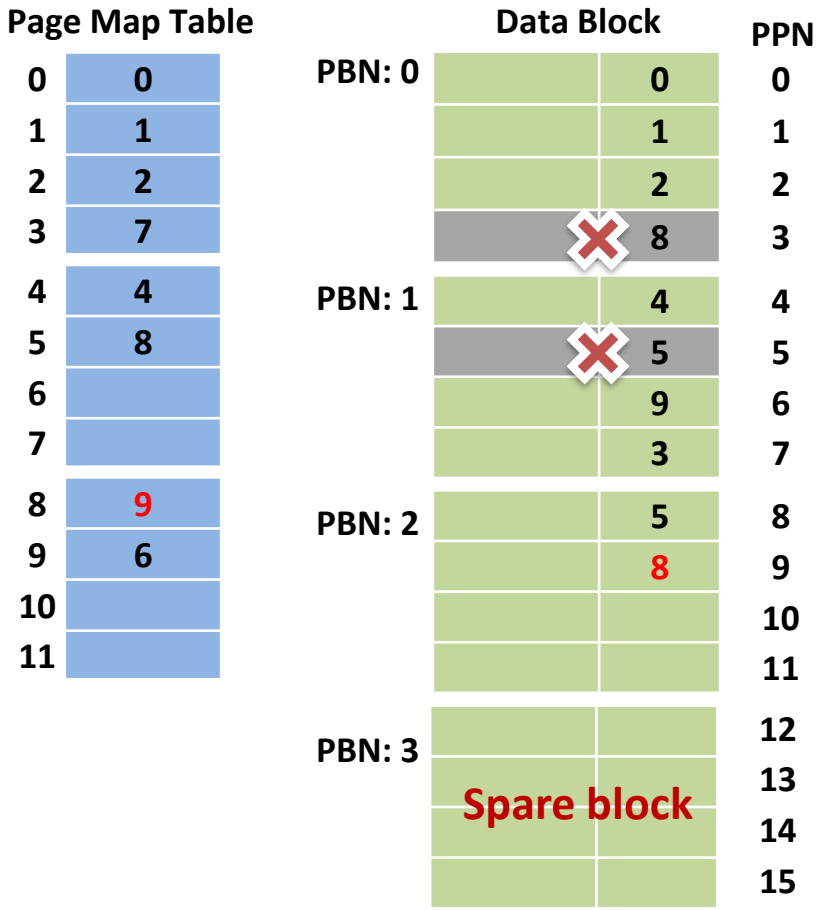
# Example: GC in Page Mapping

- Current state
  - Written to page 0, 1, 2, 8, 4, 5
  - Written to page 9, 3, 5
- New requests (in order)
  - Write to page 8
  - Write to page 9
  - Write to page 3
  - Write to page 1
  - Write to page 4



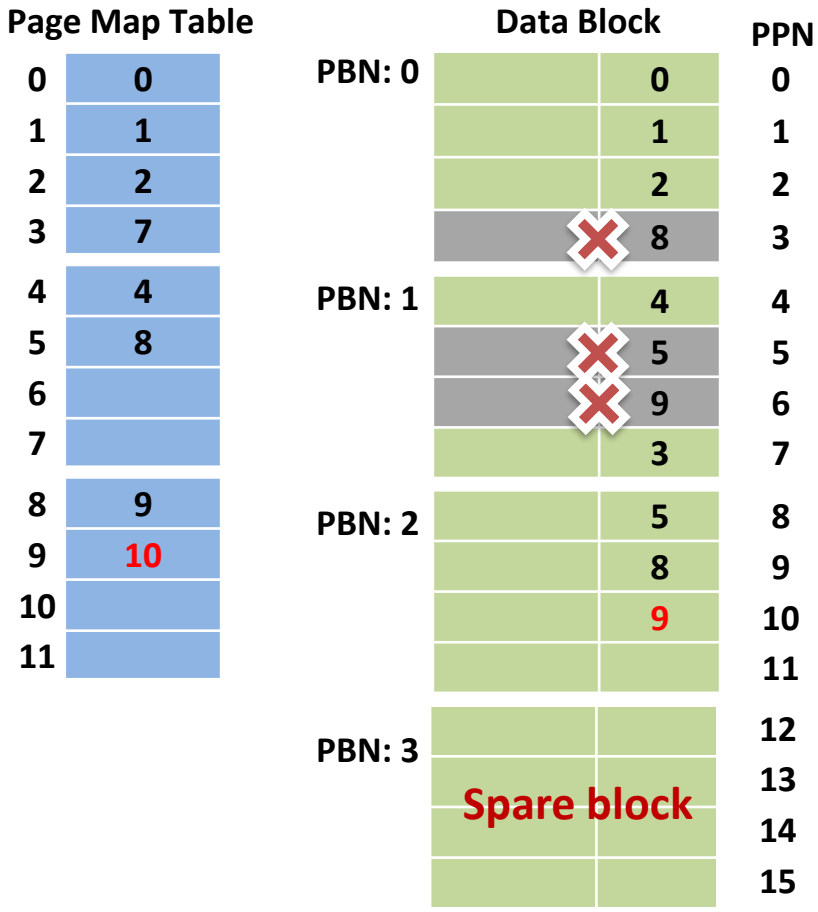
# Example: GC in Page Mapping

- Current state
  - Written to page 0, 1, 2, 8, 4, 5
  - Written to page 9, 3, 5
- New requests (in order)
  - Write to page 8
  - Write to page 9
  - Write to page 3
  - Write to page 1
  - Write to page 4



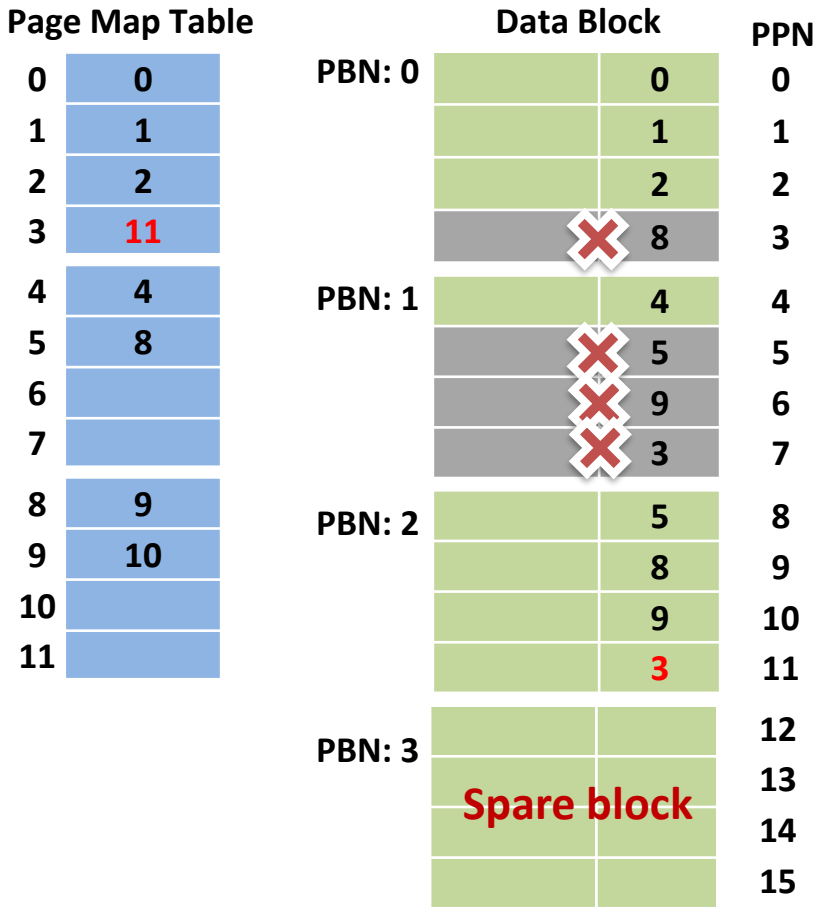
# Example: GC in Page Mapping

- Current state
  - Written to page 0, 1, 2, 8, 4, 5
  - Written to page 9, 3, 5
- New requests (in order)
  - Write to page 8
  - Write to page 9
  - Write to page 3
  - Write to page 1
  - Write to page 4



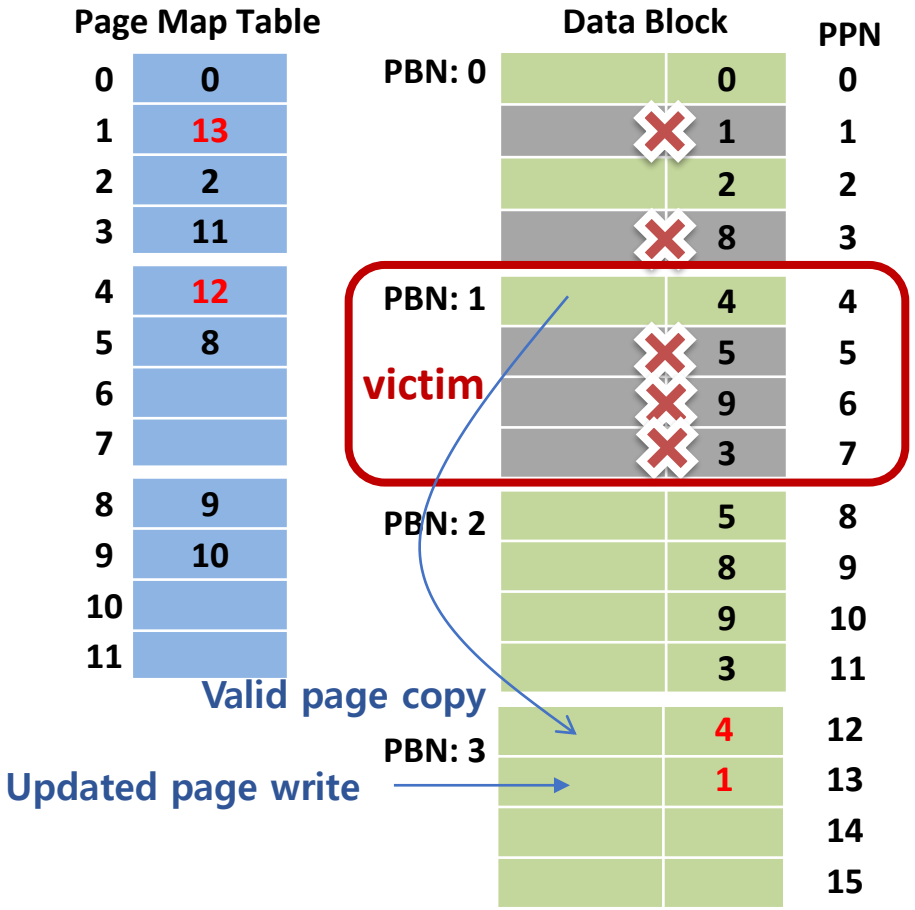
# Example: GC in Page Mapping

- Current state
  - Written to page 0, 1, 2, 8, 4, 5
  - Written to page 9, 3, 5
- New requests (in order)
  - Write to page 8
  - Write to page 9
  - Write to page 3
  - Write to page 1
  - Write to page 4



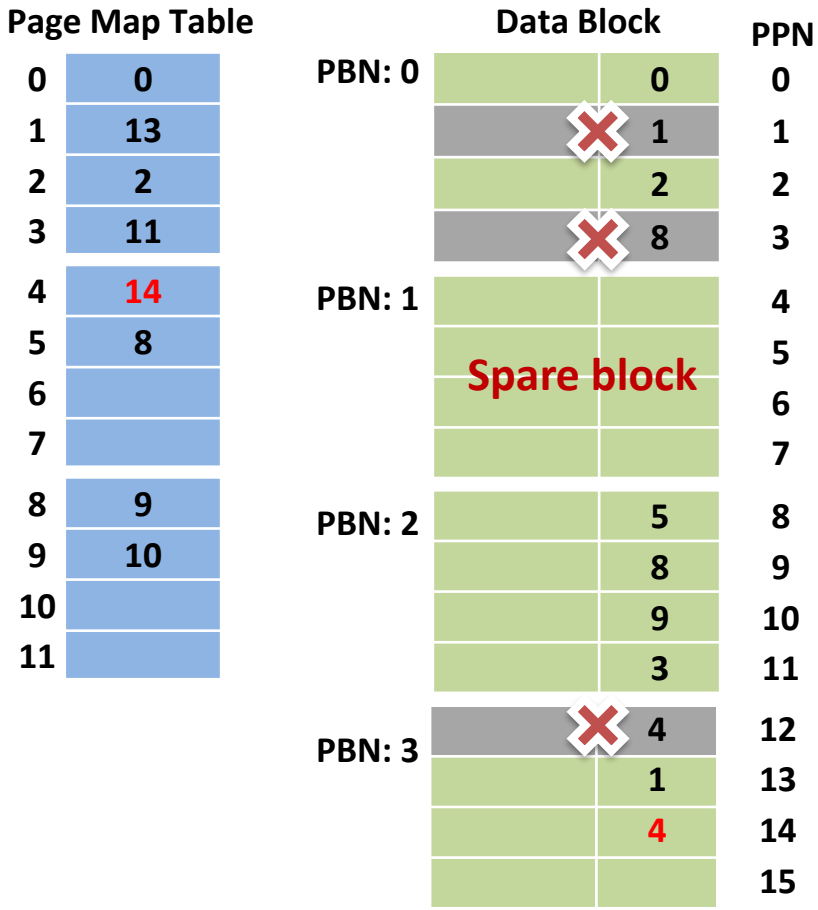
# Example: GC in Page Mapping

- Current state
  - Written to page 0, 1, 2, 8, 4, 5
  - Written to page 9, 3, 5
- New requests (in order)
  - Write to page 8
  - Write to page 9
  - Write to page 3
  - Write to page 1
  - Write to page 4



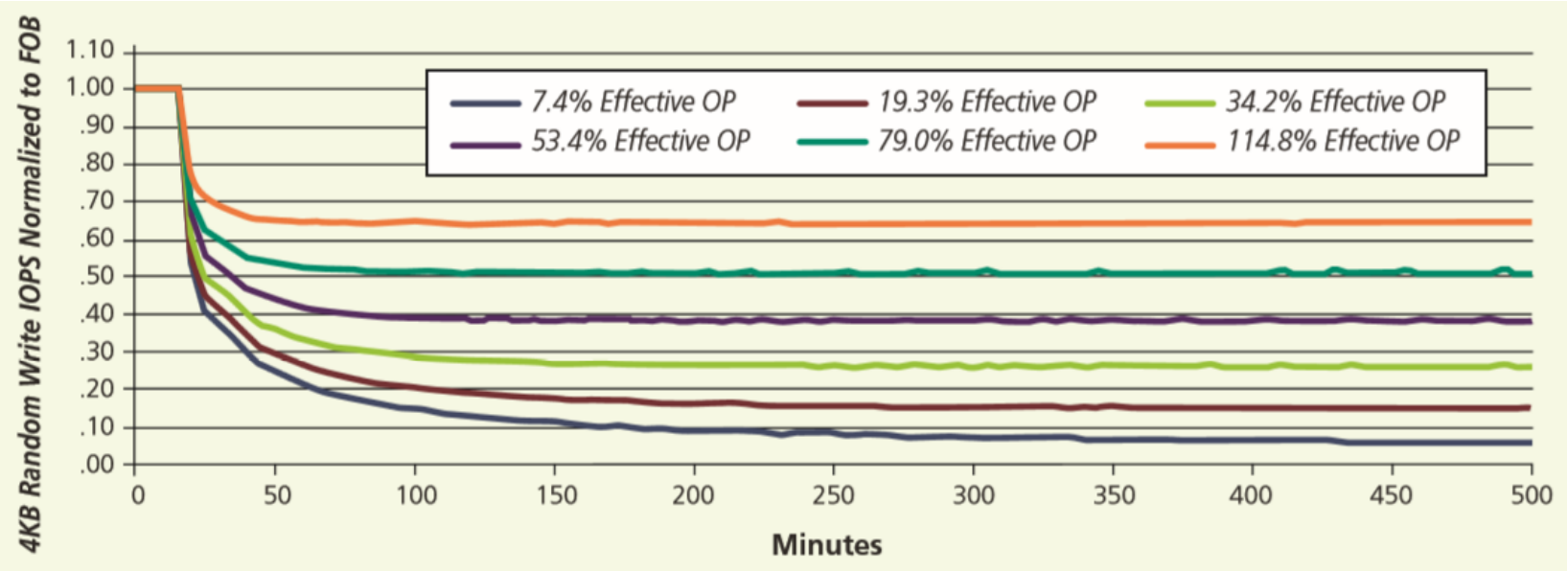
# Example: GC in Page Mapping

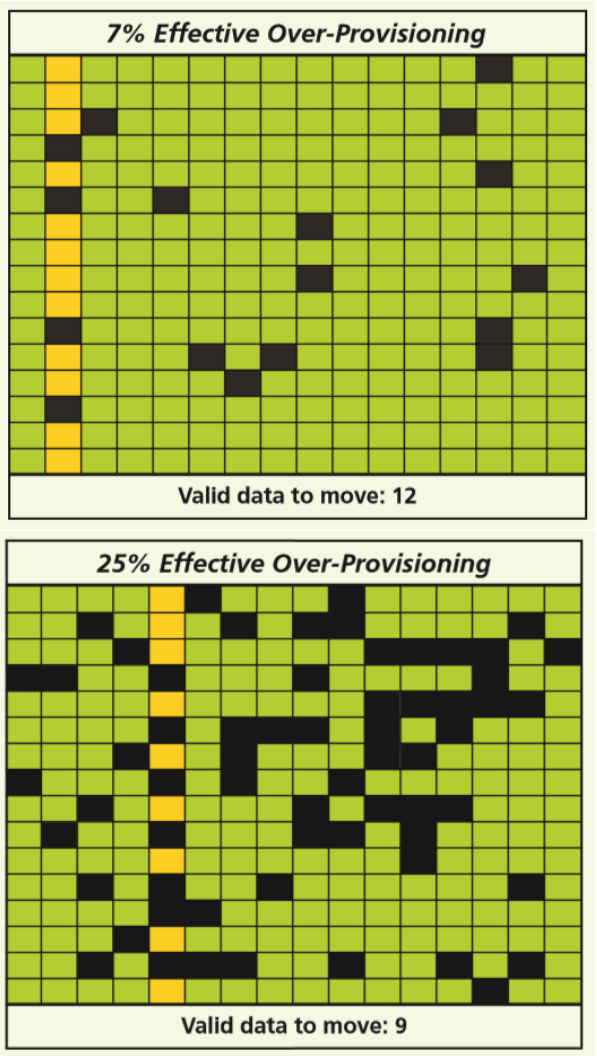
- Current state
  - Written to page 0, 1, 2, 8, 4, 5
  - Written to page 9, 3, 5
- New requests (in order)
  - Write to page 8
  - Write to page 9
  - Write to page 3
  - Write to page 1
  - Write to page 4



# Over-Provisioning and GC

- IOPS for random write workloads
  - OP (Over-Provisioning) = $\frac{Physical\ Capacity}{Logical\ Capacity} - 1$
  - What about for sequential write workloads?




*D. Glen, Differences in Personal vs. Enterprise SSD Performance, Micron Technology, Inc.*

# Challenges

| | |
|---|---|
| **No in-place update** | **Address mapping,<br>Garbage collection (with hot/cold separation)** |
| **Limited P/E cycles** | **Wear leveling** |
| **Bit errors** | **ECC** |
| **Bad blocks** | **Bad block remapping** |
| **Read/write disturbance<br>Retention errors** | **Background activity for data integrity** |
| **Multiple planes/dies/channels** | **Exploiting parallelism, prefetching** |
| **Slower & more power-consuming program/erase operations** | **Hiding latency, power throttling** |
| **Sudden power failure** | **Power loss protection** |

# OS Implications

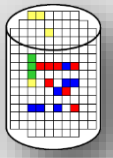
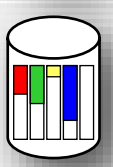
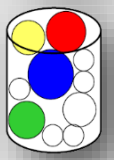
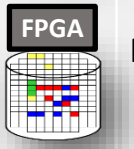
- NAND flash has different characteristics compared to disks
  - No seek time
  - Asymmetric read/write access times
  - No in-place-update
  - Good sequential read/write and random read performance, but bad random write performance
  - Wear-leveling
  - …

  - Traditional operating systems have been optimized for disks. What should be changed?

# SSD Support in OS & Applications

- Align file system partition with SSD layout
- Larger block size (4KB)
- Turn off "defragmentation" for SSDs
- New "TRIM" command (remove-on-delete)
- Simpler & scalable I/O scheduler
- Flash-aware file systems (e.g., F2FS in Linux/Android)
- New "multi-stream" interface
- User-level storage access (e.g., SPDK)
- Fairness, isolation, etc.

# New NVMe SSD Proposals

| | Interfaces | Data placement | Flash management | Overwrite | Data movement | Host CPU utilization | Hardware complexity | Computation capability | Software modification |
|---|---|---|---|---|---|---|---|---|---|
| **Conventional SSD**  | Block (SATA, SAS, NVMe) | Host | Device | Yes | - | - | - | No | - |
| **ZNS SSD**<br>**FDP SSD**  | Restricted Block (NVMe) | **ZNS**: Host<br>**FDP**: Host + Device | **ZNS**: Host<br>**FDP**: Host + Device | **ZNS**: No<br>**FDP**: Yes | High | High | Low | No | High |
| **KV-SSD**  | Key-value (NVMe) | Device | Device | Yes | Low | Low | Medium | Some | High |
| **SmartSSD**   | FPGA + Block (NVMe) | Host | Device | Yes | Low | Low | High | Yes | Very High |
| **Computational Storage**  | Compute + Block? (NVMe) | Host or device? | Device | Yes | Low | Low | High | Yes | Very High |

# Beauty and the Beast

- NAND Flash memory is a beauty
  - Small, light-weight, robust, low-cost, low-power, non-volatile device
- NAND Flash memory is a beast
  - No in-place-update
  - Much slower program/erase operations
  - Erase unit > read/write unit
  - Bit errors
  - Limited lifetime etc.
- Software support is essential for performance and reliability!