Basic

FW Concepts

Lee Hao Zhi

AGENDA

- Introduction
- Block Base Mapping
- Page Base Mapping
- Garbage Collection
- Write Amplification Factor
- Over Provision
- Summary

Introduction

Overview

Top NAND Flash Controller solution in the world

Application

Filesystem

Host Driver

Host: PC, Tablet, Smartphone, Smartwatch, etc.

Protocol: USB, SD, eMMC, UFS, SATA, NVMe, etc.

Controller

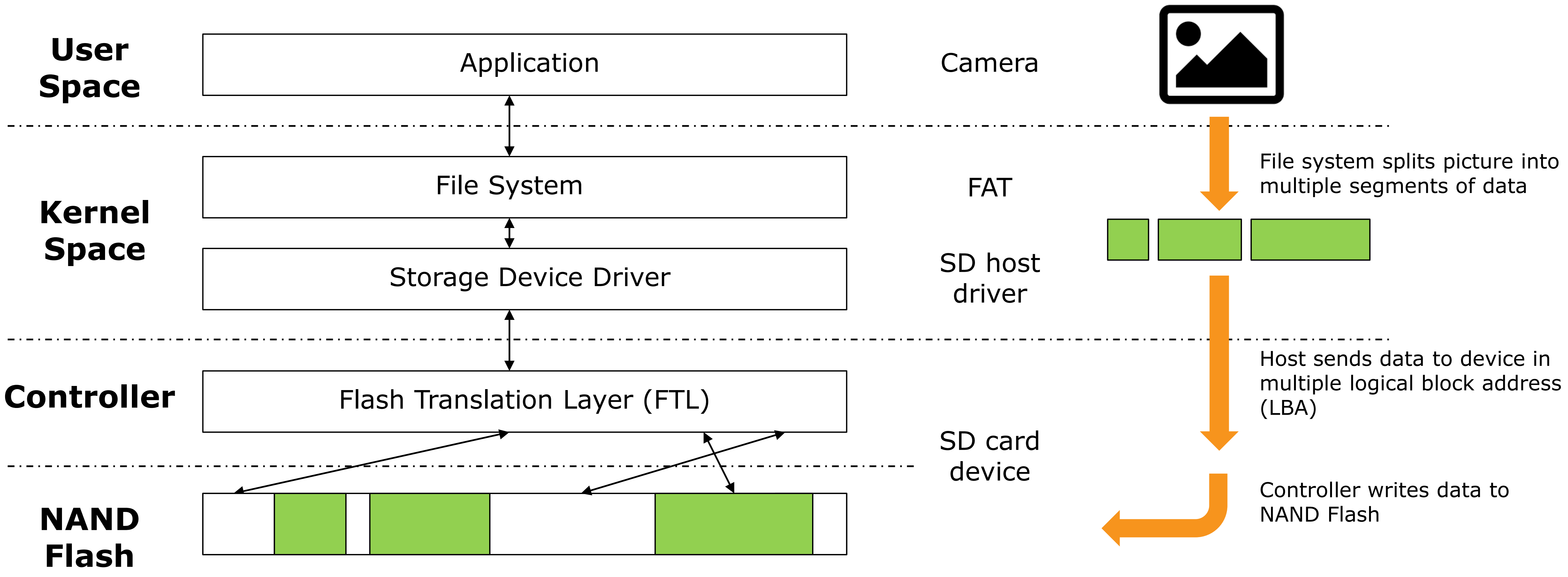
NAND Flash

Host layer

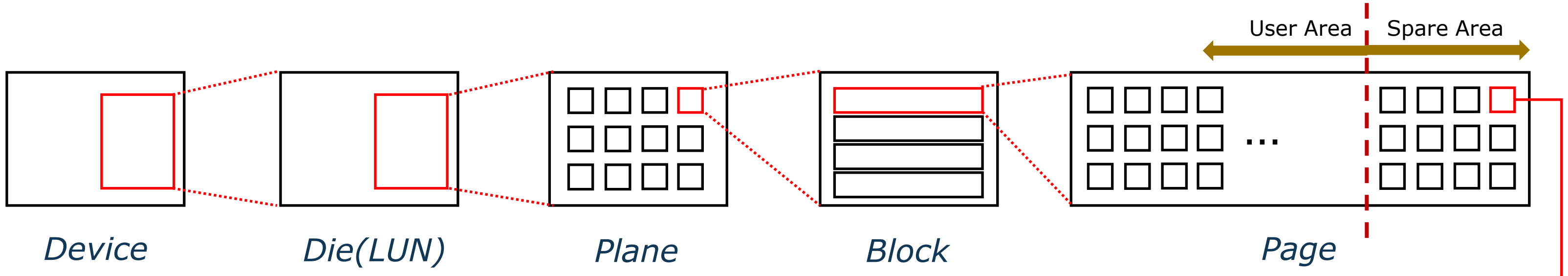
Flash Translation layer (FTL)

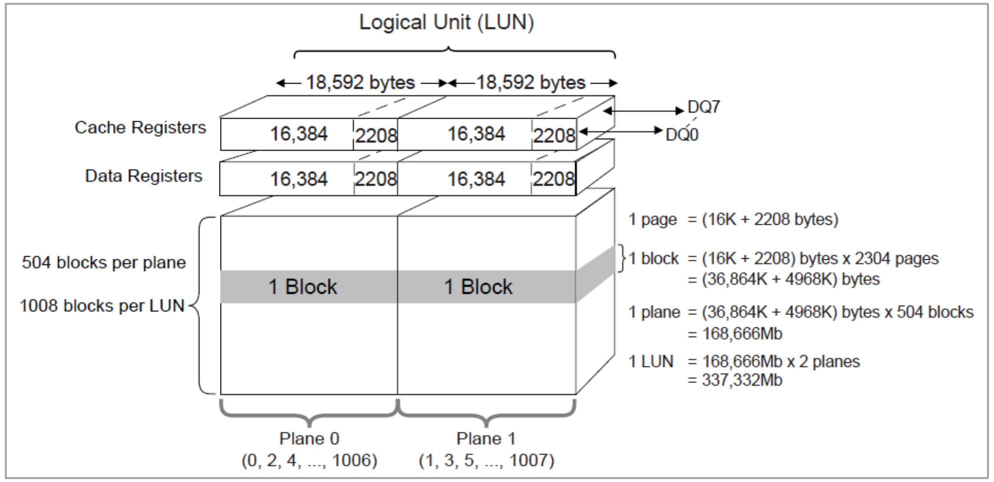
NAND layer

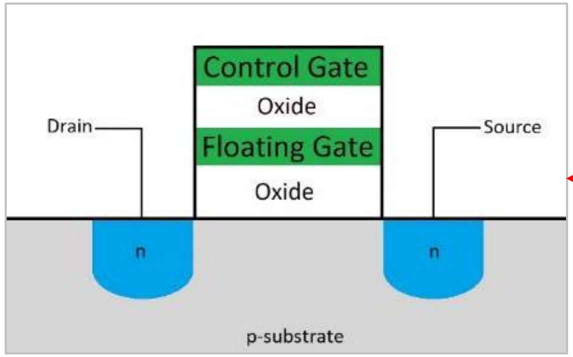
Application & Flash Storage Device



NAND Flash Structure

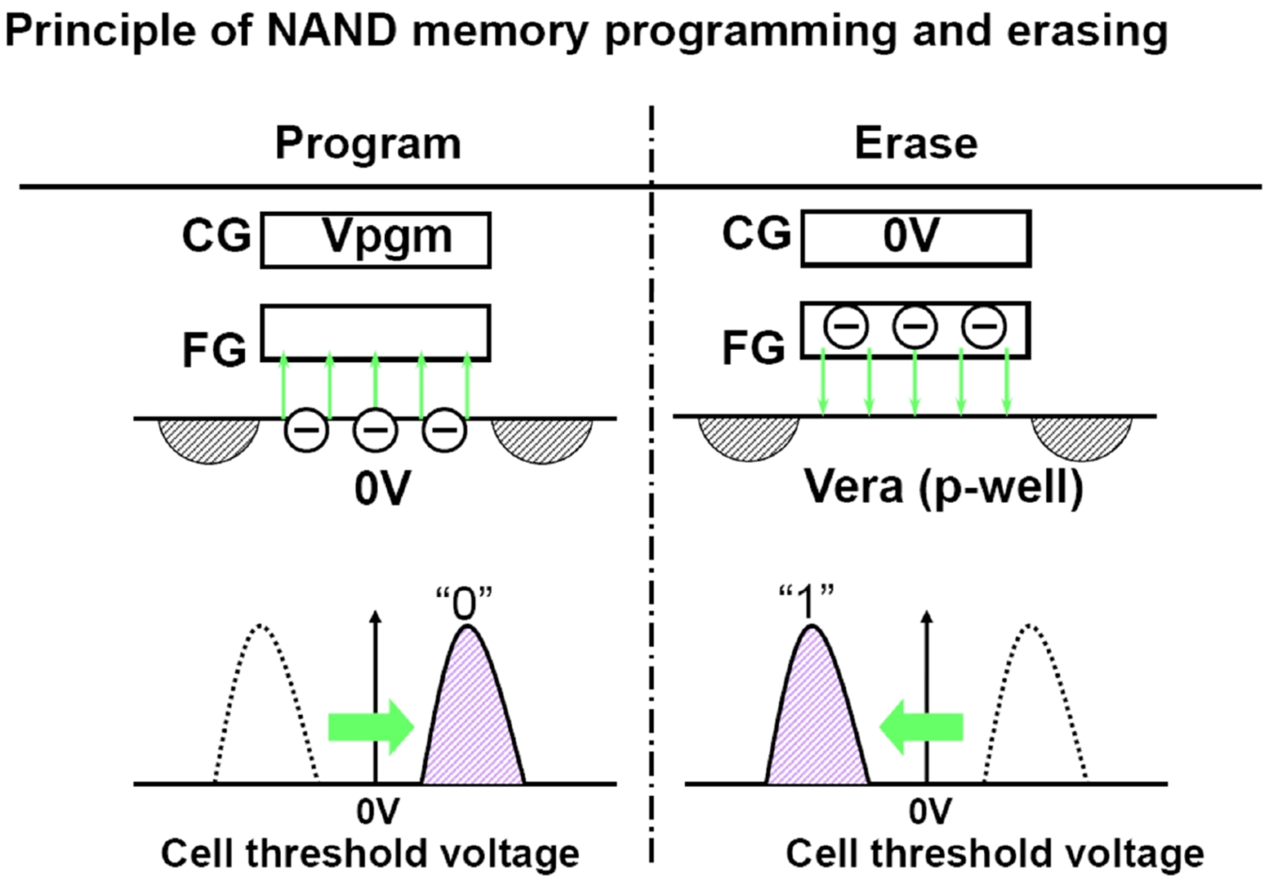






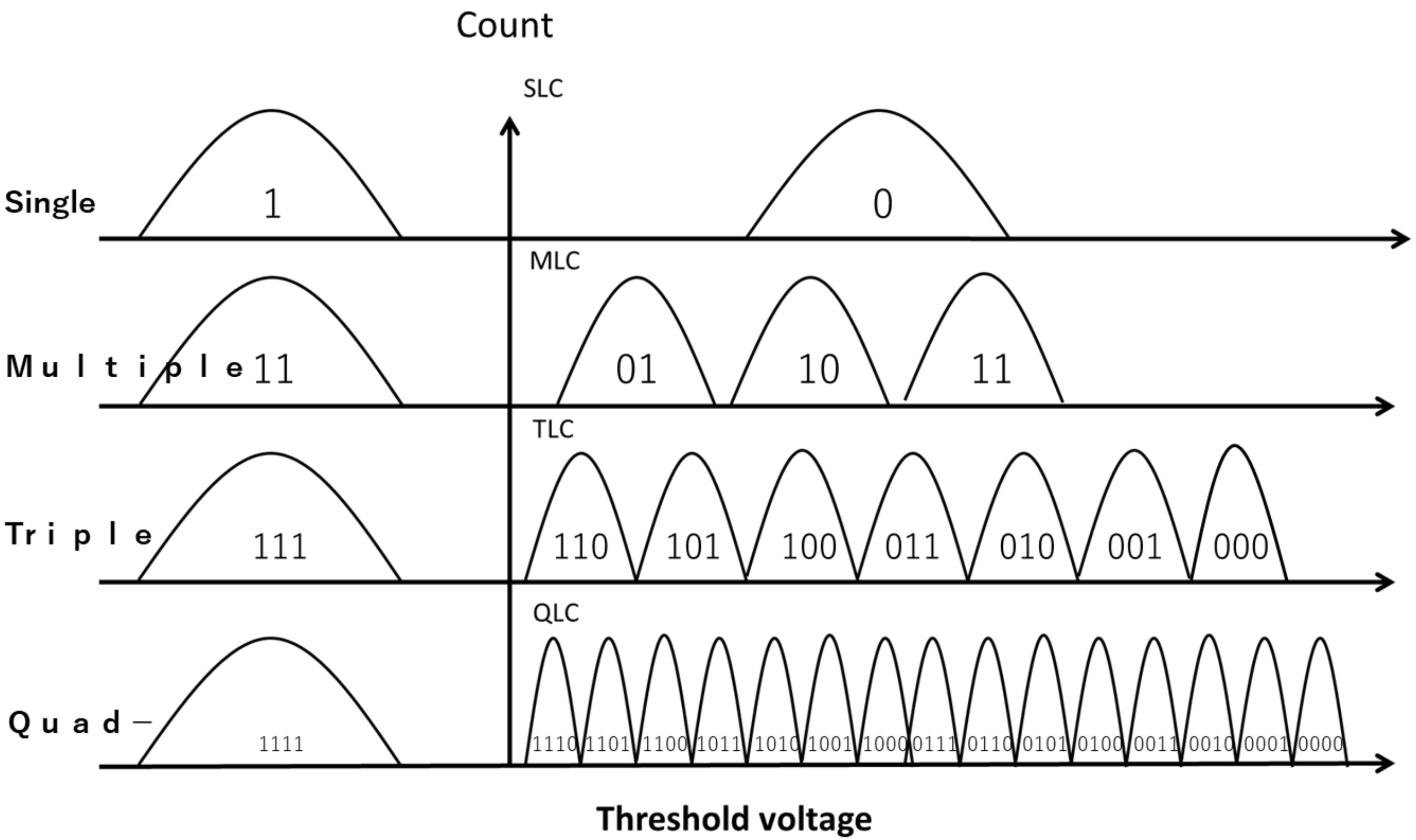
Memory Cell

Program & Erase (P/E) Operation



SLC vs MLC vs TLC

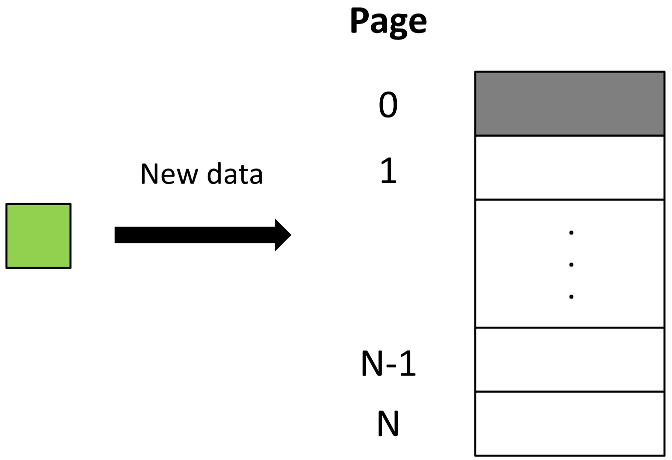
- Normal Voltage Threshold (VT) distribution

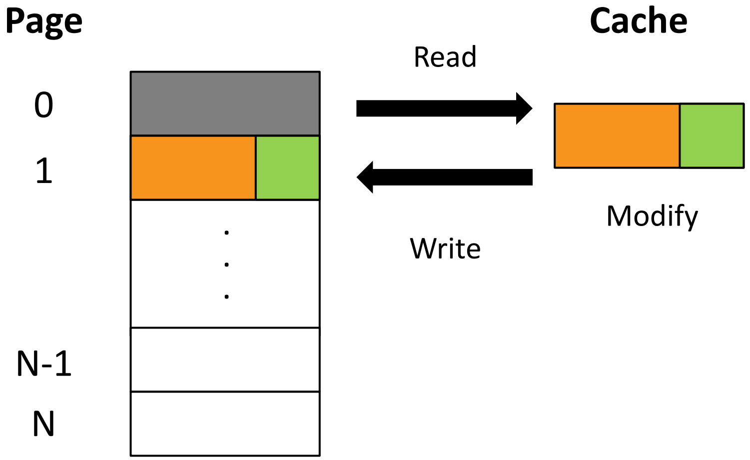


NAND Flash Limitations

Q: What should we do when we cannot overwrite?

- **Page** is the smallest unit for read and program
- **Block** is the smallest unit for erase
- Must erase before program (cannot overwrite)

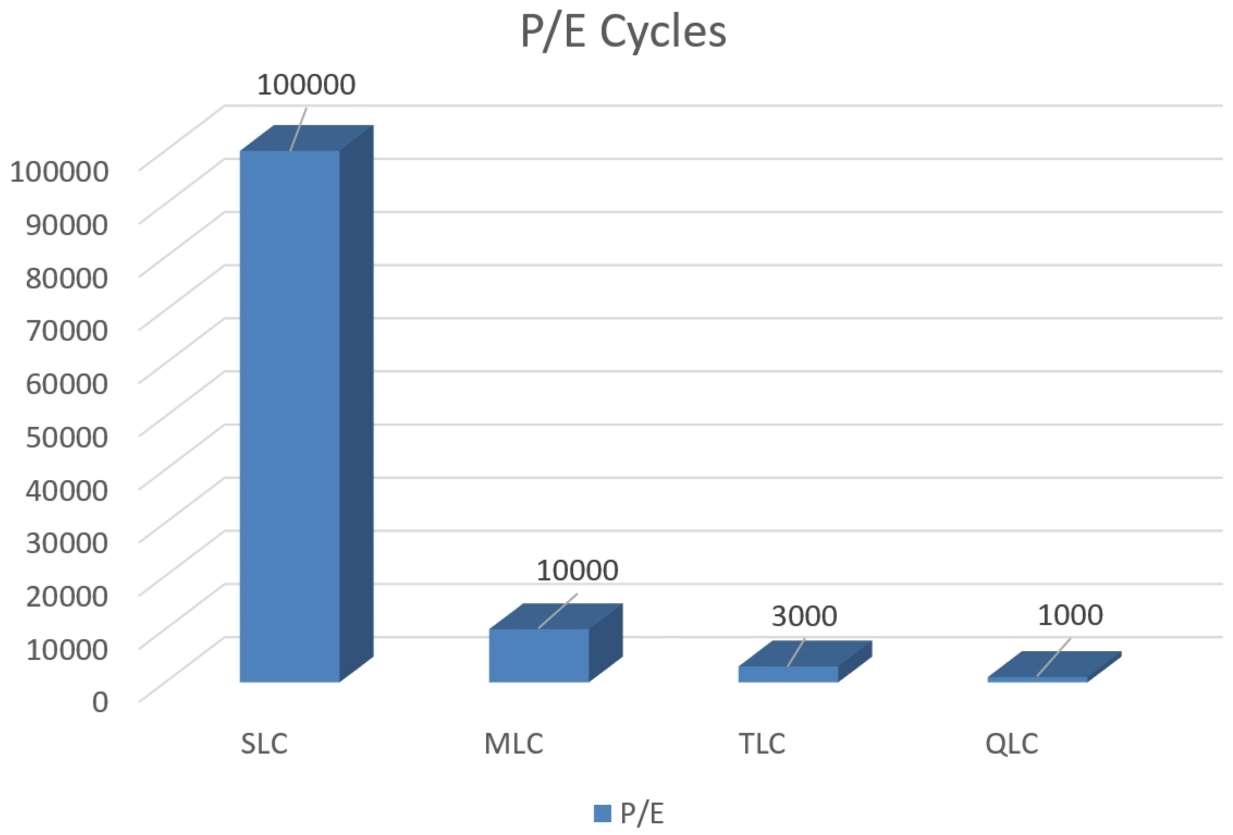




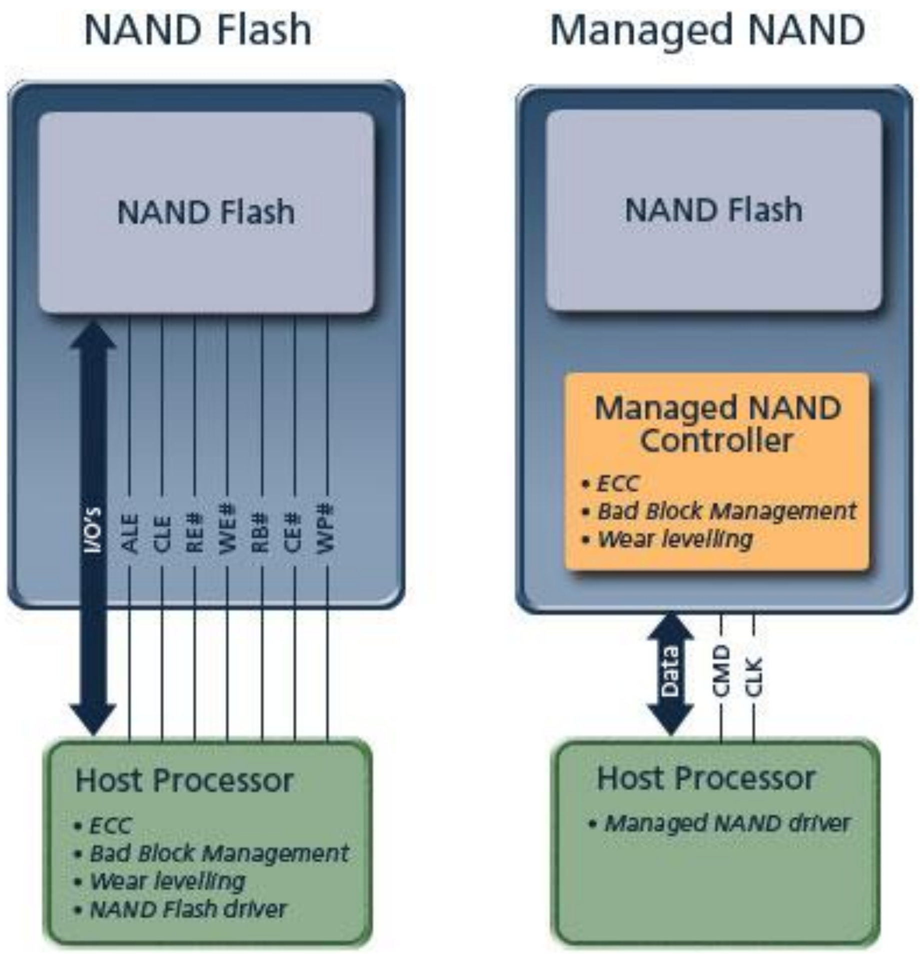
- Each read/program/erase operation has busy time to complete

NAND Flash Limitations

- Program/Erase (PE) Cycle: SLC = 100k, MLC = 10k, TLC = 3k
- Initial and runtime bad blocks
- Read errors & disturbance



NAND Flash Controller



Block Base Mapping

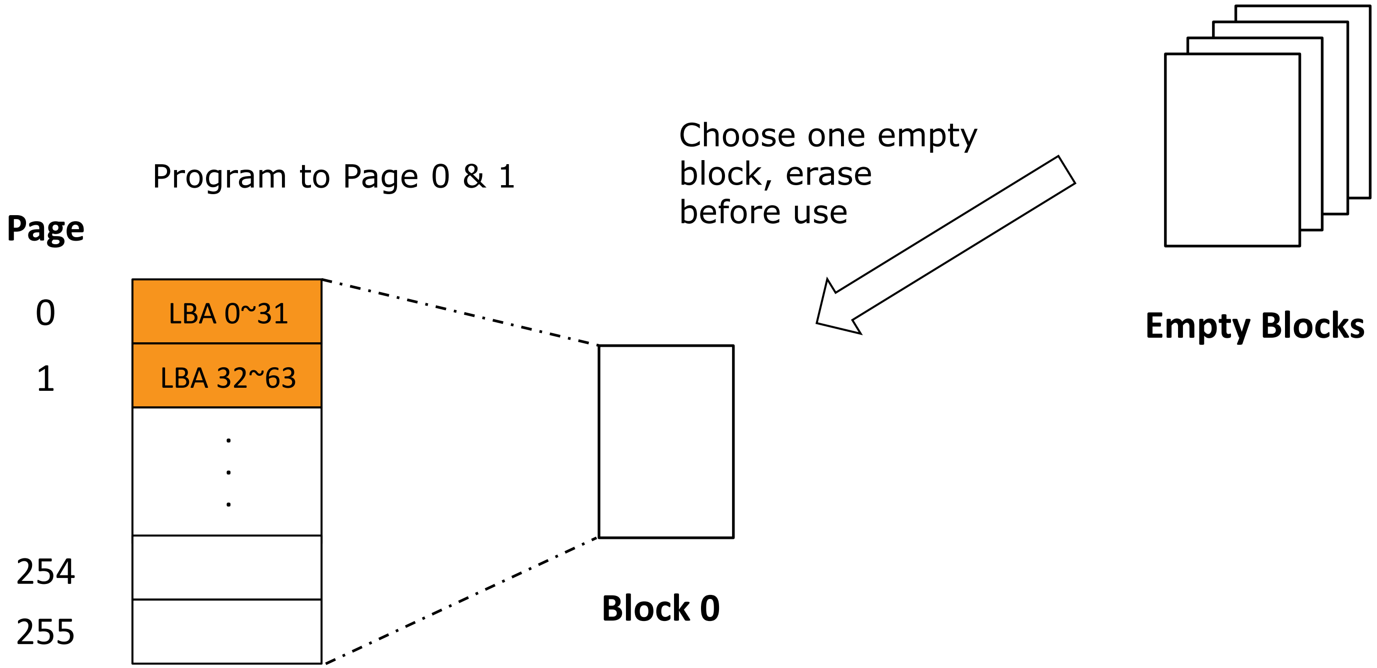
Definitions & Rules

- 8GB NAND Flash
  - 1 die = 2048 blocks
  - 1 block = 256 pages
  - 1 page = 16KB = 32 sectors
  - 1 LBA/sector = 512B

- Limitation
  - **Page** is the smallest unit for read and program
  - **Block** is the smallest unit for erase
  - Must erase before program (cannot overwrite)

Write

- Host writes 2 chunk of data (16KB each)
  - LBA 0~31
  - LBA 32~63

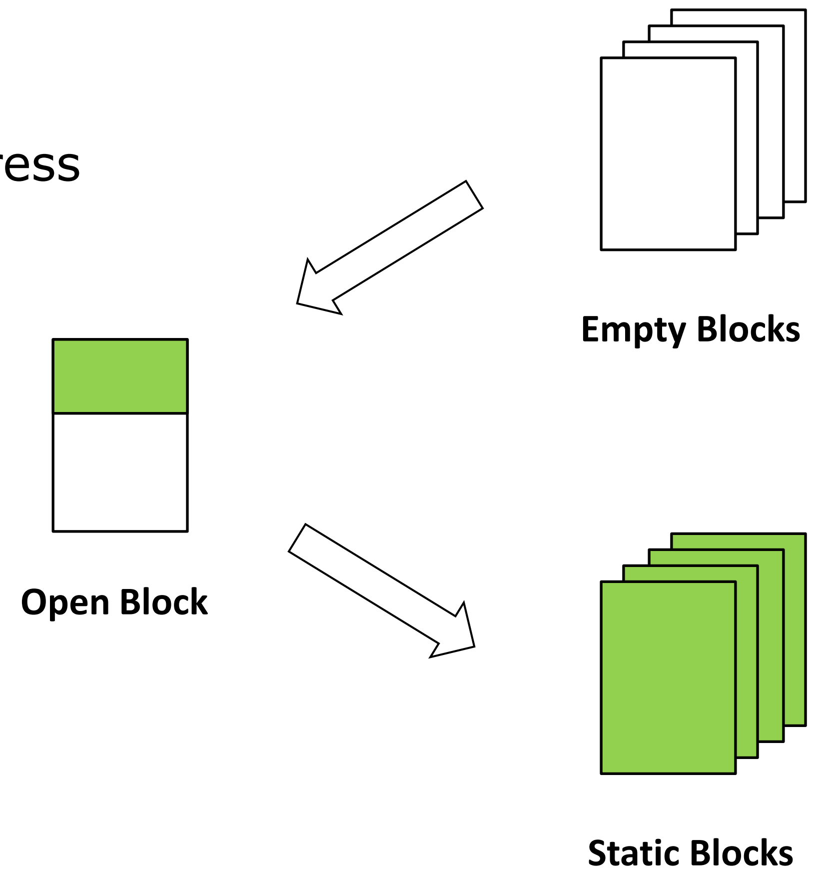


Write

Q: How do we build mapping table?

- Host continues to write large chunk size data…
- Open block becomes static when full
- We record logical to physical block mapping
- Each Logical Block (LB) index maps to Block (PB) address

LB	PB
0	0
1	1
	. . .
2046	0xFFFF
2047	0xFFFF



- Address = (PB)
- Might need 2B to store address

Read

- Host wants to read LBA 32~63
- How do we know where is the data located in NAND Flash?

Read

Q: What is the formula to calculate offset?

- Logical to Physical (L2P) translation

Step 1: Calculate Logical Block (LB) Index

$$LB = \frac{LBA}{\text{Total Physical Sector Number (in block)}}$$

Step 2: Get Physical Block (PB) address from mapping table

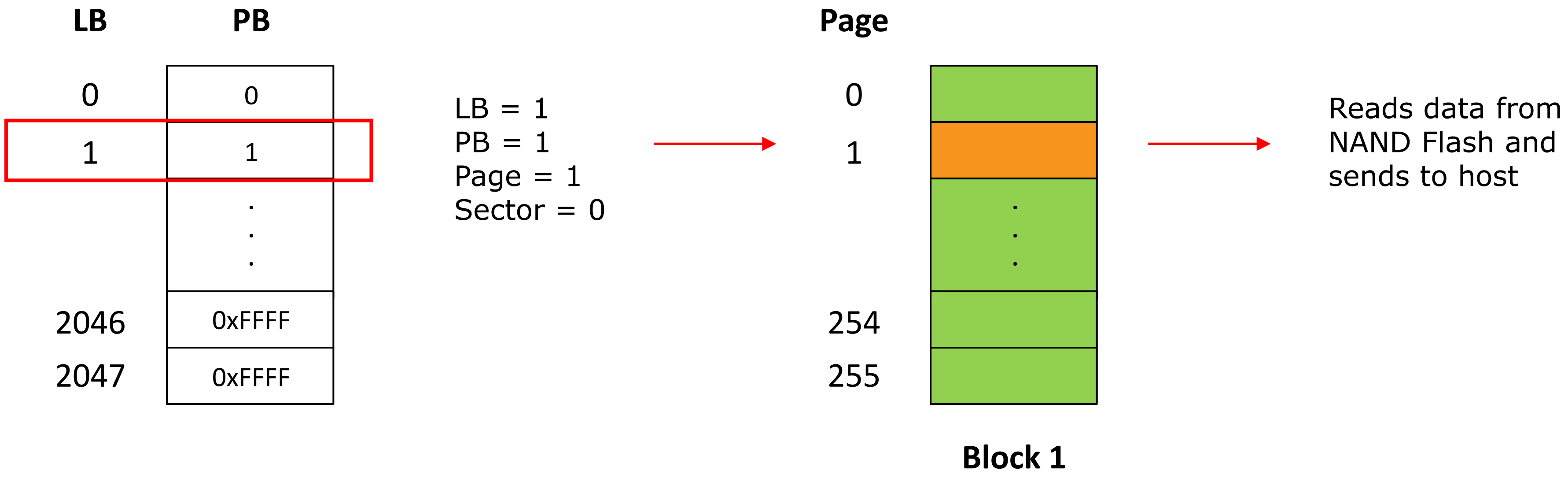
Step 3: Calculate Offset

$$Page = \frac{LBA \,\%\, \text{Total Physical Sector Number (in block)}}{\text{Total Physical Sector Number (in page)}}$$

$$Sector = LBA \,\%\, \text{Total Physical Sector Number (in block)} \,\%\, \text{Total Physical Sector Number (in page)}$$

Read

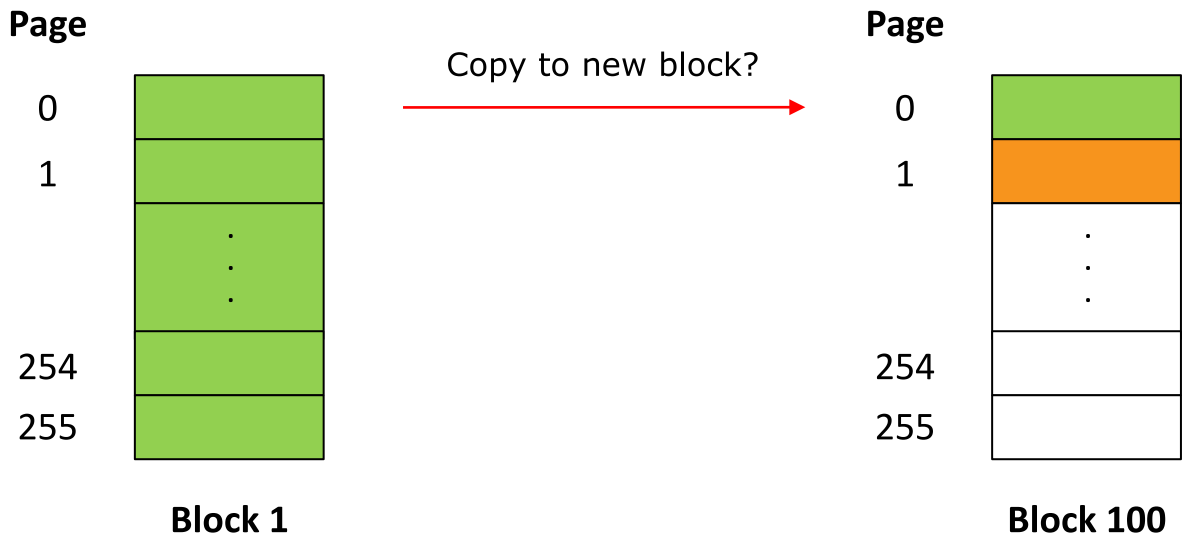
- Host wants to read LBA 8224~8255



Overwrite

Q: How do we handle overwrite page?

- What if host overwrites same address LBA 8224~8255?
- Remember, we cannot overwrite a page that has been programmed



- Not so efficient

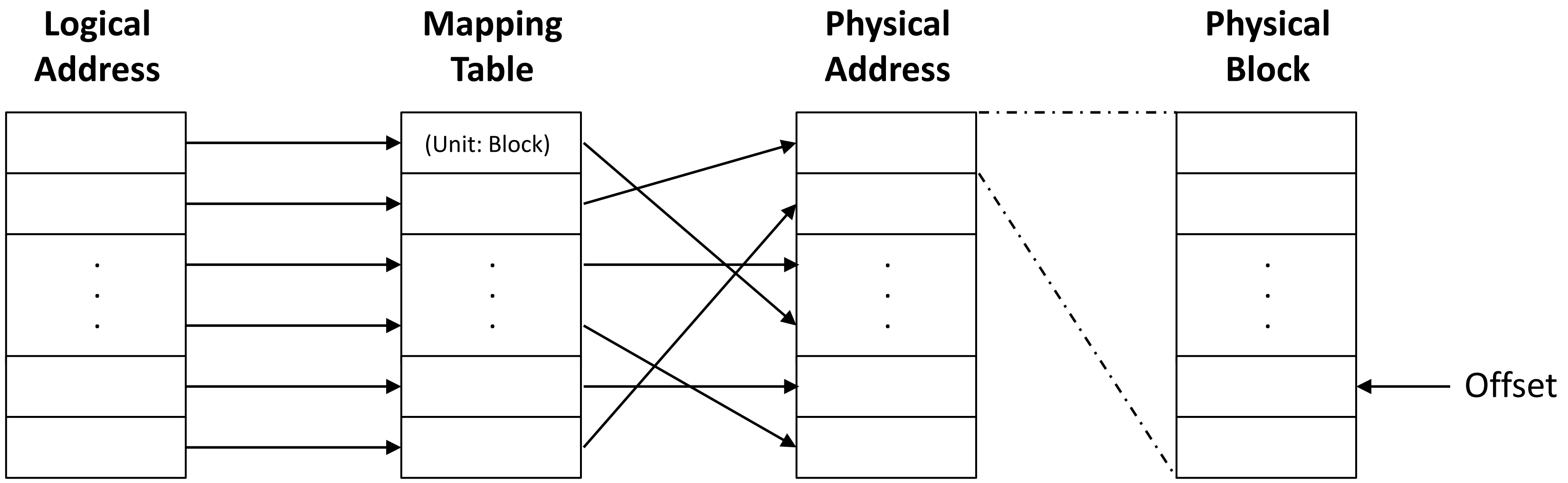
Table Size (for Block Mapping)

- For 8GB NAND Flash, total table size required is 4KB
  - Table Size = 2048 blocks × 2B = 4KB
  - Table Entries = 2048

LB	PB
0	0
1	1
	. . .
2046	0xFFFF
2047	0xFFFF

Block Base Mapping Table

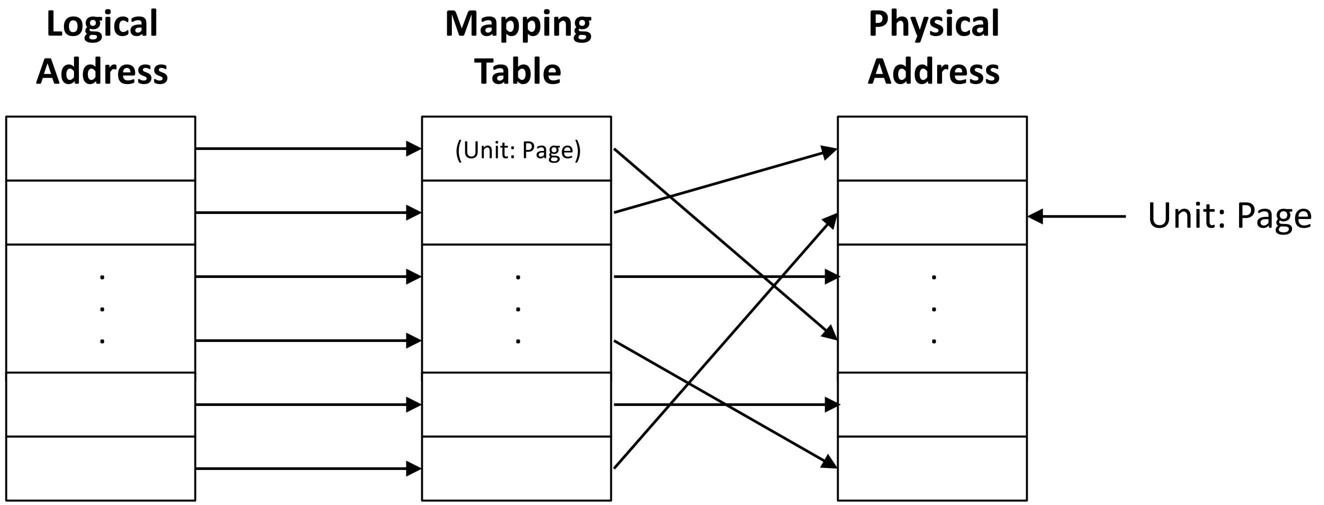
- In a block level address mapping, a logical page address is made up of both a **logical block number** and an **offset**



Page Base Mapping

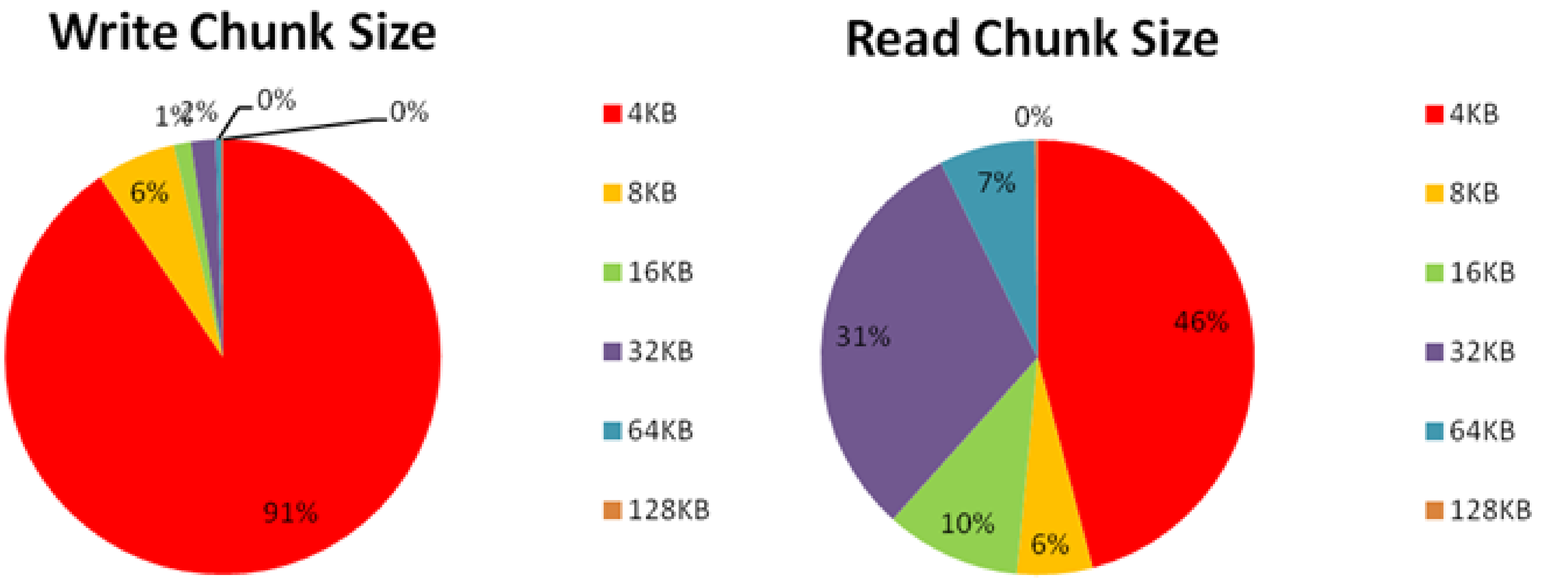
Page Base Mapping Table

- In a page level address mapping, a logical page can be **mapped into any physical page** in flash memory



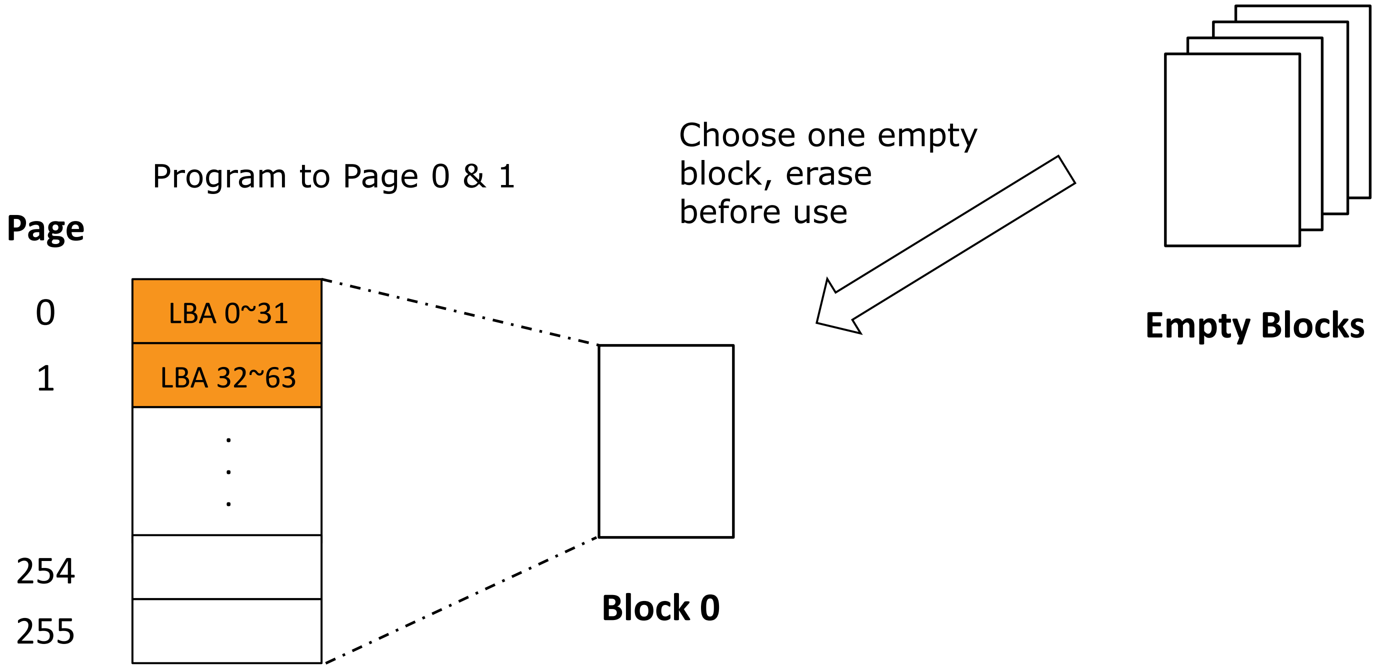
Page Base Mapping Table

- Data structure of host (eMMC) operations on WHCK performance test
- 4KB chunk size has higher percentage than other chunk size



Write

- Host writes 2 chunk of data (16KB each)
  - LBA 0~31
  - LBA 32~63



Write

Q: How do we record page base mapping?

- Instead of block mapping, we now record logical to physical page mapping
- Each Logical Page (LP) index maps to Physical Page (PP) address

LP	PP
0	0, 0, 0
1	0, 1, 0
	. . .
542286	0xFFFFFFFF
542287	0xFFFFFFFF

- Address = (Block, Page, Sector)
- Might need 4B (instead of 2B) to store address

Write

- Host writes additional 2 chunk of data (16KB each)
  - LBA 64~95
  - LBA 32~63 (overwrite)

Page	Block 0
0	LBA 0~31
1	LBA 32~63
2	LBA 64~95
3	LBA 32~63
	. . .
254	
255	

No need to copy data to new block during overwrite, old data becomes invalid

LP	PP
0	0, 0, 0
1	0, 3, 0
2	0, 2, 0
3	0xFFFFFFFF
	. . .
542286	0xFFFFFFFF
542287	0xFFFFFFFF

Read

- Host wants to read LBA 32~63
- How do we know where is the data located in NAND Flash?

Read

Q: What is the formula to calculate Logical Page?

- Logical to Physical (L2P) translation

  Step 1: Calculate Logical Page (LP) Index

$$LP = \frac{LBA}{Total\ Physical\ Sector\ Number\ (in\ page)}$$

  Step 2: Get Physical Page (PP) address from mapping table

Read

- Host wants to read LBA 32~63

LP	PP
0	0, 0, 0
1	0, 3, 0
2	0, 2, 0
3	0xFFFFFFFF
	. . .
542286	0xFFFFFFFF
542287	0xFFFFFFFF

LP = 1
Block = 0
Page = 3
Sector = 0

Page	Block 0
0	LBA 0~31
1	LBA 32~63
2	LBA 64~95
3	LBA 32~63
	. . .
254	
255	

Block 0

Reads data from NAND Flash and sends to host

Erase

Q: How do we erase LBA?

- Host wants to erase LBA 64~95

LP	PP
0	0, 0, 0
1	0, 3, 0
2	0xFFFFFFFF
3	0xFFFFFFFF
. . .	. . .
542286	0xFFFFFFFF
542287	0xFFFFFFFF

LP = 2
Block = 0
Page = 2
Sector = 0

Page	Block 0
0	LBA 0~31
1	LBA 32~63
2	LBA 64~95
3	LBA 32~63
. . .	. . .
254	
255	

- Mark corresponding LP as invalid

Table Size (for 16KB Page Mapping)

- For 8GB NAND Flash, total table size required is 2MB
  - Table Size = 2048 blocks × 256 pages × 4B = 2MB
  - Table Entries = 2048 blocks × 256 pages = 524288

LP	PP
0	0xFFFFFFFF
1	0xFFFFFFFF
	⋮
542286	0xFFFFFFFF
542287	0xFFFFFFFF

Table Size (for 4KB Page Mapping)

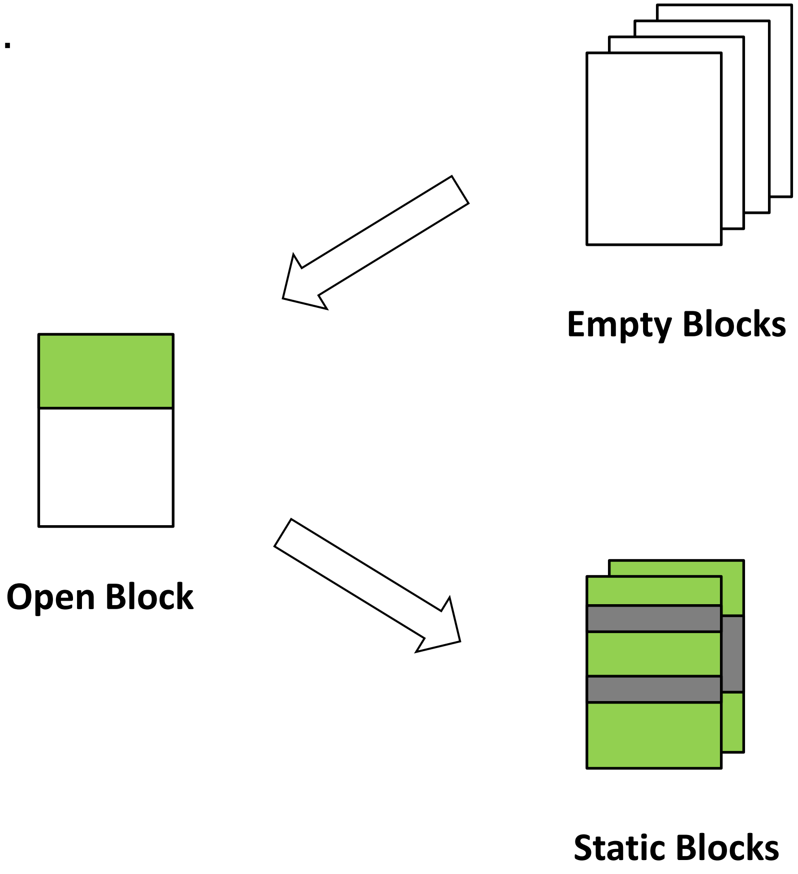
- For 8GB NAND Flash, total table size required is 8MB
  - Table Size = 2048 blocks × 256 pages × 4 nodes × 4B = 8MB
  - Table Entries = 2048 blocks × 256 pages × 4 nodes= 2097152

LP	PP
0	0xFFFFFFFF
1	0xFFFFFFFF
	⋮
2097150	0xFFFFFFFF
2097151	0xFFFFFFFF

Garbage Collection (GC)

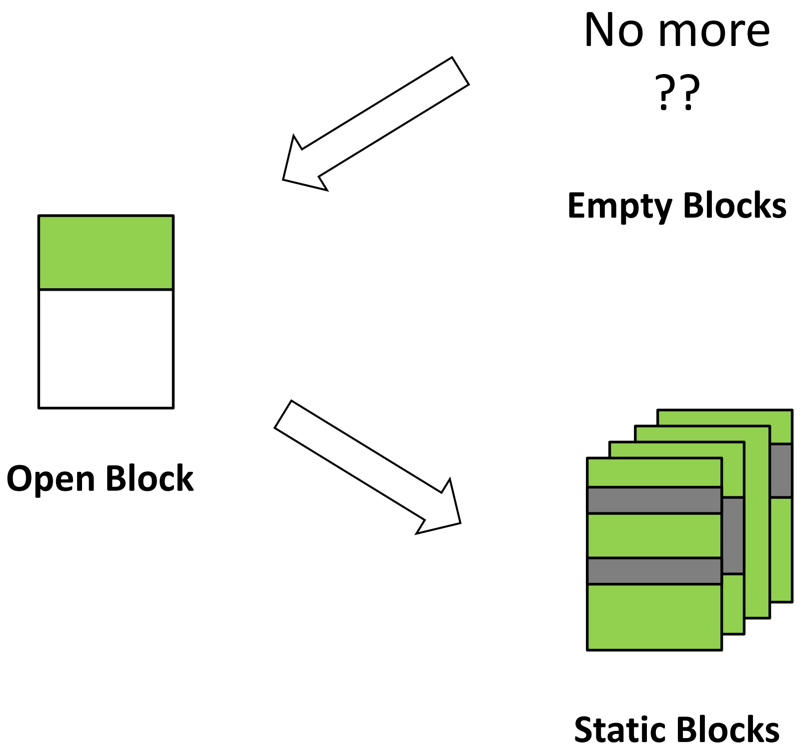
Garbage Collection

- What is GC and why do we need to do GC?
- Host continues to write large chunk size data…
- Open block becomes static when full
- Some physical page become invalid



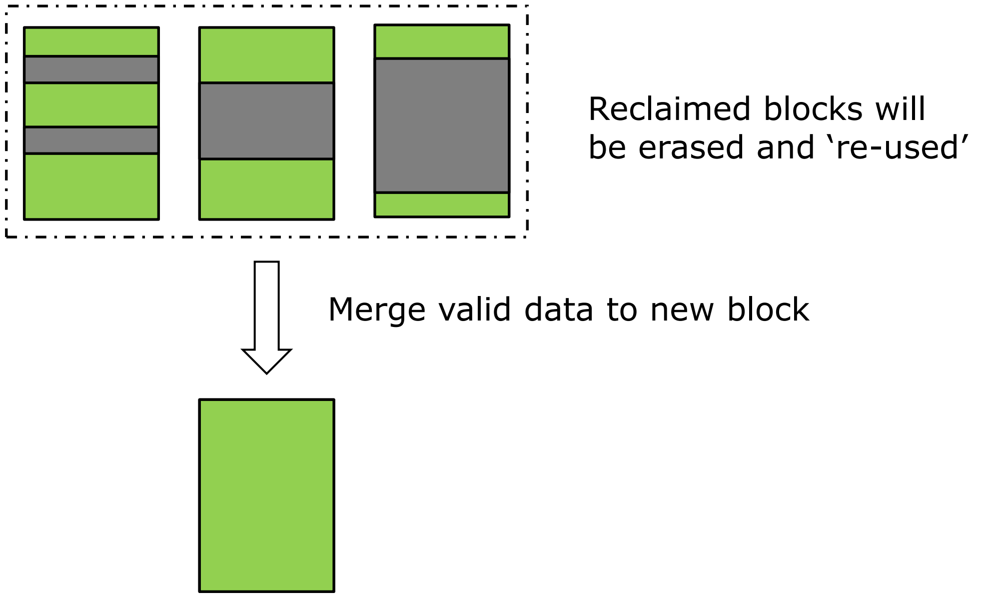
Garbage Collection

- What if host keeps writing until empty block runs out?
- Before empty block runs out, we should do Garbage Collection (GC)



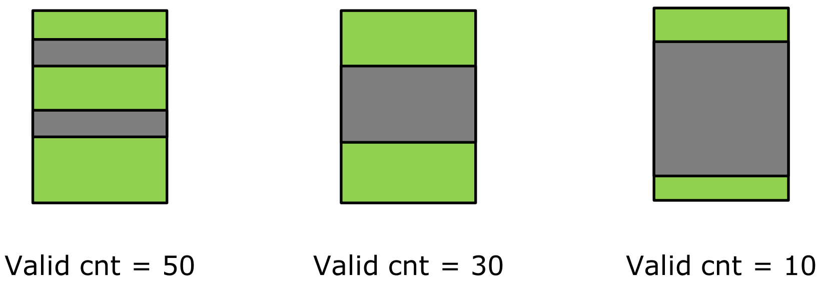
Garbage Collection

- We collect valid data to new block and reclaim blocks filled with invalid data
- So that we can erase the reclaimed blocks and use them for new data

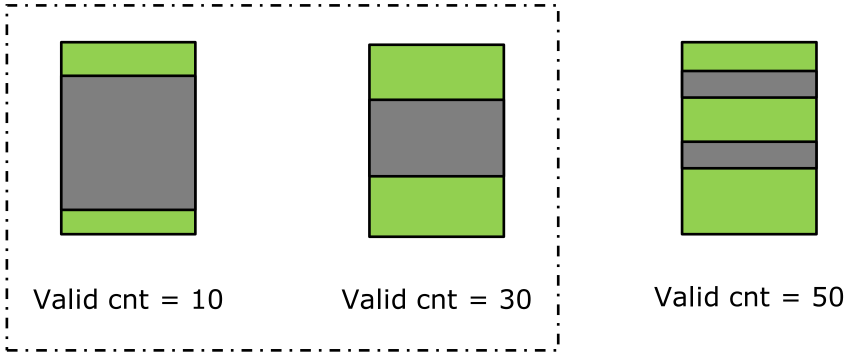


Garbage Collection

- How do we pick static blocks as source of GC?



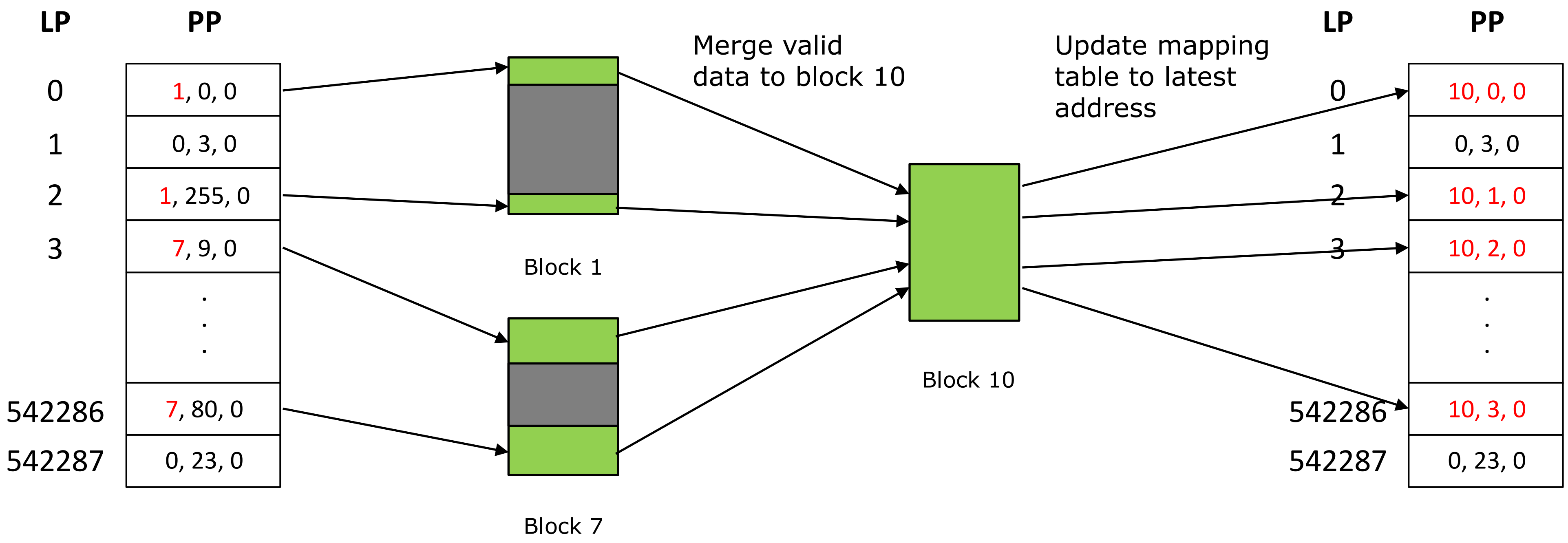
- Pick the block(s) with least valid count



Garbage Collection

Q: What should we update to complete GC?

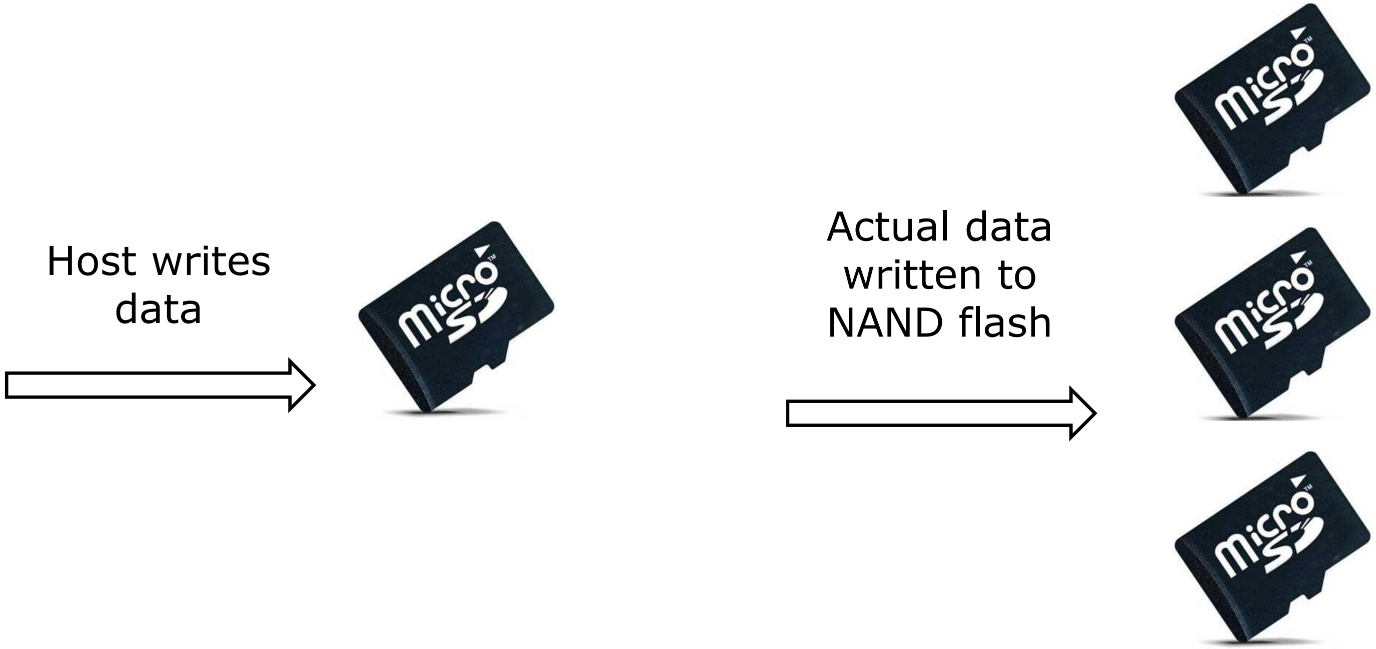
- How do we know which data is valid or invalid in the source blocks?
- Check the mapping table, look for the entries that point to these source blocks



Write Amplification Factor (WAF)

Definition

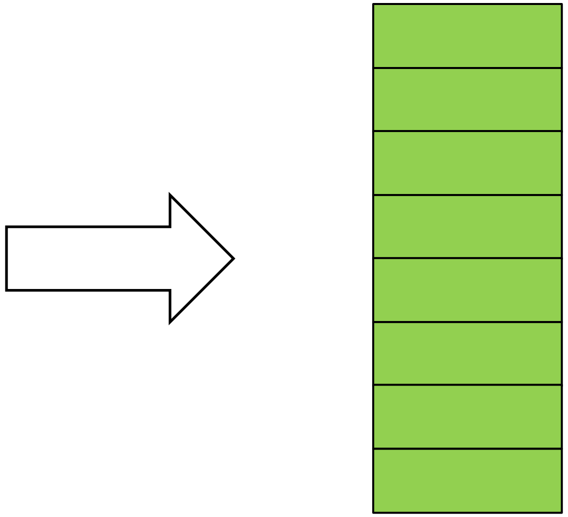
$$\textit{Write Amplification Factor (WAF)} = \frac{\textit{Data written to NAND Flash}}{\textit{Data written by host}}$$



Example

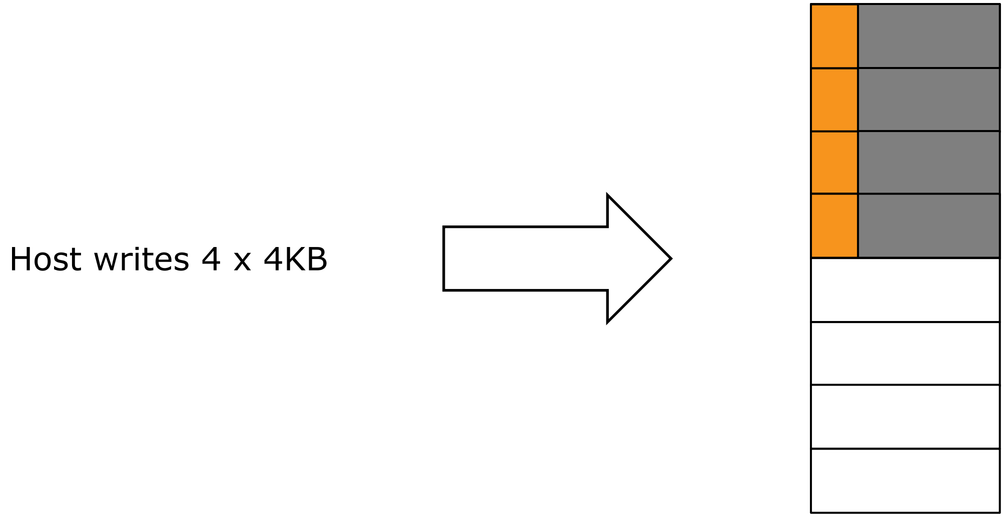
- WAF = 1

Host writes large sequential data



Example

- WAF = 4



- Still remember what happens when host overwrites data of same LBA using block level mapping?

PHISON

Summary

- Disadvantages of high WAF
  - Low performance
  - High erase count

Terabytes Written (TBW)

Q: What is the formula to calculate TBW?

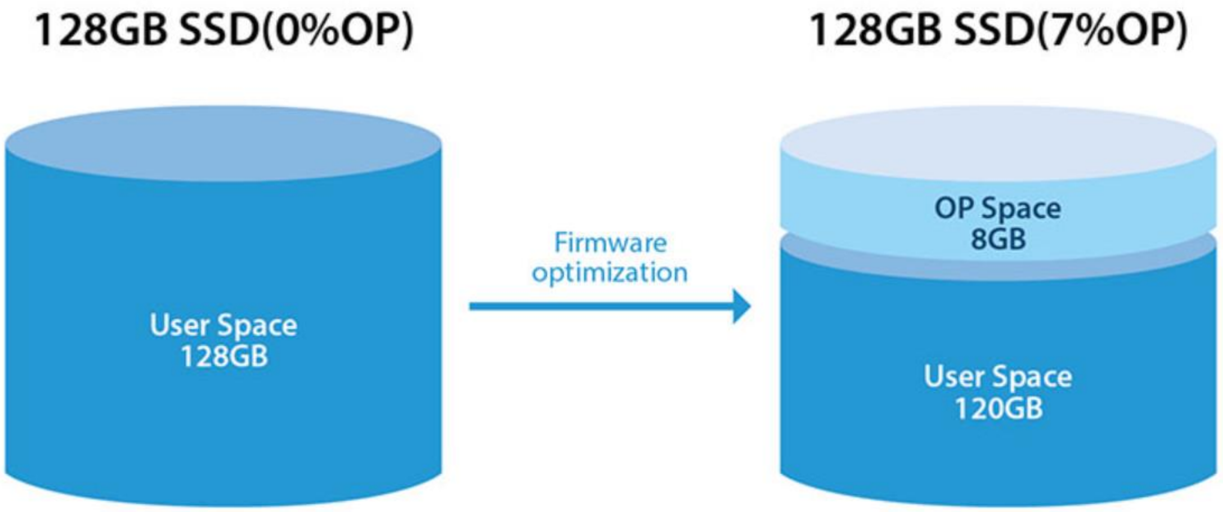
- Total amount of data that can be written to a storage device until it reaches its lifetime

$$\textit{Terabytes Written (TBW)} = \frac{\textit{User Capacity (GB)} \times \textit{NAND P/E Cycles}}{\textit{WAF} \times 1024}$$

Over Provision (OP)

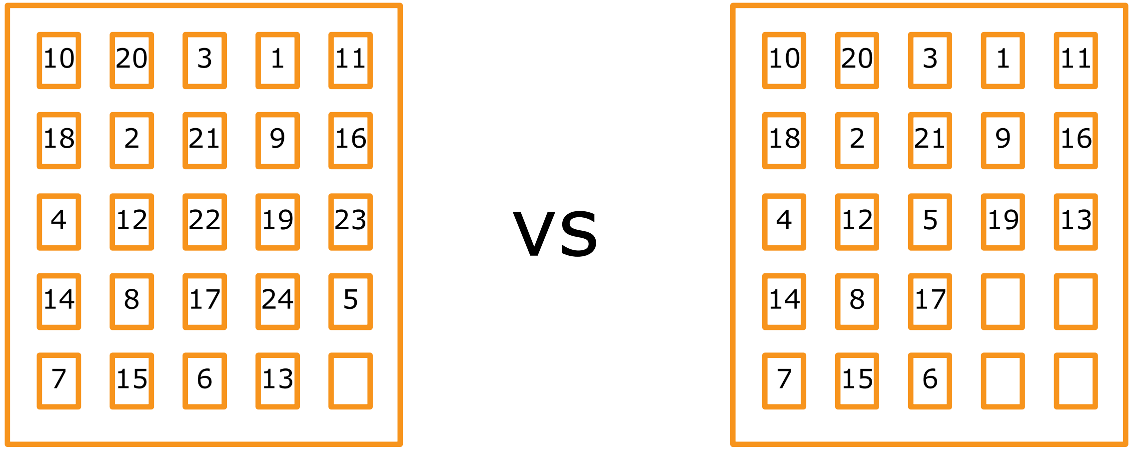
Definition

$$\text{Over-Provision} = \frac{\text{Physical capacity} - \text{User capacity}}{\text{User capacity}}$$



Example

- The larger the size of the spare area, the higher the operating efficiency, and the better the performance become



The Sliding Puzzle

OP vs WAF

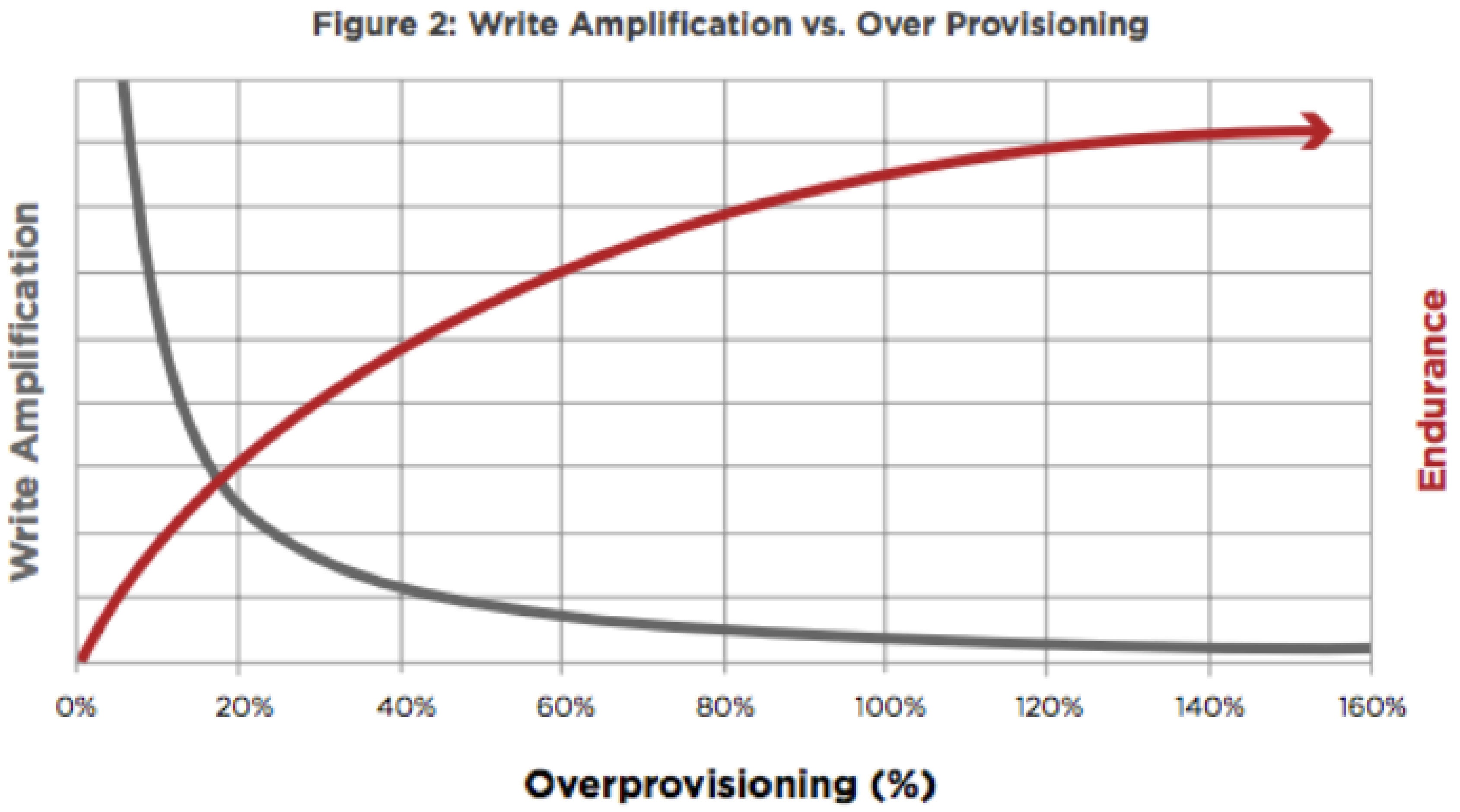


Figure 2: Write Amplification vs. Over Provisioning
Write Amplification
Endurance
0%
20%
40%
60%
80%
100%
120%
140%
160%
Overprovisioning (%)

Summary

- Advantage(s) of higher OP due to less background data movement required
  - Higher performance
  - Better endurance (lower WAF)
- Disadvantage(s) of higher OP due to more reserved spare blocks
  - Less usable storage space

Summary

NAND Flash Limitation

- **Page** is the smallest unit for read and program
- **Block** is the smallest unit for erase
- Must erase before program (cannot overwrite)

Page Base vs Block Base

Q: Page vs Block Base, which is better?

	Page Base	Block Base
Mapping Unit	Page (or 4KB)	Block
Table Size	Large (store in NAND Flash)	Small (store in SRAM)
R/W Performance	Excellent random write performance	Slow random write performance, but impressive read performance
Garbage Collection	Collect valid nodes when empty block becomes insufficient	Collect valid nodes when overwrite previous data or erase data
WAF	Low (efficient block utilization)	High (expensive merge operation)

Cold Facts

- Do you know that your storage device with higher OP (less user space) has better performance and lifetime?
- Do you know that when you keep your storage usage full, the performance and lifetime become worst (due to frequent GC operation)?
- When purchasing storage device, consider performance vs lifetime (such as WAF or TBW)

Advanced Questions

- What if power cycle occurs when we program NAND Flash?
- What if bad block occurs when we program NAND Flash?
- What if read error occurs when we read NAND Flash?
- What if certain blocks wear out quickly than other blocks?

FTL

Table Management

Garbage Collection

Read Disturb Management

Wear Leveling

Bad Block Management

Power-up Rebuild

THANK YOU!