Overview of Flash Management SW

Jihong Kim
Dept. of CSE, SNU

Contents

- Flash Management Tasks
  - Address Translation
  - Garbage Collection
  - Wear Leveling
- S/W Support for NAND Flash Memory
  - Flash Translation Layers
  - Flash File Systems
  - Memory Technology Device

Remind: NAND Flash Memory

- Pros
  - Nonvolatile
  - Fast random access
  - Lower power consumption
  - Small size
  - Shock resistance
- Cons
  - Out-place Update
  - A limited number of erase operations
  - Expensive

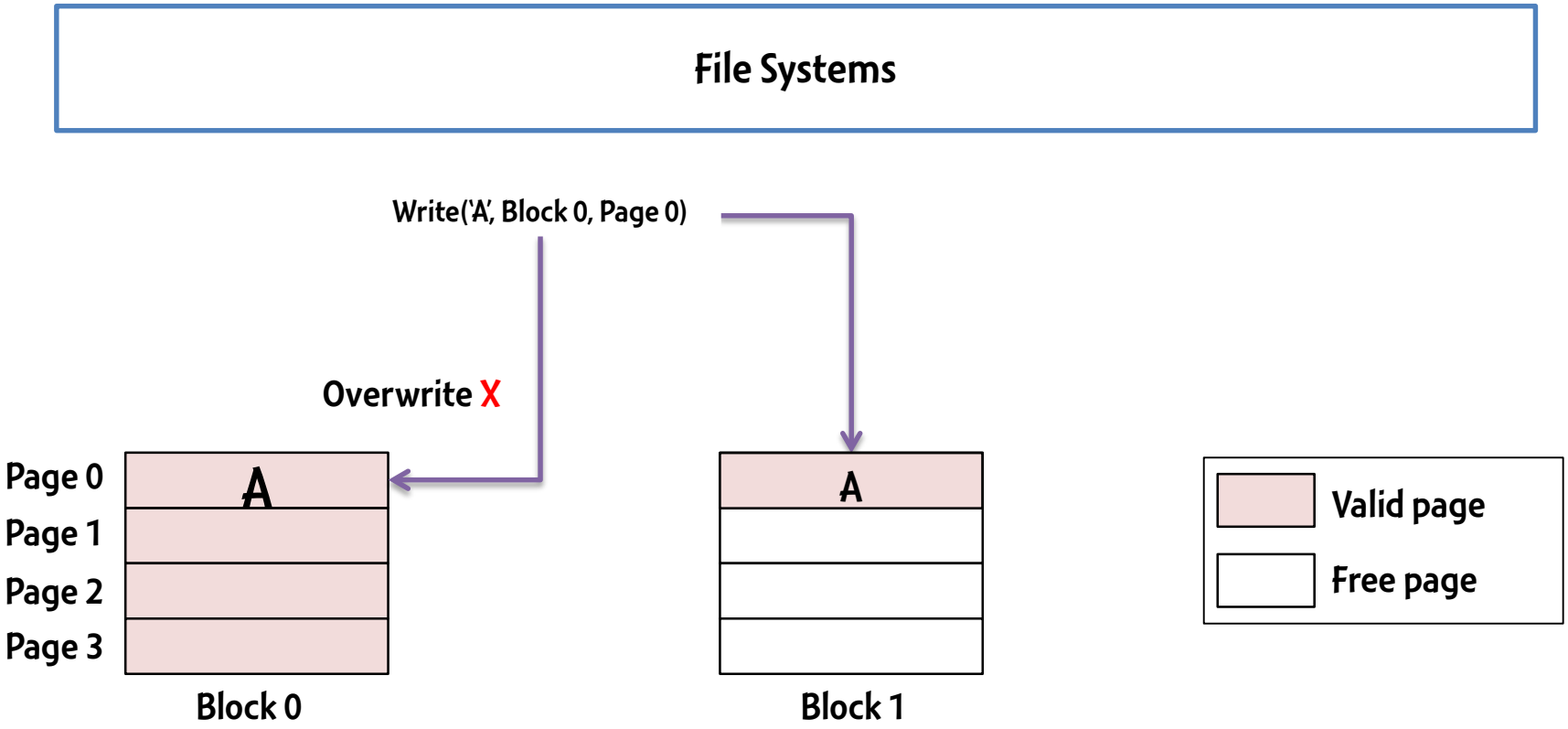
Remind: Basic Operations

- Read
  - Fast (~200 µs)
  - Page (2~8 KB)
- Write
  - Slow (~800 µs)
  - Page (2~8 KB)
  - Out-place update
- Erase
  - Very slow (~2ms)
  - Block (128~512 KB)

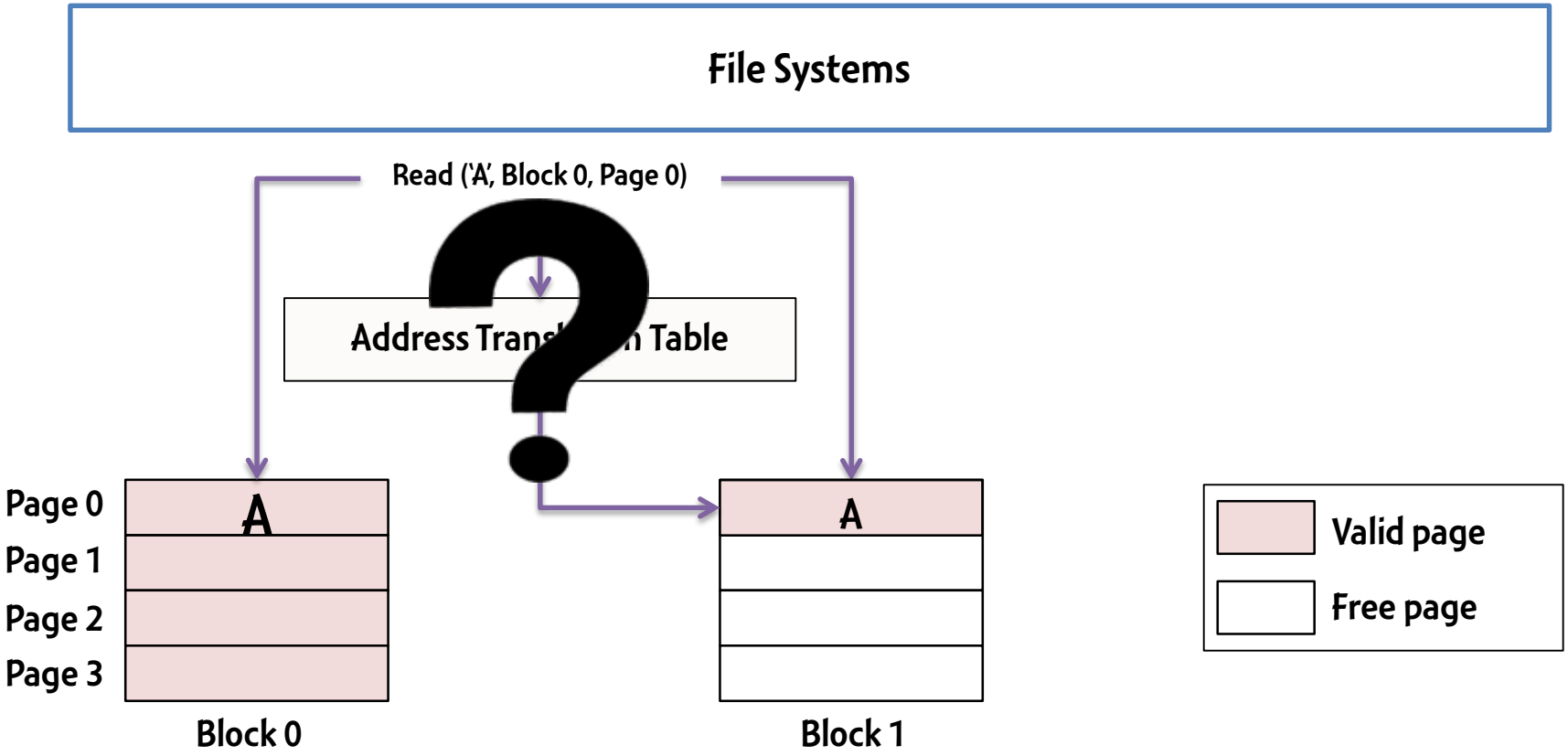
Flash Management Tasks

- Essential
  - Address Translation
    - Avoid in-place update
    - Logical Block Address (LBA) -> Physical Block Address (PBA)
  - Garbage Collection
    - Reclaim invalid blocks -> Get new free blocks
- Optimization
  - Wear Leveling
    - Erase all blocks evenly -> Extend life time

Out-place Update

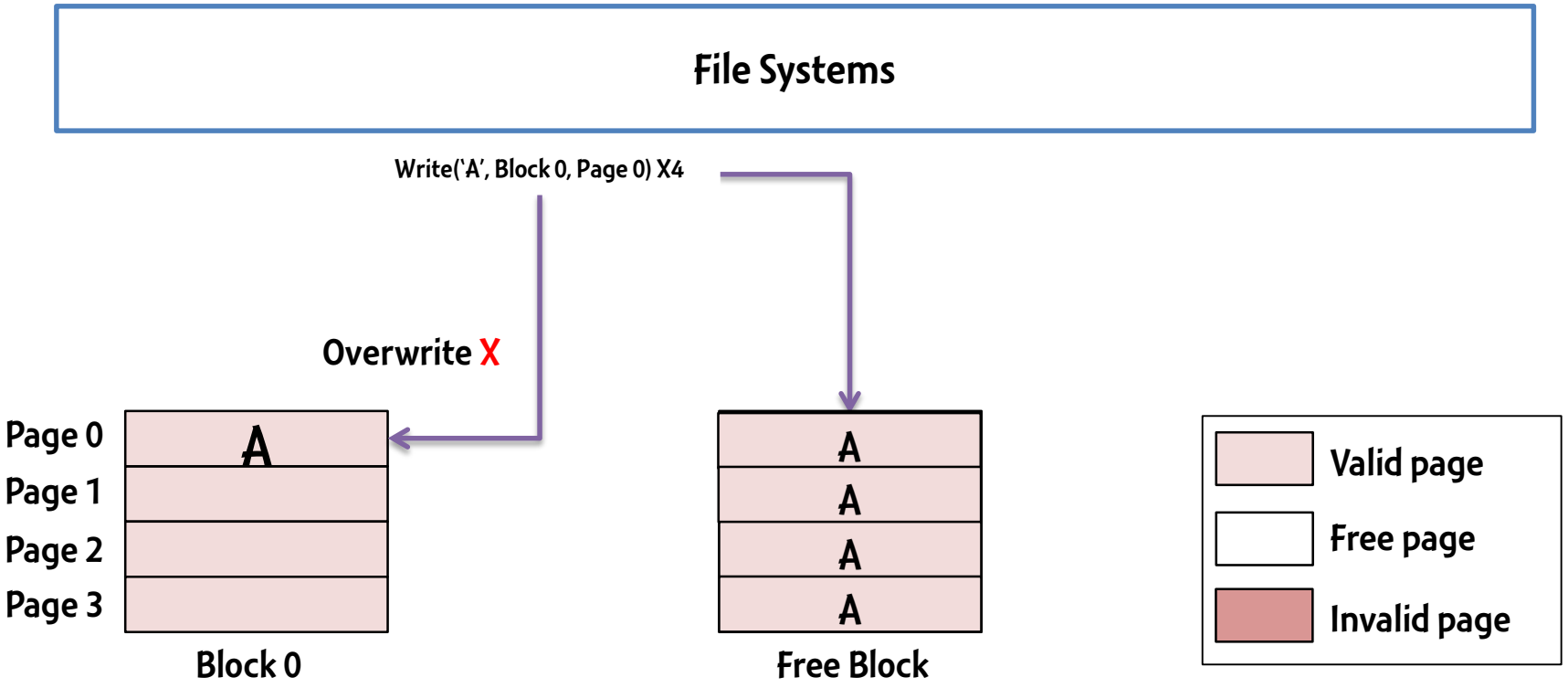


Address Translation



Garbage Collection

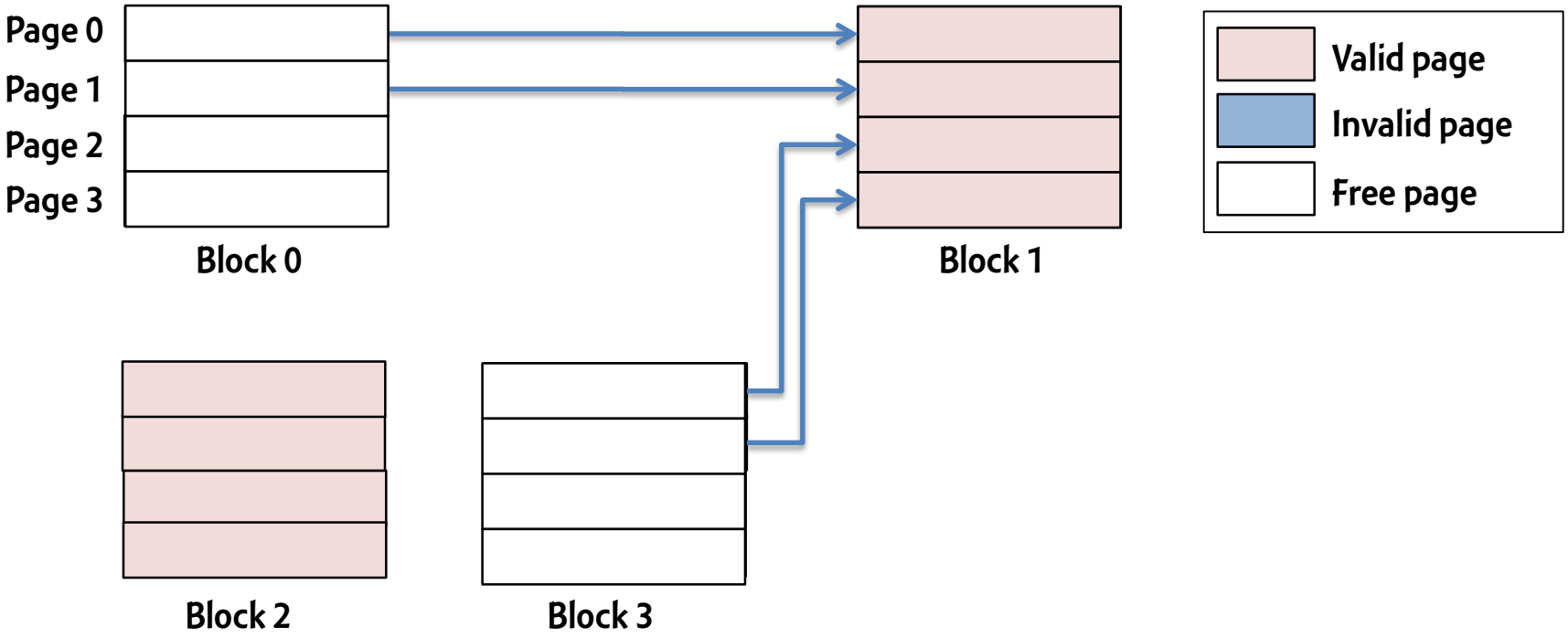
- NAND flash memory does not allow *in-place update*



- NAND flash memory will be full of invalid data…

Garbage Collection

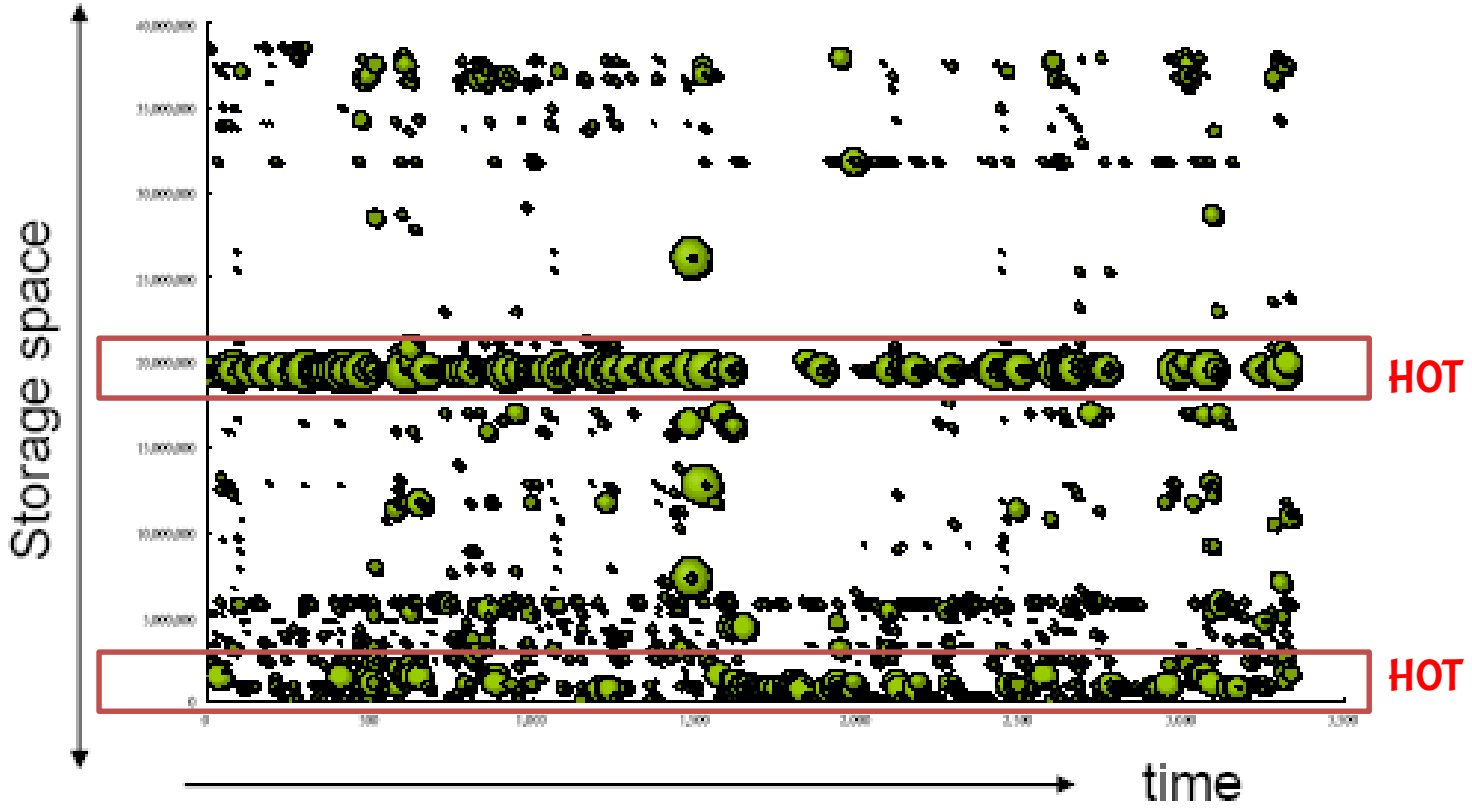
- *Gather valid data -> Erase invalid data*



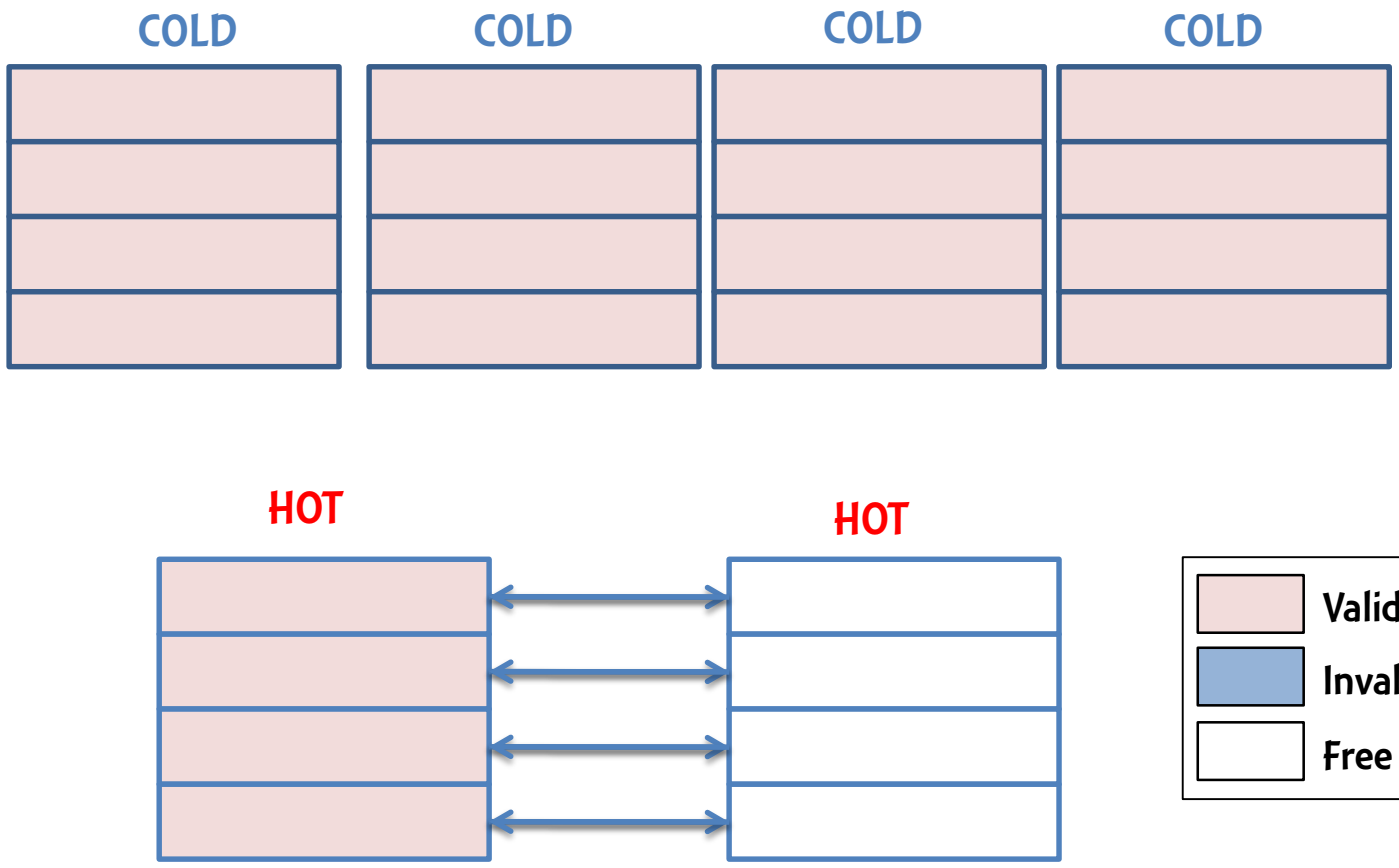
Limited Erase Cycles

- Remind
  - Basic units of operations
    - Erase: Block
    - Read & Write: Page
  - Erasure before write
  - Flash memory block has a limitation (*erasure cycle*) on the number of erase operations (about 3K~10K in MLC)
  - Blocks should be evenly erased to lengthen the overall lifetime of flash memory

Spatial Locality of Write Request



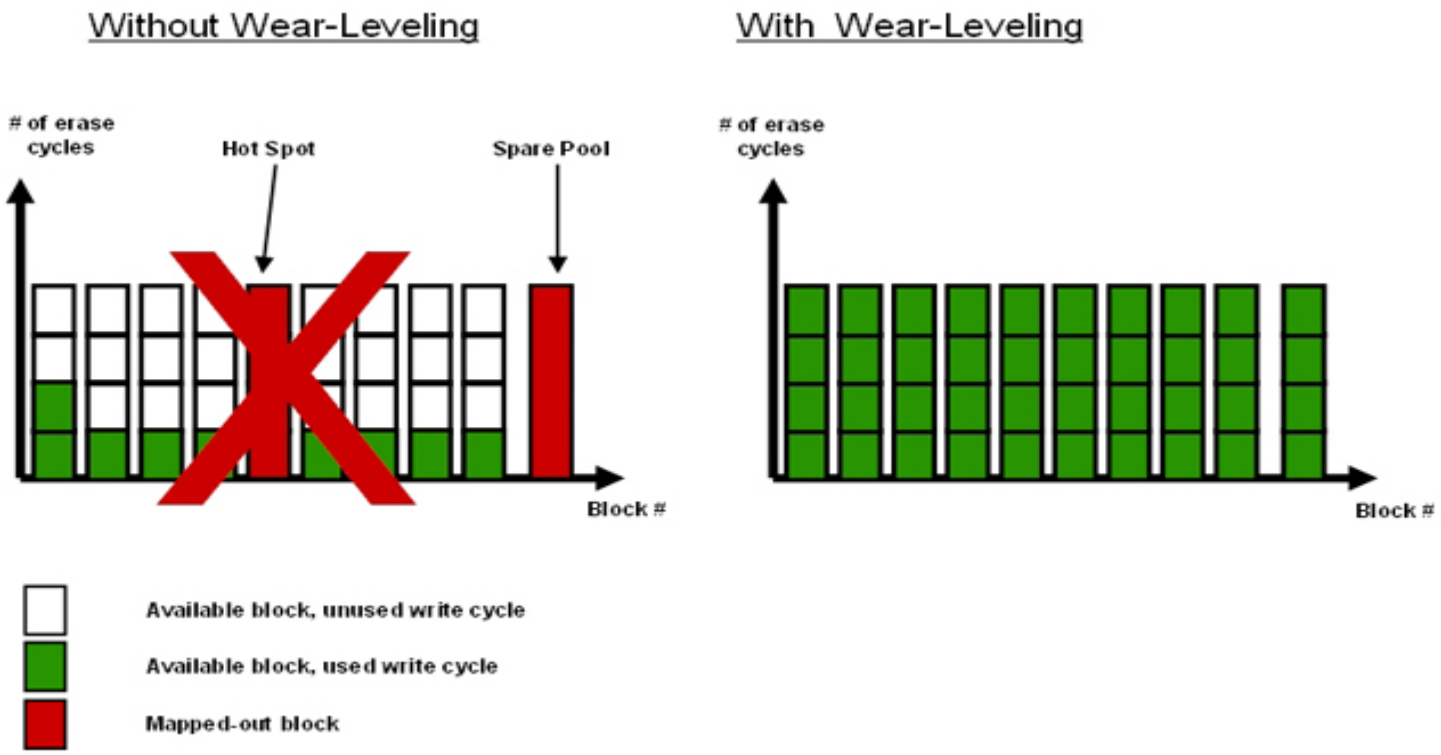
Wearing of Flash Memory Blocks



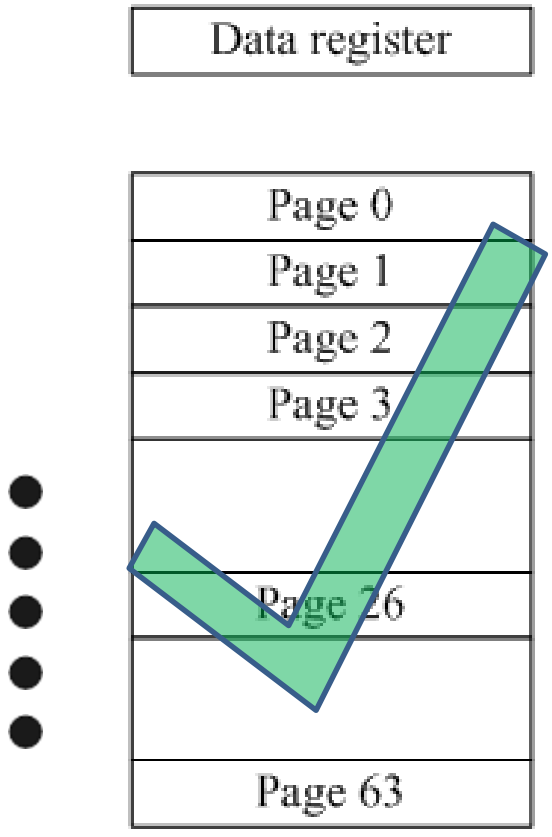
- Blocks with hot data are likely to be erased more

Wear Leveling

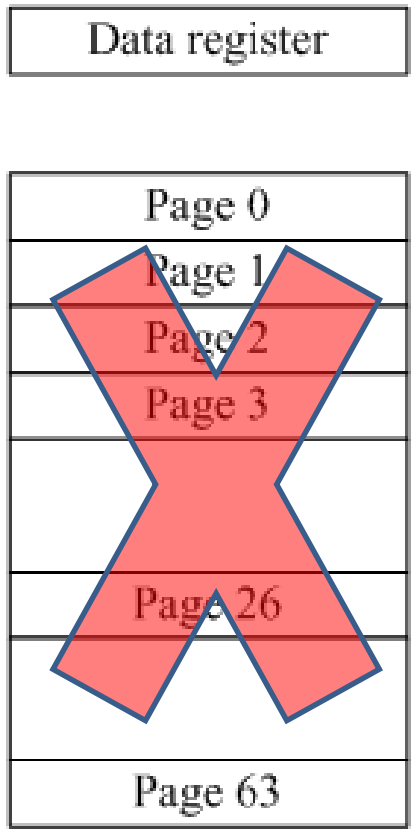
- Erase evenly all blocks



Side: Program Sequence Constraint



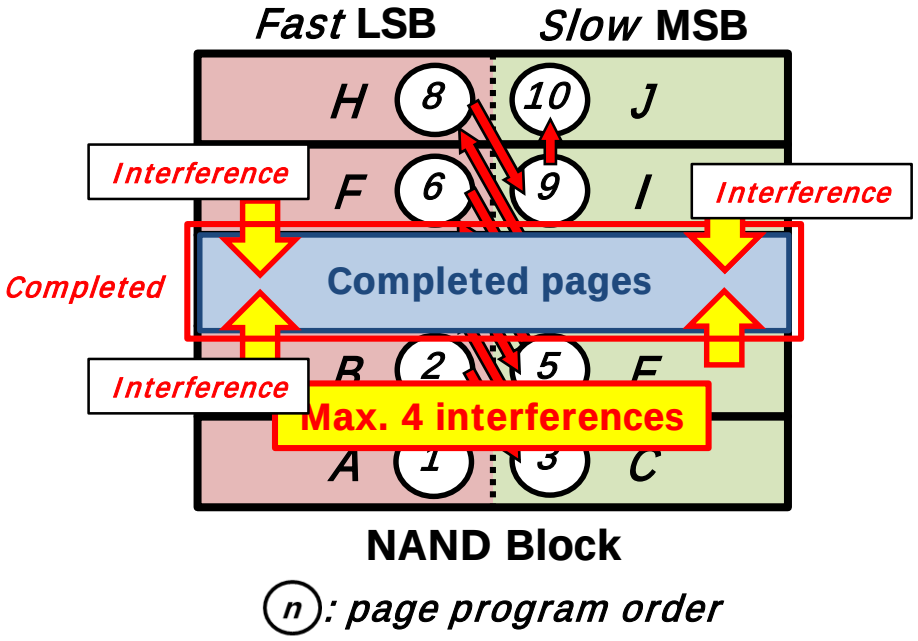
Consecutive page program

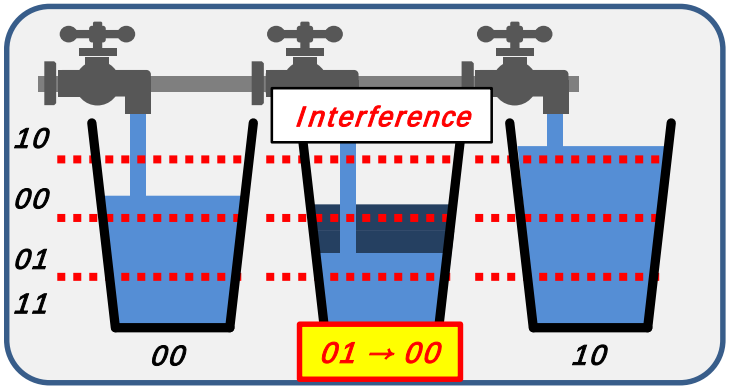


Random page program

FPS: Fixed Program Sequence (Conventional)

- **Cell-to-cell interference: a completed cell (whose LSB and MSB are both programmed) can be affected by program operations for neighboring cells.**

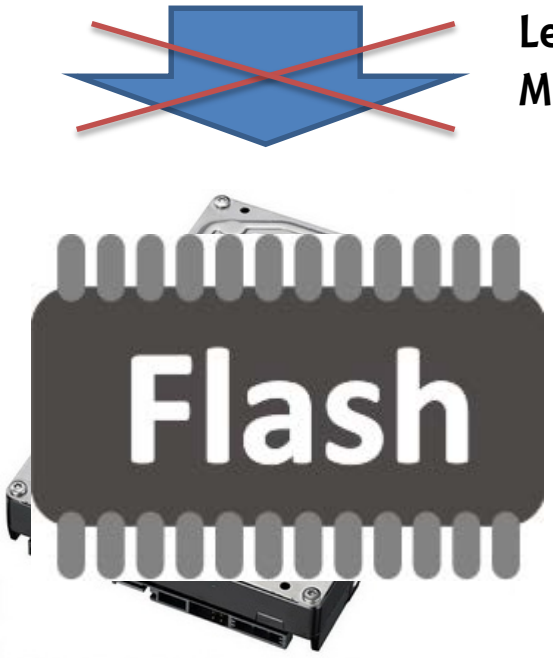




- ***Fixed* program sequence for minimizing the cell-to-cell interference**
  - **All pages are interfered just once (by the upper MSB page).**

File Systems for H.D.D

File Systems (e.g., EXT3, NTFS, FAT32, …)



Legacy file systems don't understand NAND Management operations:

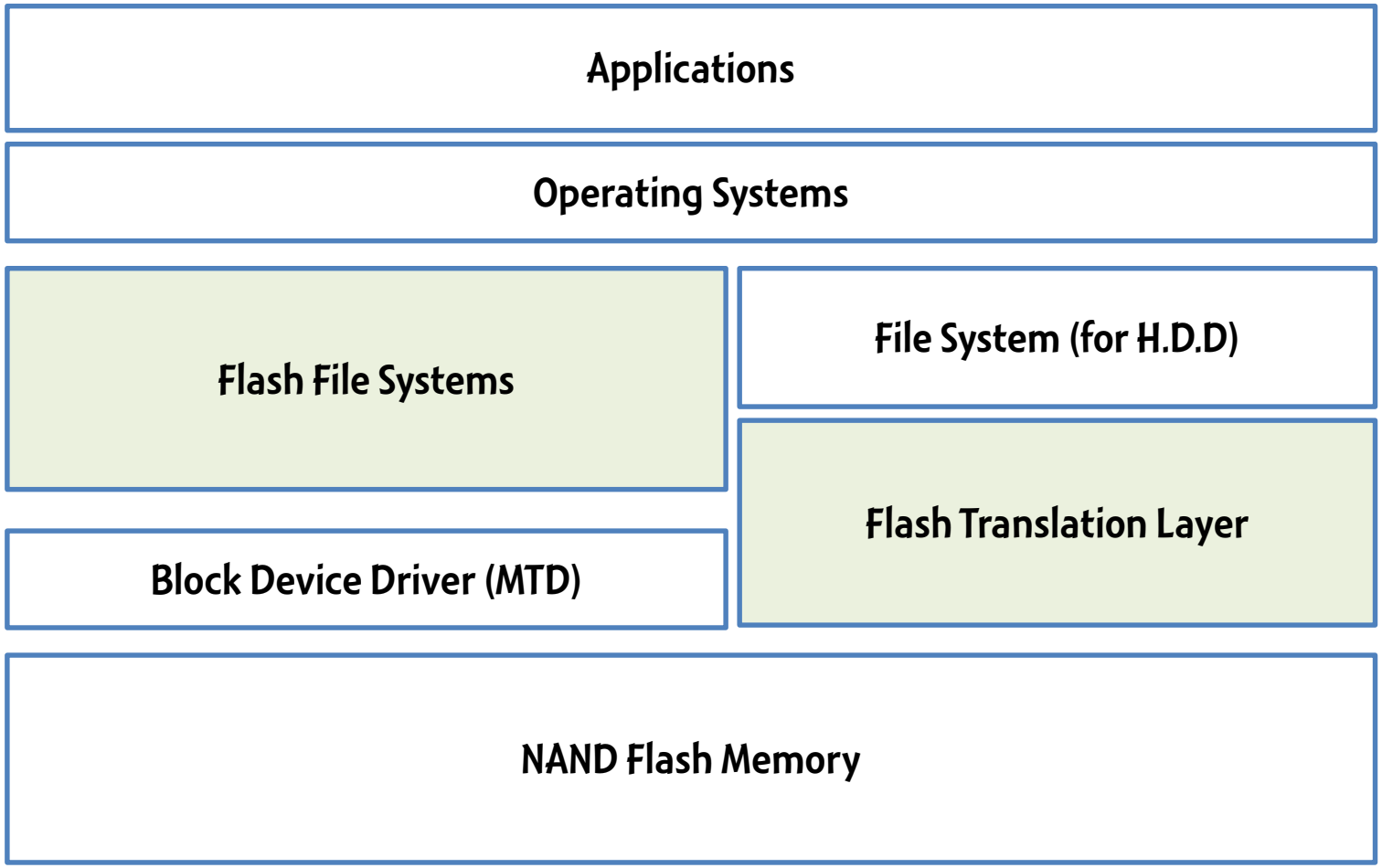
- "what is garbage collection?"
- " what is address translation?"

Allows in-place update
Does not need to garbage collection
…

S/W for NAND Flash Memory

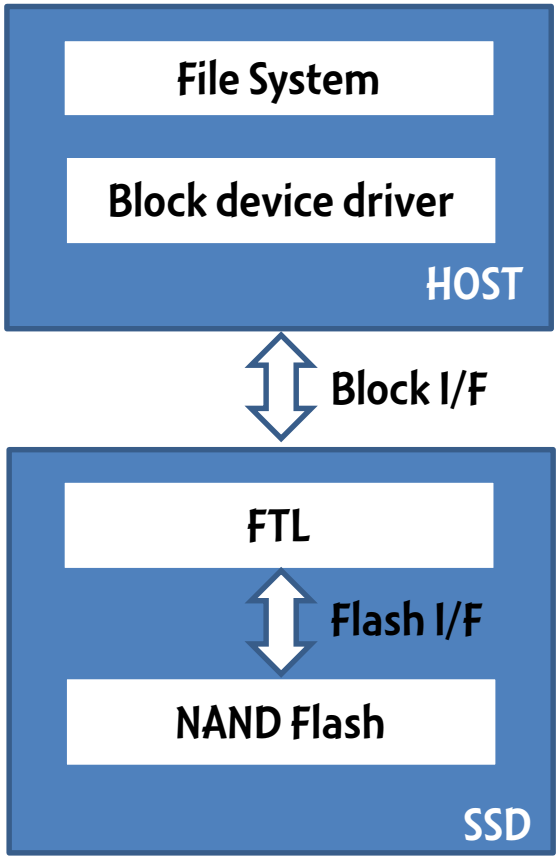
- Flash Translation Layer
  - Additional S/W layer for NAND flash memory
  - Emulates conventional block devices (such as HDDs).
- Flash File Systems
  - NAND flash-aware file system
  - Replaces legacy file systems in a flash-friendly fashion.
- MTD Driver
  - Provides a generic API for flash-based storage devices.

S/W Architecture



Flash Translation Layer

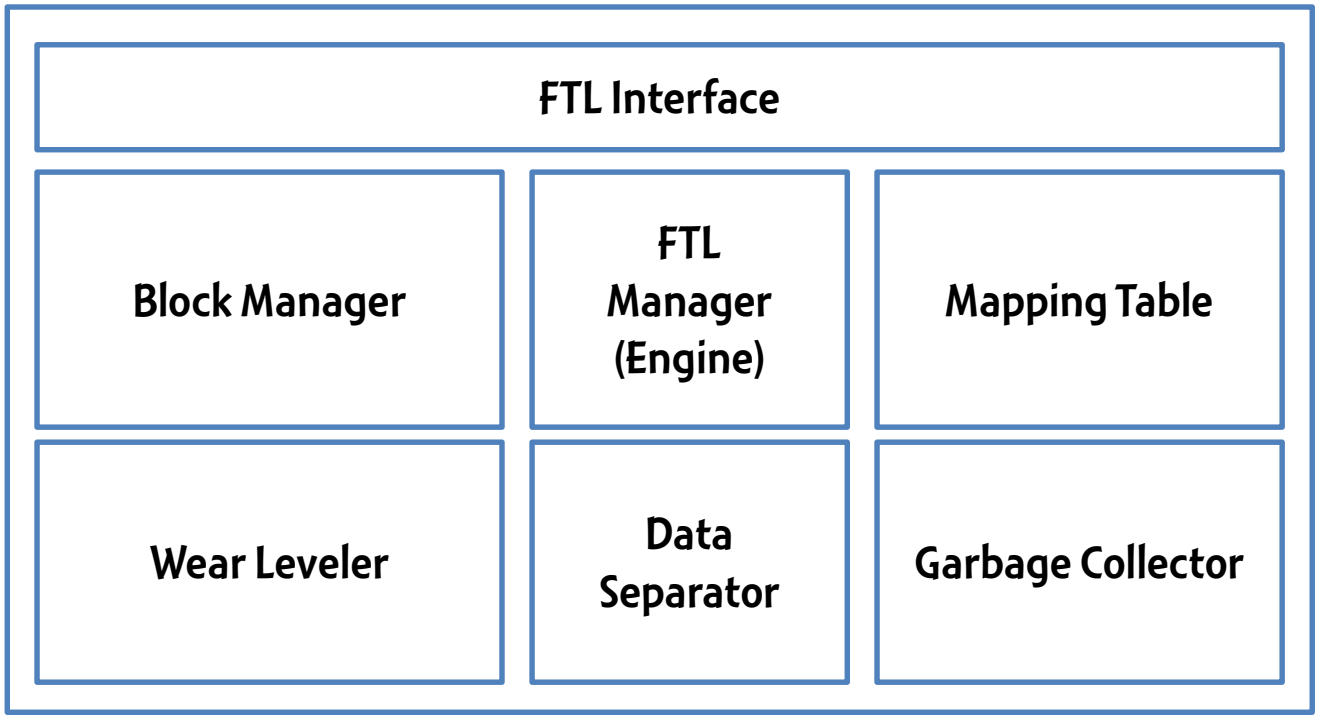
- A software layer to make NAND flash emulate traditional block devices (or disks)



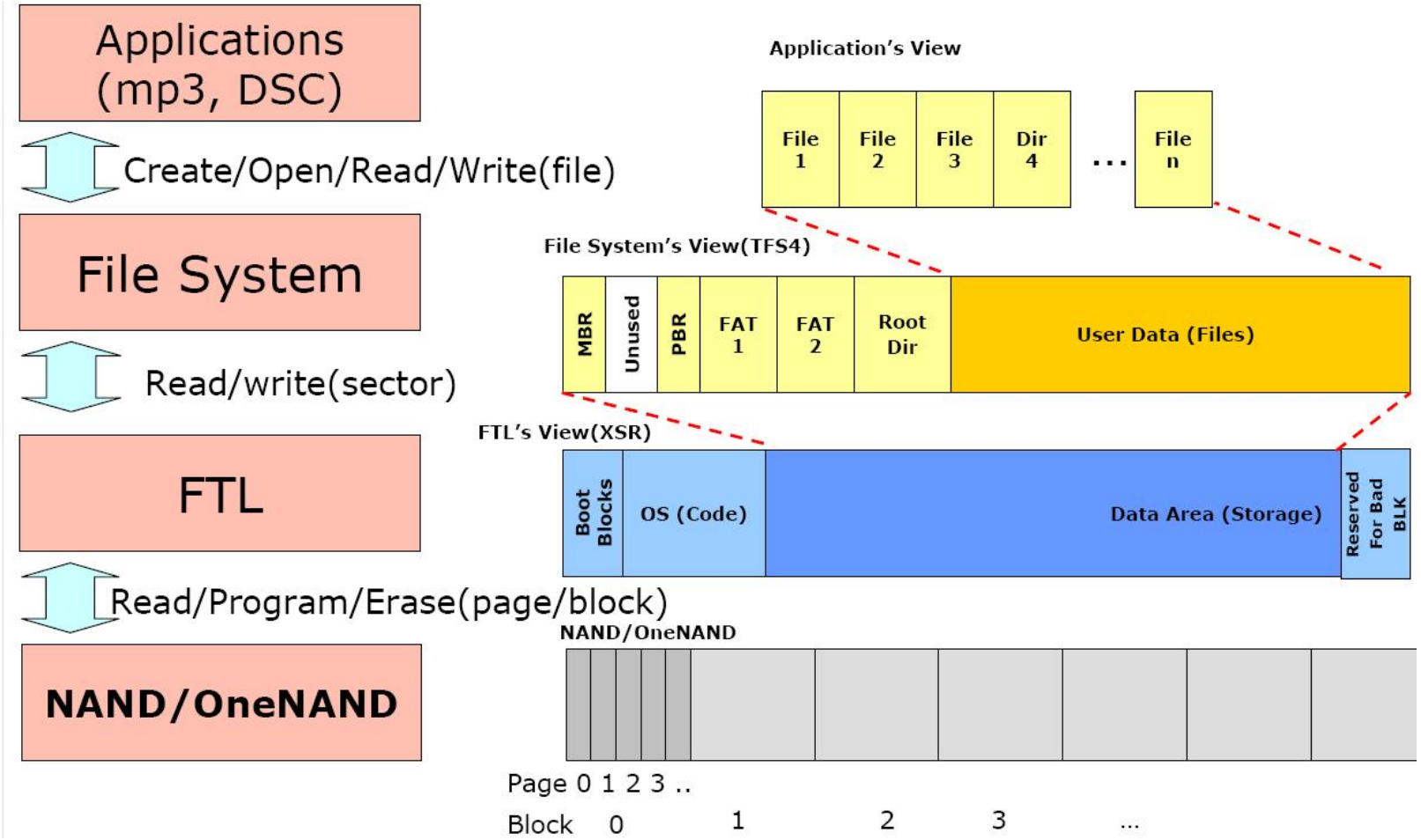
Roles of FTL for NAND Flash Memory

- For performance
  - Indirect mapping (address translation)
  - Garbage collection
  - Hot/cold separation
  - Interleaving over multiple channels/flash chips/planes
  - Request scheduling
  - Buffer management
  - …
- For Reliability
  - Bad block management
  - Wear-leveling
  - Power-off recovery
  - Error correction code (ECC)
  - …
- Other Features
  - Encryption
  - Compression
  - Deduplication
  - …

Layout of FTL



Logical Layout



Flash File Systems

- NAND flash memory-aware file systems
  - Provides general file system operations
  - Directly addressing the characteristics of flash memory
    - Metadata have physical address of raw device
  - Wear leveling
  - Bad block management
  - In general, flash file systems are based on Log-structured File System
    - NAND-flash friendly structure

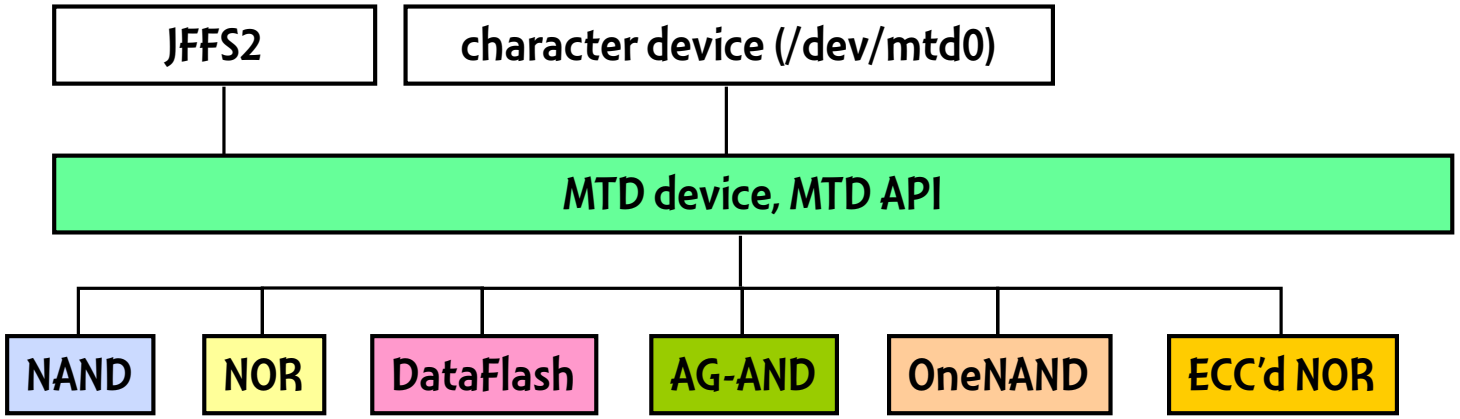
FTL & Flash File Systems

	FTL	Flash file system
Pros	Compatibility Utilize Legacy File Systems	Exploits fully characteristics of NAND flash memory
Cons	Inefficiencies in flash memory management	Limited expandability

Device Drivers for Flash-based Storage

- MTD (Memory Technology Devices) overview
  - Linux subsystem (`drivers/mtd/`)
  - Provides uniform access to various flash devices
  - Provides a generic API for NAND-based storage systems
  - Provides an "MTD device" abstraction

MTD overview



MTD device vs Block device

Block device

- Consists of sectors
- Sectors are small (512, 1024 bytes)
- 2 operations: ***read*** and ***write***
- Bad sectors are hidden by hardware
- Sectors do not get worn out

MTD device

- Consists of eraseblocks
- Eraseblocks are larger (32-128 Kilobytes)
- 3 operations: ***read***, ***write*** and ***erase***
- Bad eraseblocks are not hidden
- Eraseblocks get worn-out after 10^4-10^5 erasures.

Important Issues in Flash S/W

- Garbage Collection
  - Triggering Condition
    - If there are no free blocks
    - Periodically
    - When the number of free blocks is under a certain threshold
    - When an I/O request does not happen
  - Victim block Selection
  - The number of reclaimed blocks
    - One block at a time
    - Until the number of free blocks exceeds a certain threshold

Important Issues in Flash S/W

- Mapping Scheme
  - Page? Block? Hybrid?
- Wear Leveling
  - Triggering Condition
  - Reducing Overhead
- Bad Block Management
- Reducing Memory Footprint
- Minimizing Computation Overhead
- Flash-Aware Data Structure Design

Reference

- Aayush Gupta et al., "DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings", ASPLOS 2009

- Yang Hu et al., "Performance Impact and Interplay of SSD parallelism through Advanced Commands, Allocation Strategy and Data Granularity", International Conference on Supercomputing, 2011

- http://www.linux-mtd.infradead.org/