Flash Translation Layers

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

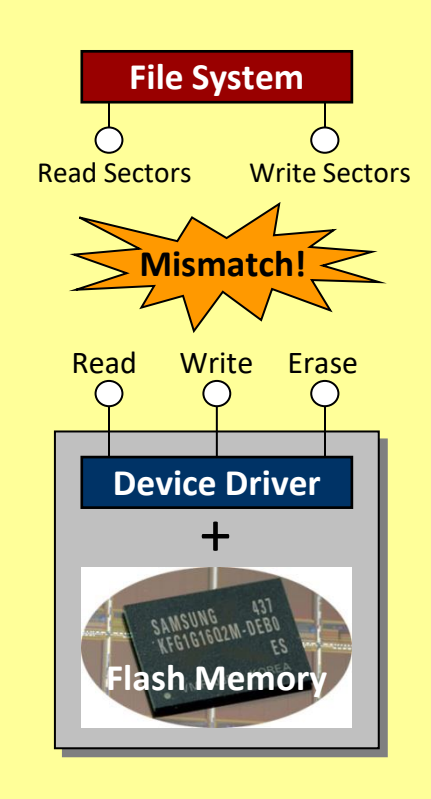
Systems Software &
Architecture Lab.

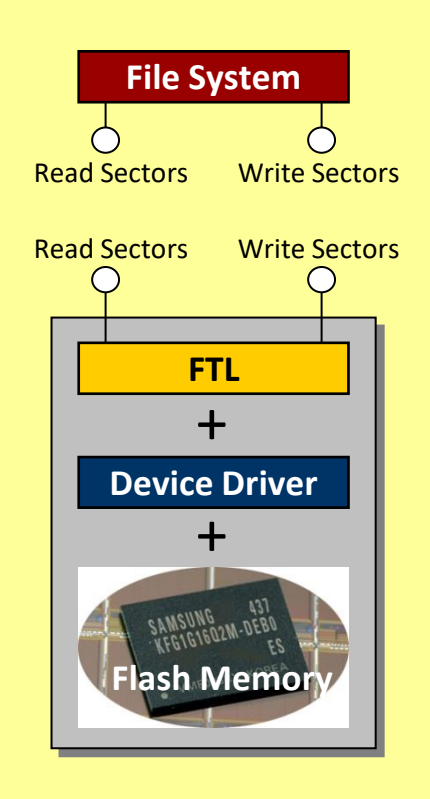
Seoul National University

Spring 2024

What is FTL?

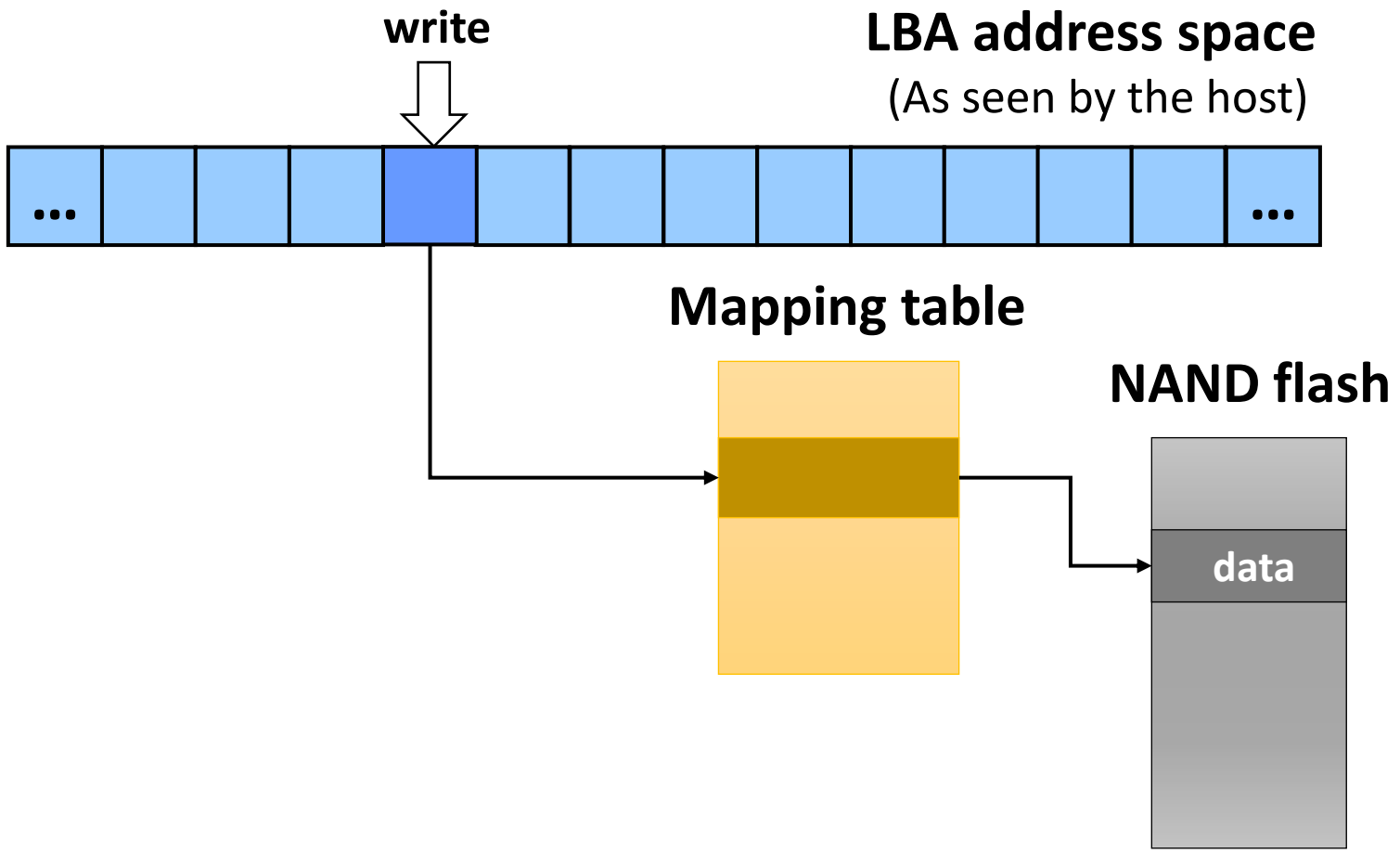
- A software layer to make NAND flash fully emulate traditional block devices (or disks)





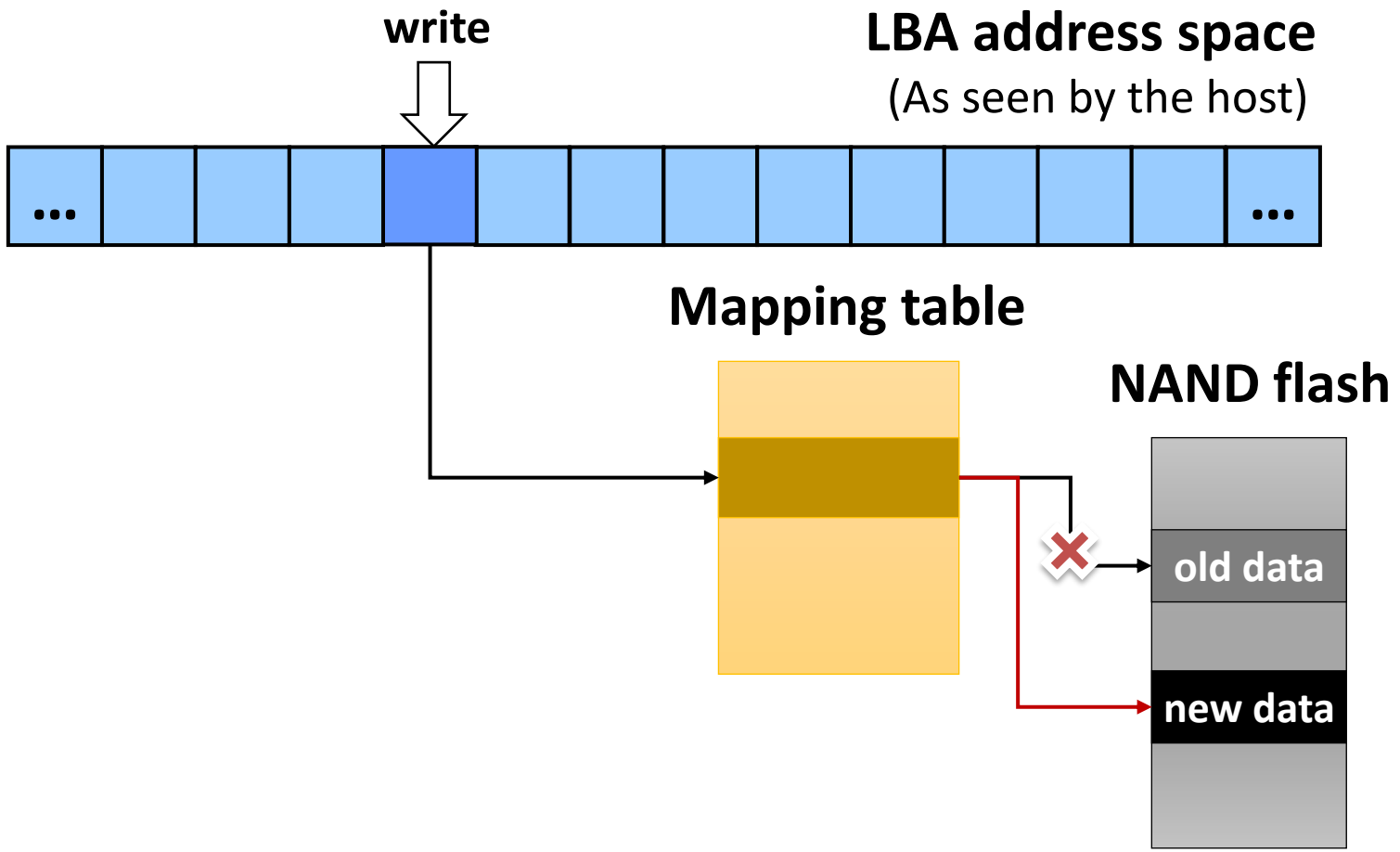
Source: Zeen Info. Tech.

Address Mapping



Address Mapping

- Required due to "no overwrite" characteristic



Plethora of FTLs

HFTL
SAST
SFTL
MS FTL
BPLRU
BFTL
AFTL
FAST
LazyFTL
KAST
Chameleon
CNFTL
DFTL
LAST
MNFTL
super-block scheme
CFTL
Log block scheme
GFTL
µ-FTL
JFTL
zFTL
Hydra FTL
Vanilla FTL
Replacement block scheme
YanusFTL
Reconfigurable FTL
.....and so on
WAFTL
UFTL

E. H. Nam, HIL: FTL Design Framework with Provably-correct Crash Recovery, NVRAMOS, 2013.

Mapping Schemes

- Page mapping
  - Fine-granularity page-level map table
  - Hugh amount of memory space required for the map table
- Block mapping
  - Coarse-granularity block-level map table
  - Small amount of memory space required for the map table
- _____ mapping
  - Use both page-level and block-level map tables
  - Higher algorithm complexity

Page Mapping FTL

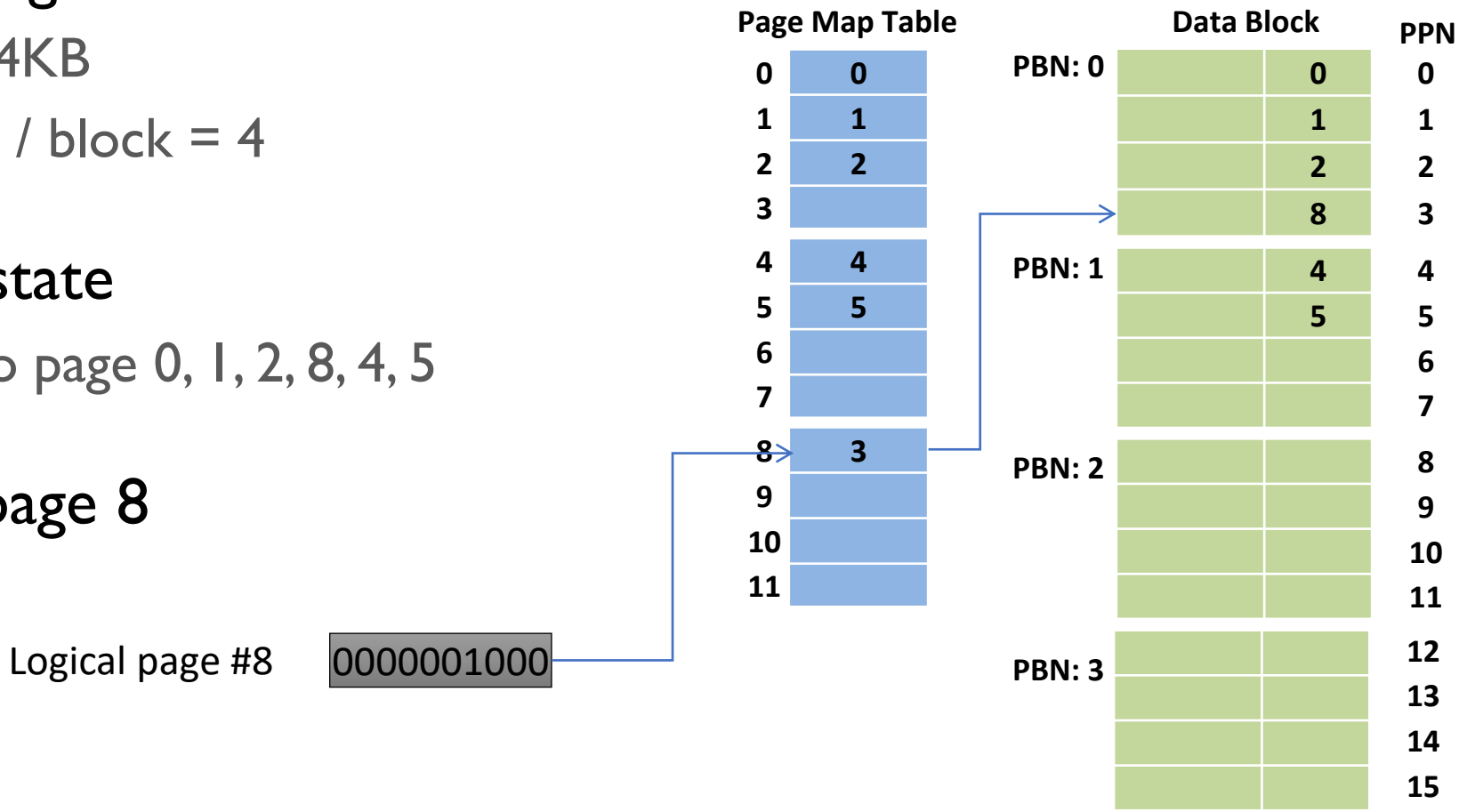
Page Mapping

- Mapping in page-level
  - Logical page number → physical page number
  - Page mapping table (PMT) required
  - # entries in PMT == # pages visible to OS

- Translation
  - Step 1: logical sector number (LSN) → logical page number (LPN)
  - Step 2: LPN → physical page number (PPN) via PMT

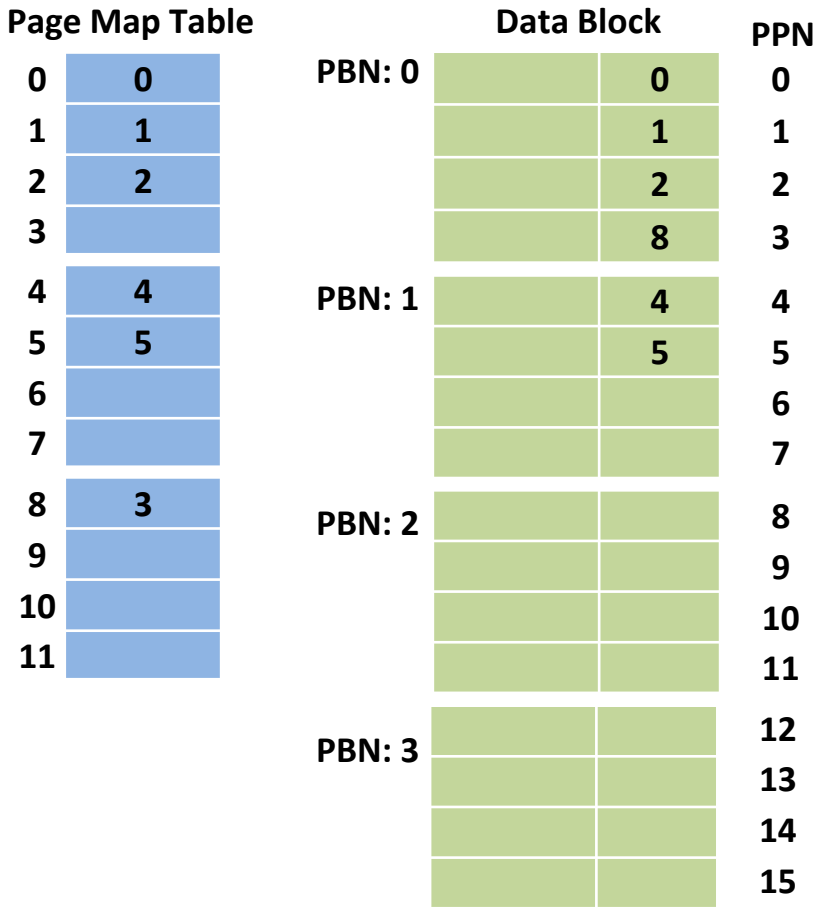
Example: Page Mapping

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



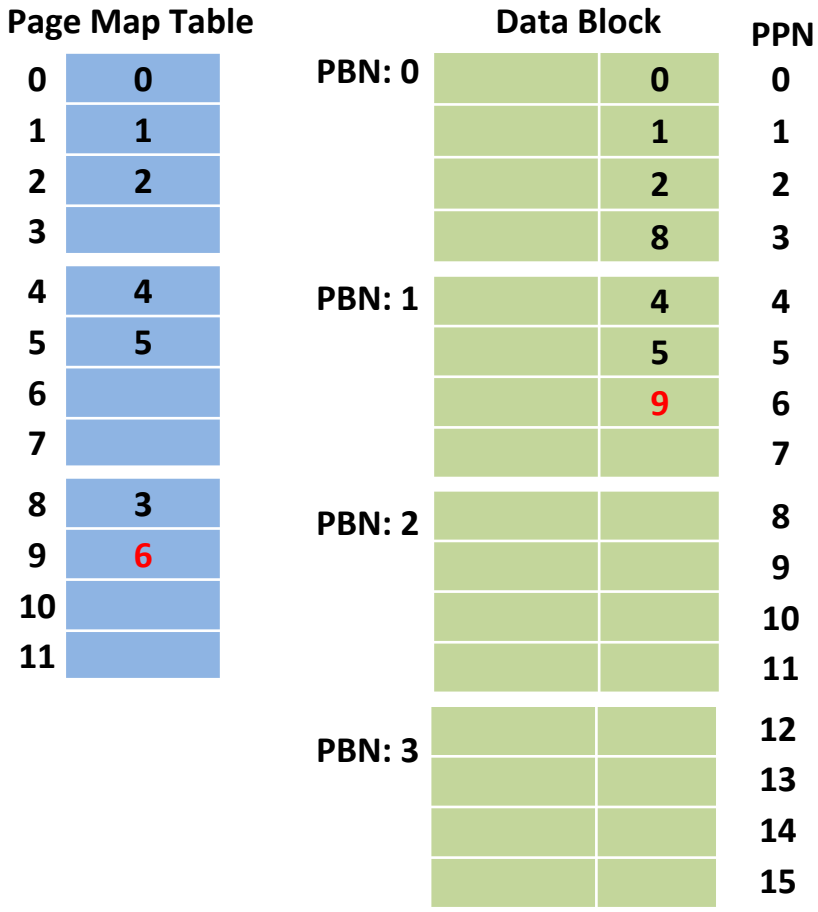
Example: Page Mapping

- 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



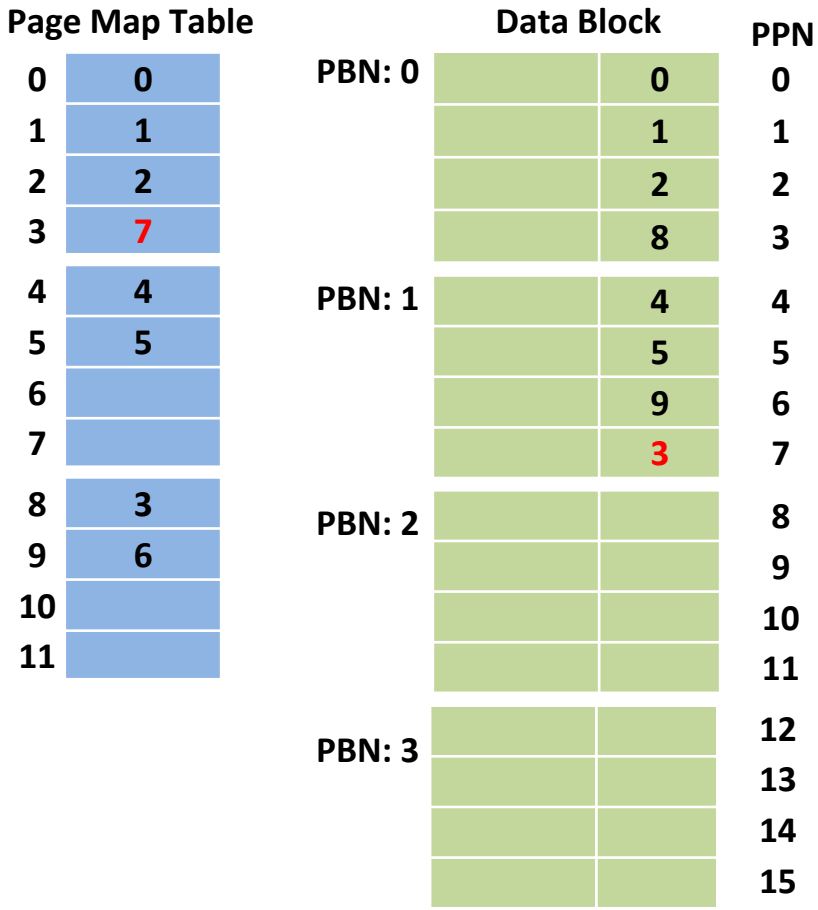
Example: Page Mapping

- 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



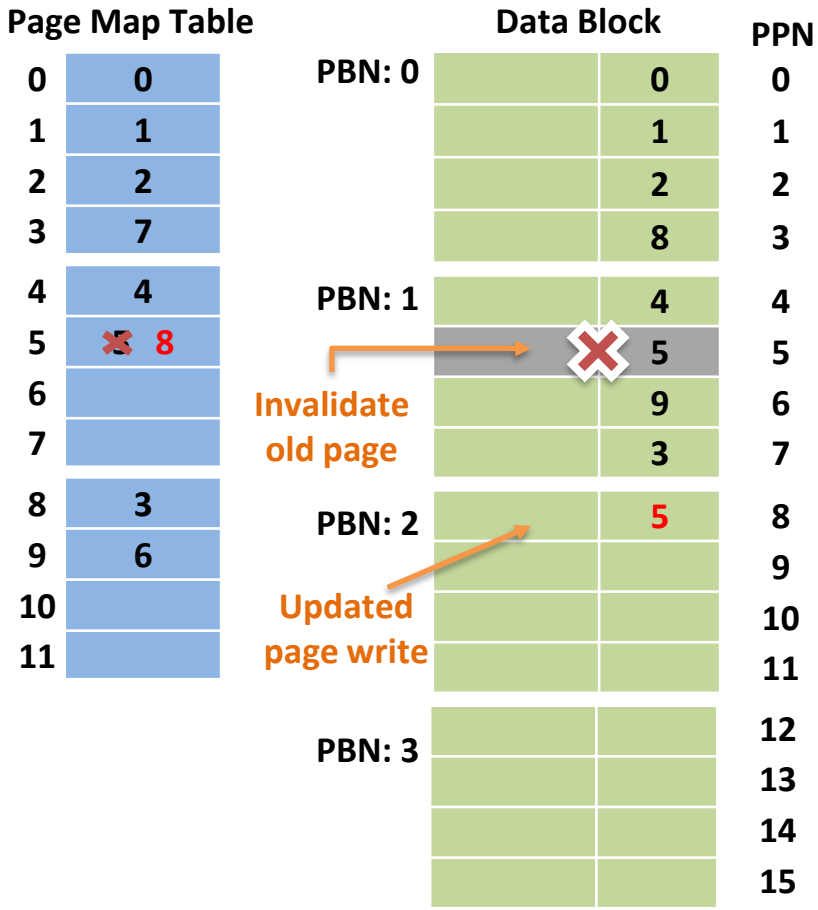
Example: Page Mapping

- 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

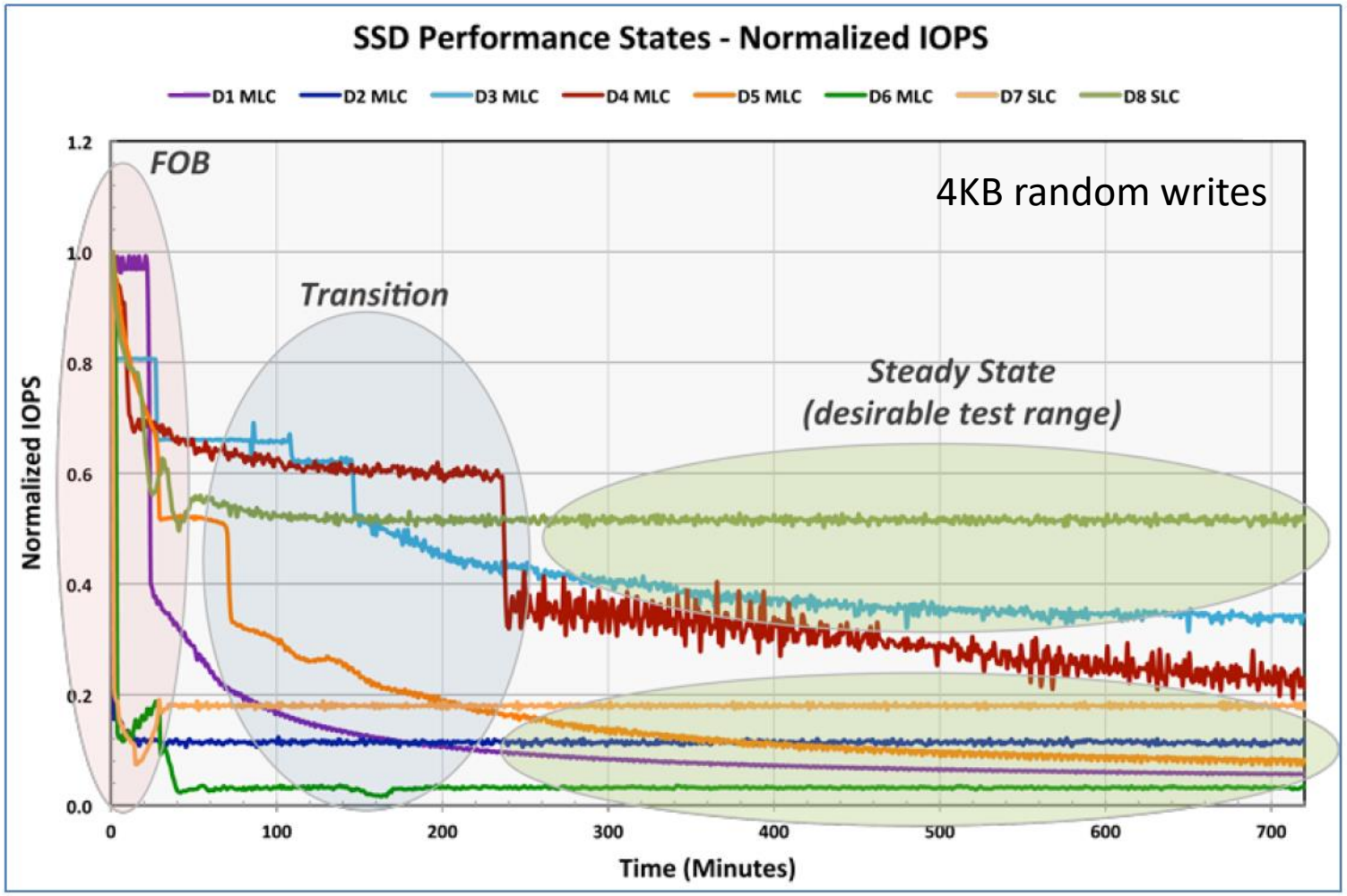
- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
- Current state
  - Written to page 0, 1, 2, 8, 4, 5
- New requests (in order)
  - Write to page 9
  - Write to page 3
  - Write to page 5



Page Mapping

- Pros
  - Most flexible
  - Efficient handling of small random writes
    - A logical page can be located anywhere within the flash storage
    - Updated page can be written to any free page
- Cons
  - Large memory footprint
    - One page mapping entry per page
    - 32MB for 32GB (4KB page)
  - Sensitive to the amount of reserved blocks
  - Performance affected as the system ages

Why?



http://tfindelkind.com/2015/08/20/

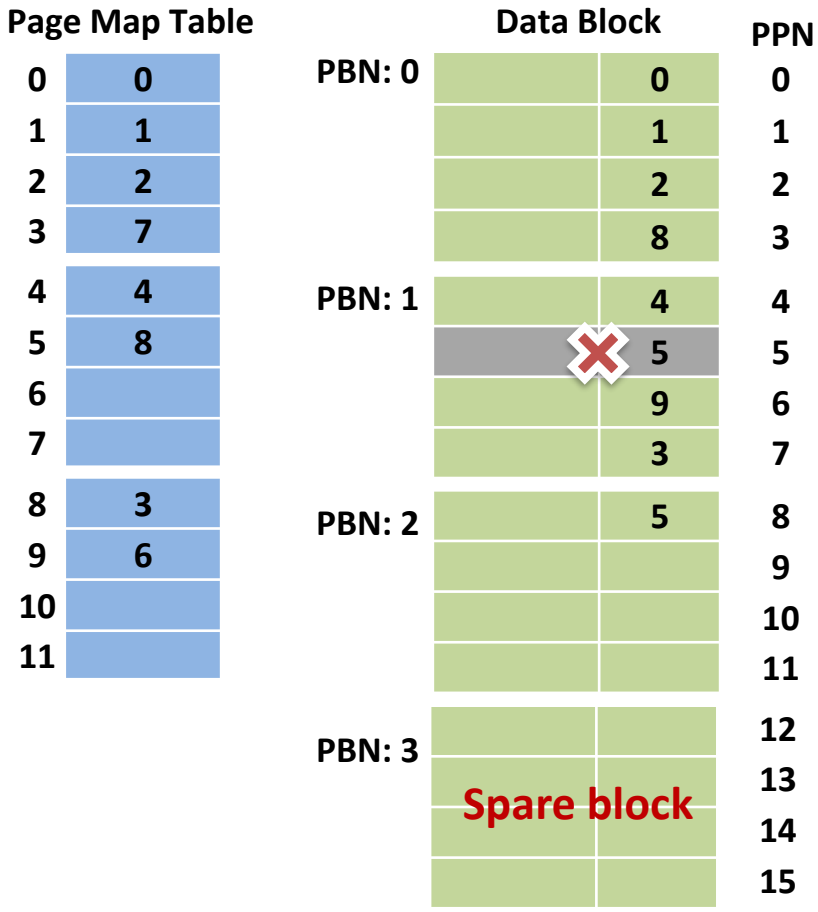
Garbage Collection

- 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

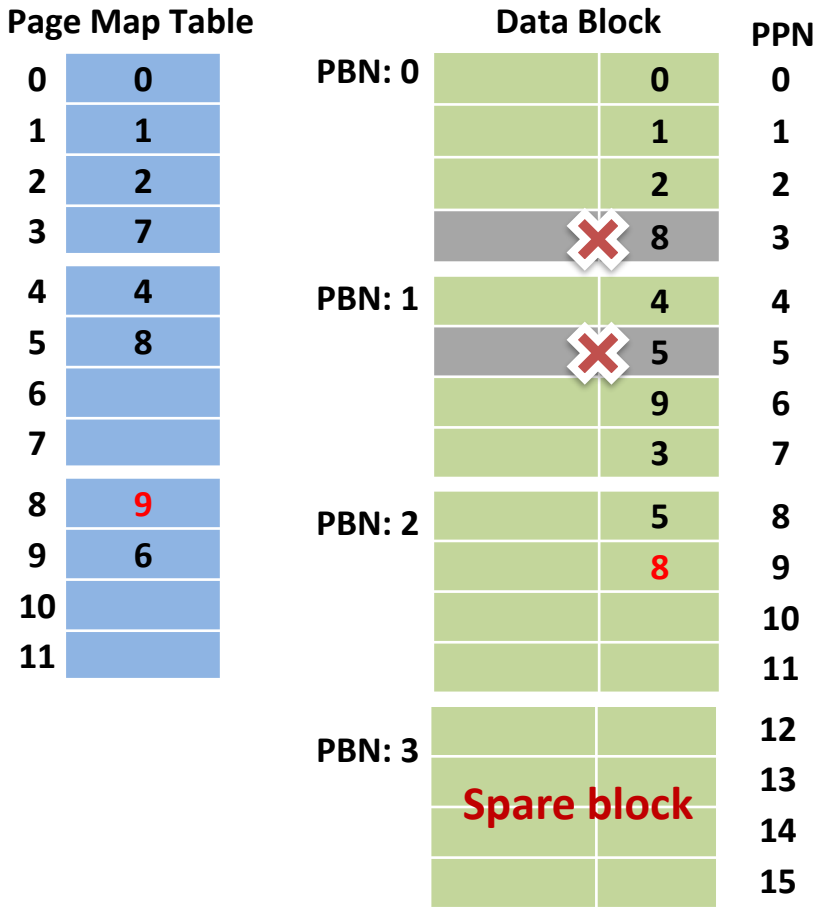
Example: GC in Page Mapping

- 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



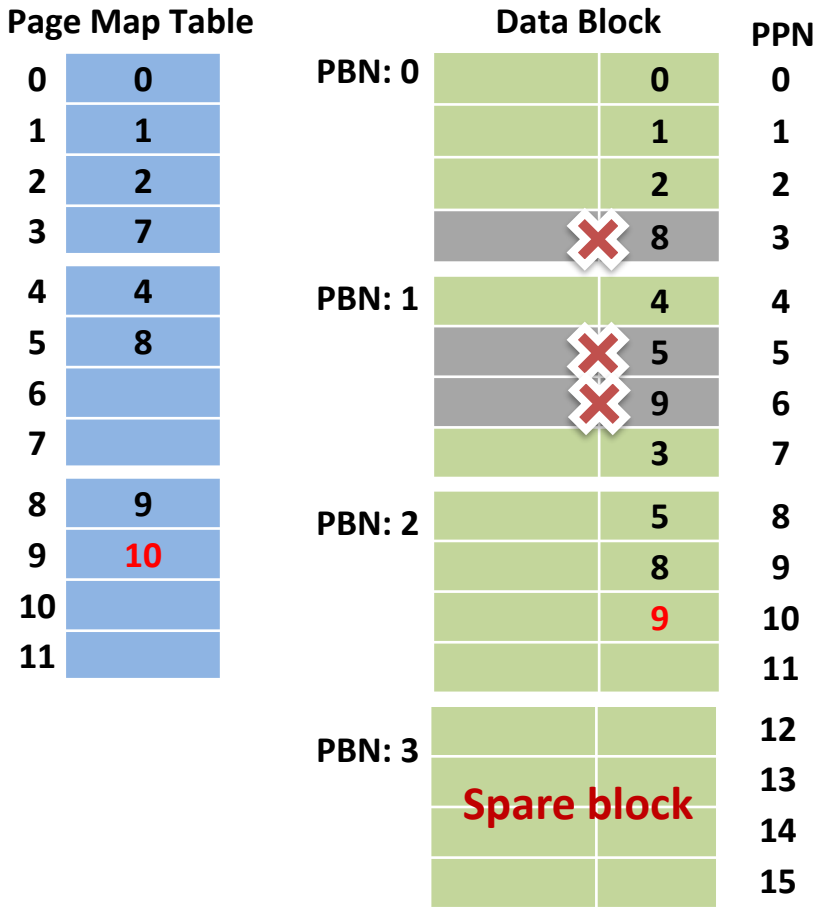
Example: GC in Page Mapping

- 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



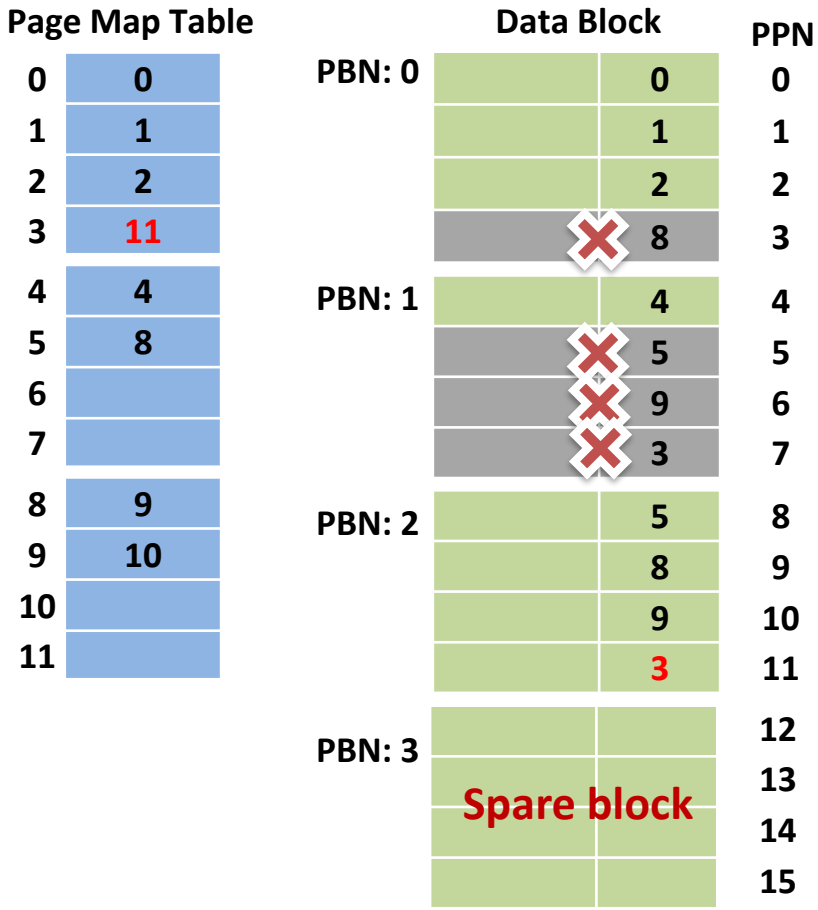
Example: GC in Page Mapping

- 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



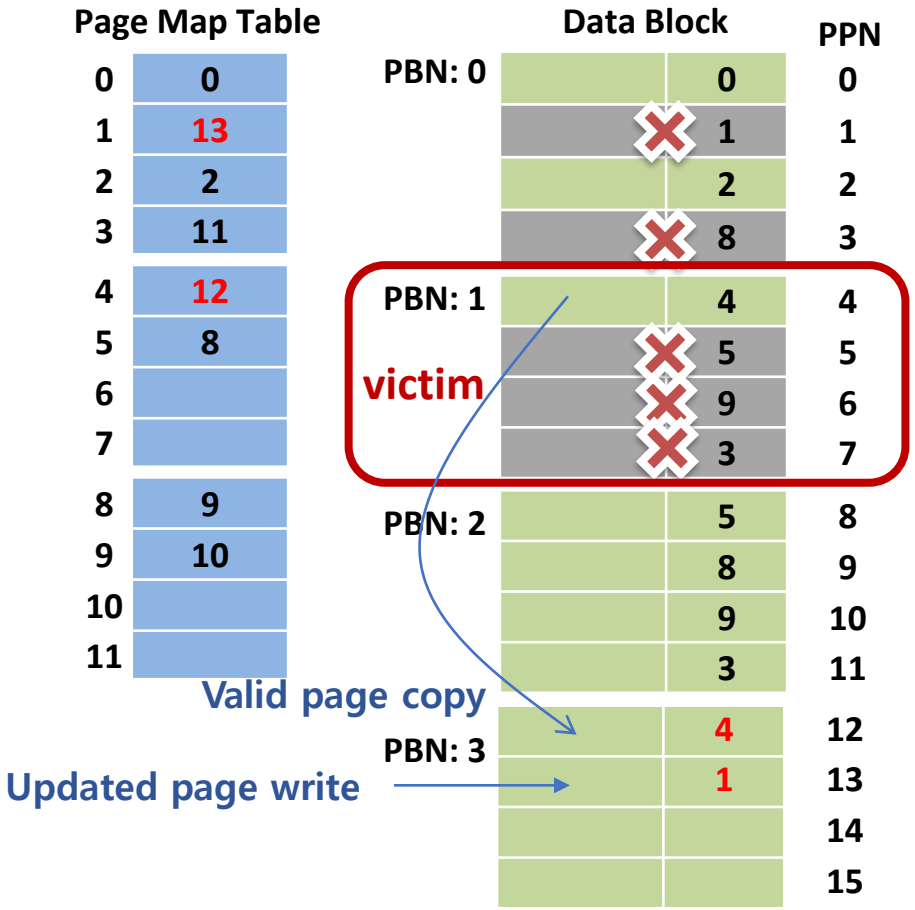
Example: GC in Page Mapping

- 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



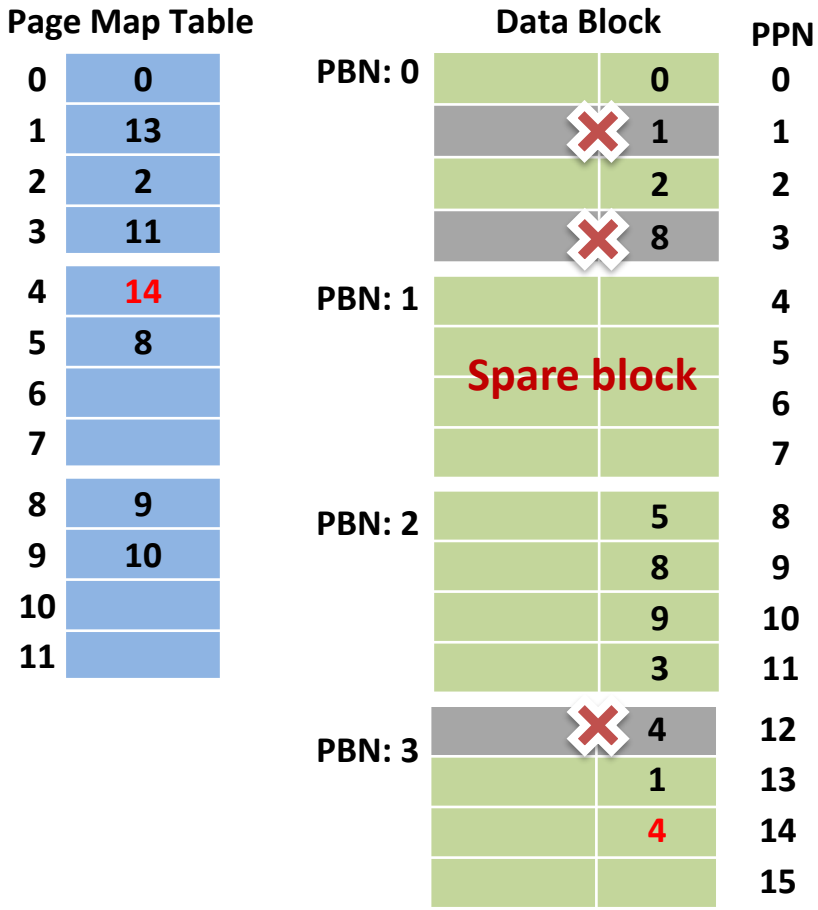
Example: GC in Page Mapping

- 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



Write Amplification

- Ratio of data written to flash to data written from host

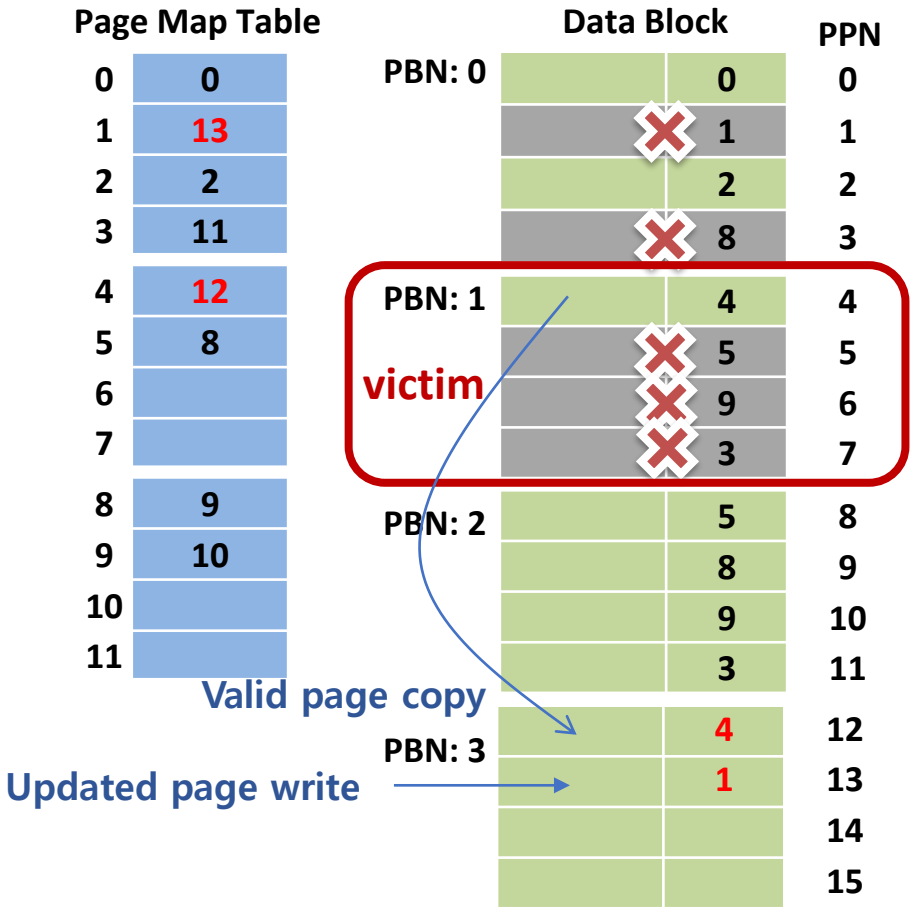
- Write Amplification Factor (WAF)

$$= \frac{Bytes\ written\ to\ Flash}{Bytes\ written\ from\ Host} = \frac{Bytes\ written\ from\ Host + Bytes\ written\ during\ GC}{Bytes\ writen\ from\ Host}$$

- Generally, WAF is greater than one in flash storage
  - Due to valid page copies made from victim block to free block during GC
  - WAF is one of metrics that shows the efficiency of GC

Example: Write Amplification

- 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, 9, 3, 1
- WAF = 1.08
  - Total host writes: 13
  - Total flash writes: 14



Victim Selection Policy: Greedy

- Selects a block with the largest amount of invalid data
- A block with the _____ utilization *u*

$$u = \frac{Number\ of\ valid\ pages\ in\ a\ block}{Number\ of\ Pages\ in\ a\ block}$$

- ***Pros?***

- ***Cons?***

Victim Selection Policy: Cost-Benefit

- Selects a block with the _____

$$\frac{Benefit}{Cost} = \frac{(1-u)}{2u} \times age$$

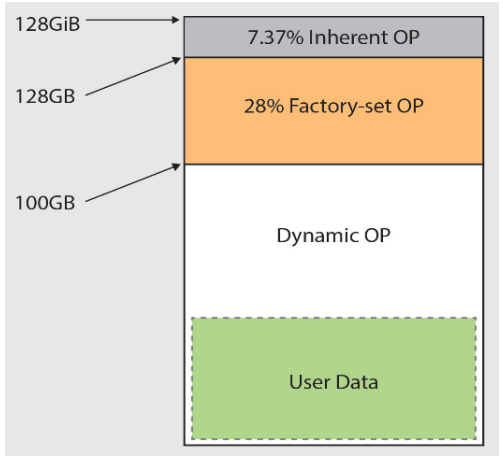
  - u: utilization
  - age: the time since the last modification

- ***Pros?***

- ***Cons?***

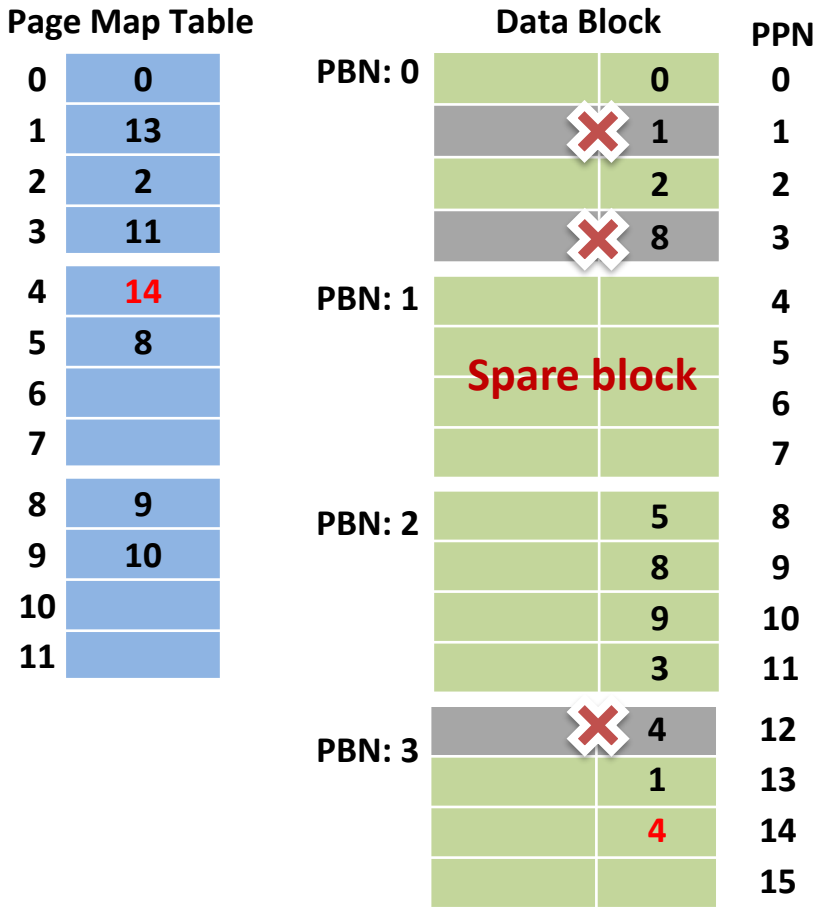
Over-Provisioning

- OP (Over-Provisioning) = $\frac{Physical\ Capacity}{Logical\ Capacity} - 1$
  - Extra media space on an SSD that does not contain user data
- Typical SSDs have more space than is advertised
  - Consumer SSDs: ~ 7%
    - 1 Gigabyte (GB) = 10^9 bytes = 1,000,000,000 bytes
    - 1 Gibibyte (GiB) = 2^{30} bytes = 1,073,741,824 bytes
  - Enterprise SSDs: > 25%
    - 100GB user space on 128GiB SSD: 37.4%



Example: Over-Provisioning

- OP = 33%
  - Logical capacity: 3 blocks
  - Physical capacity: 4 blocks

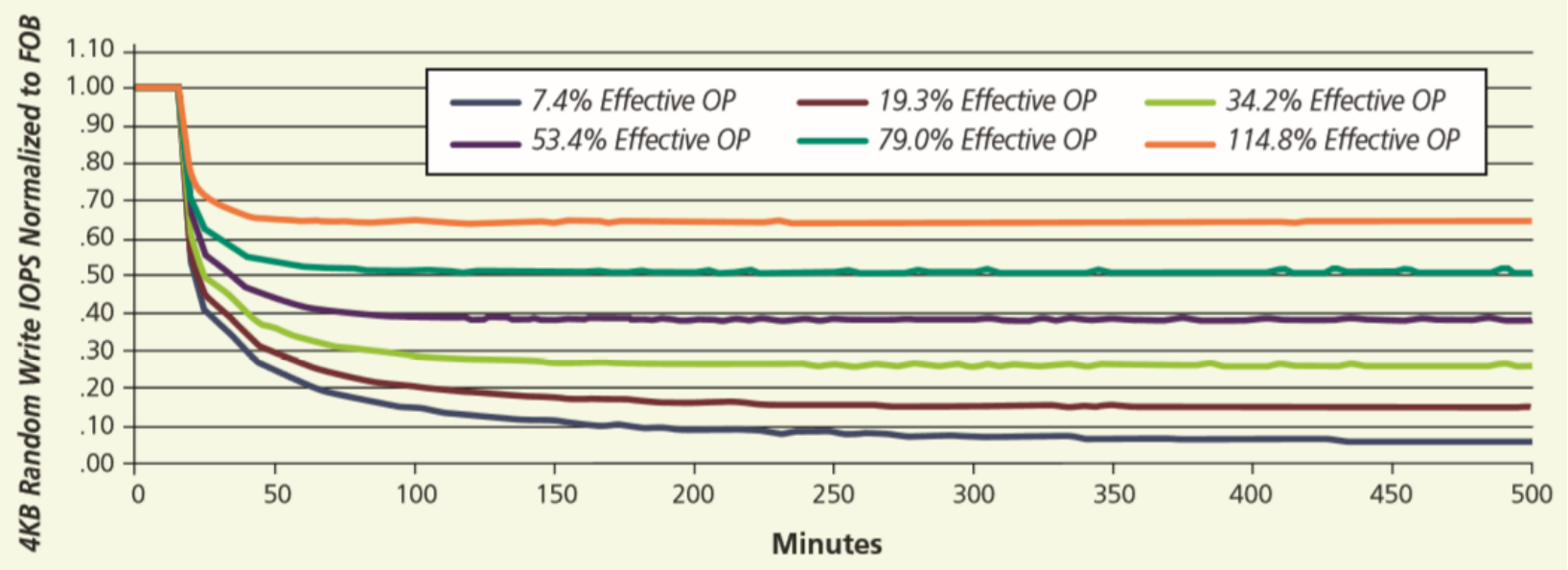


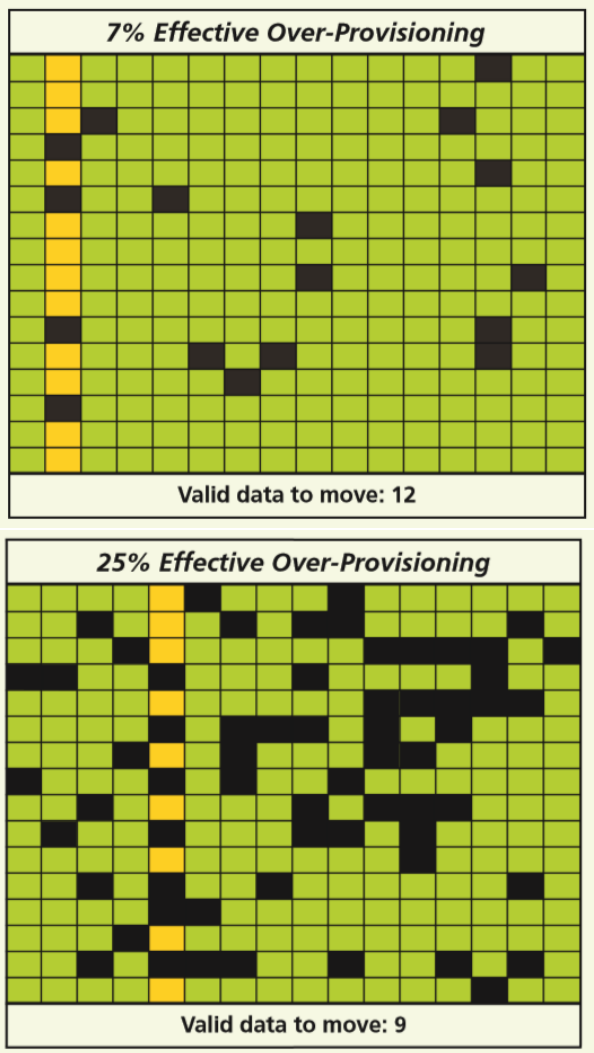
Why Over-Provisioning?

- Over-Provisioning Space (OPS) is used for
  - Write buffers
  - _____
  - _____
  - _____
- Garbage collection cost
  - Affected by utilization of SSD space and Over-Provisioning
  - _____ utilization → Better performance
  - _____ OP → Better performance

Over-Provisioning and GC

- IOPS for random write workloads
  - What about for sequential write workloads?





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

Block Mapping FTL

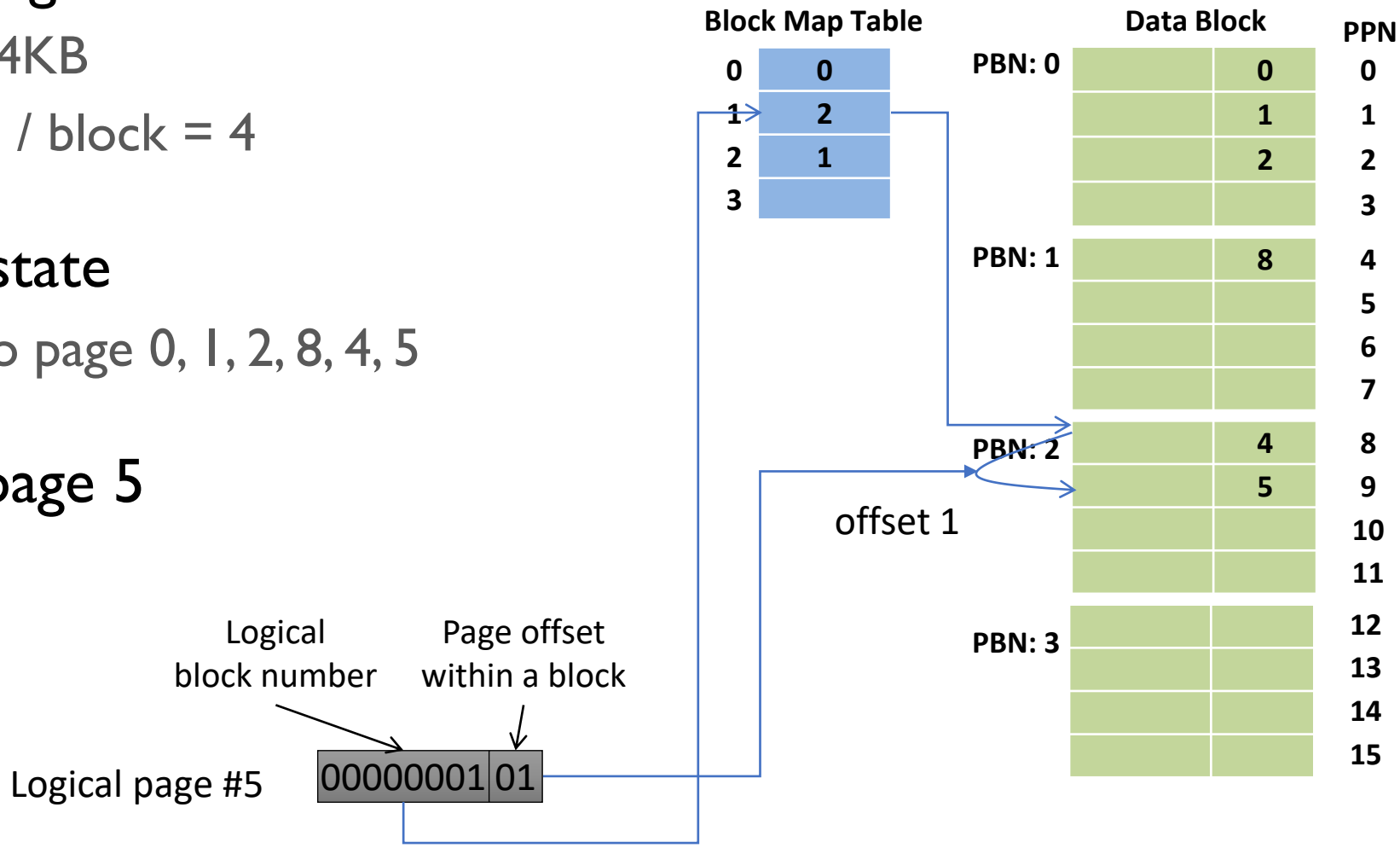
Block Mapping

- Mapping in block-level
  - Logical block number → physical block number
  - Block mapping table (BMT) required
  - Page offset remains the same

- Translation
  - Step 1: logical sector number → logical page number (LPN)
  - Step 2: LPN → logical block number (LBN) + page offset
    - LBN = LPN / N, where N: # of pages in a block
    - Page offset = LPN % N
  - Step 3: LBN → physical block number (PBN) via BMT
    - Use the same page offset

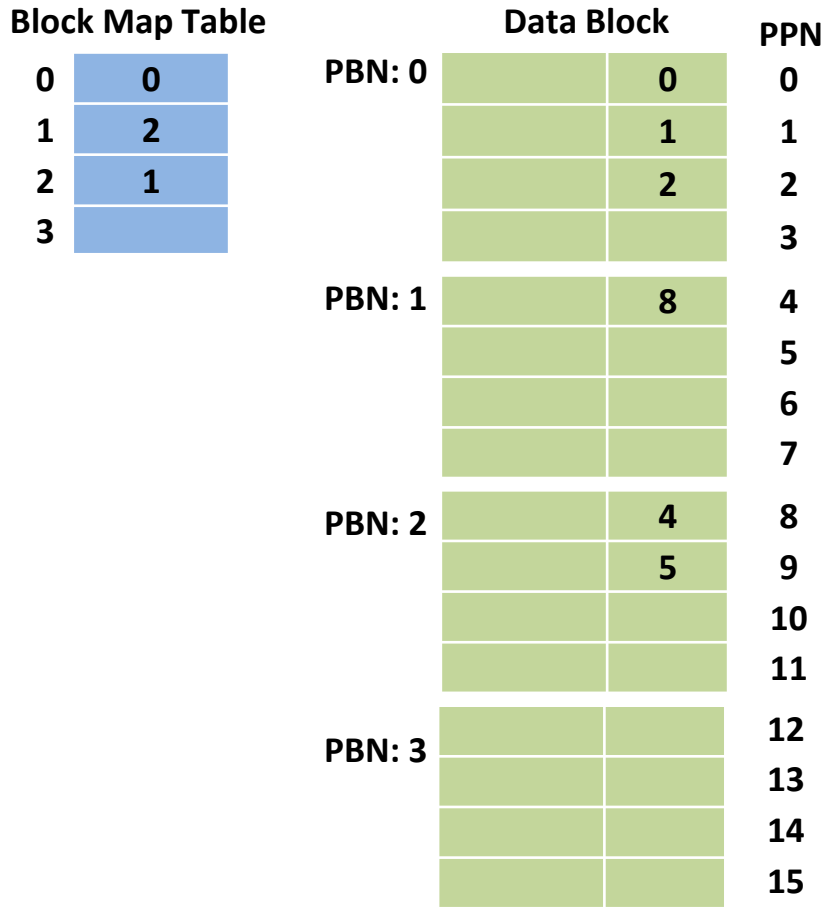
Example: Block Mapping

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



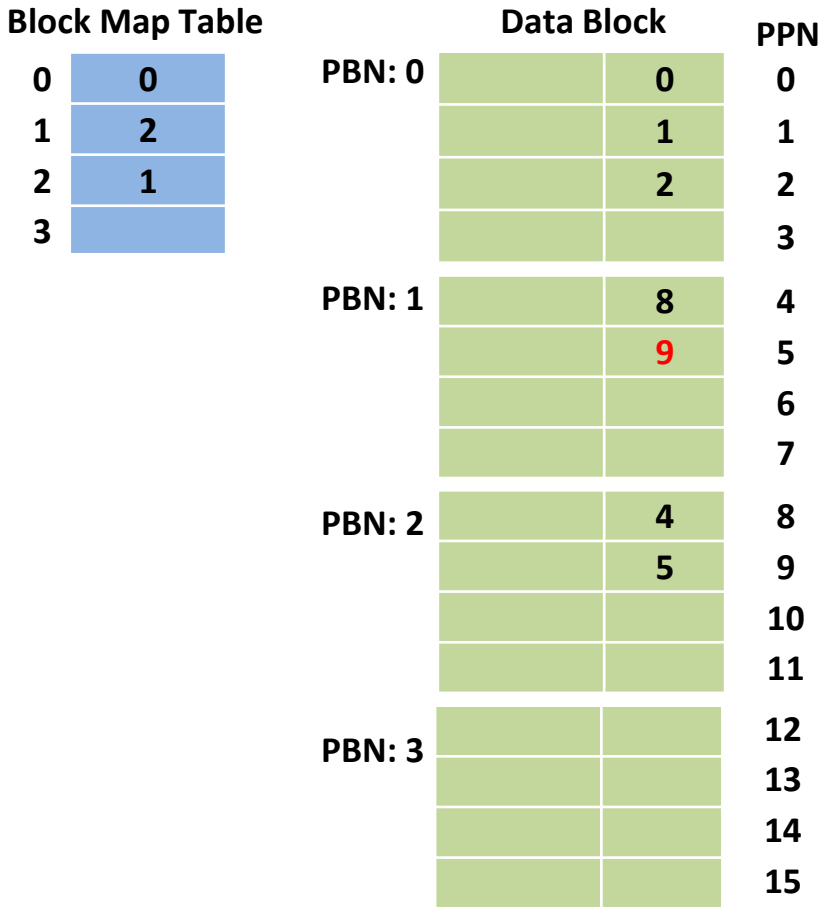
Example: Block Mapping

- 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 (update)



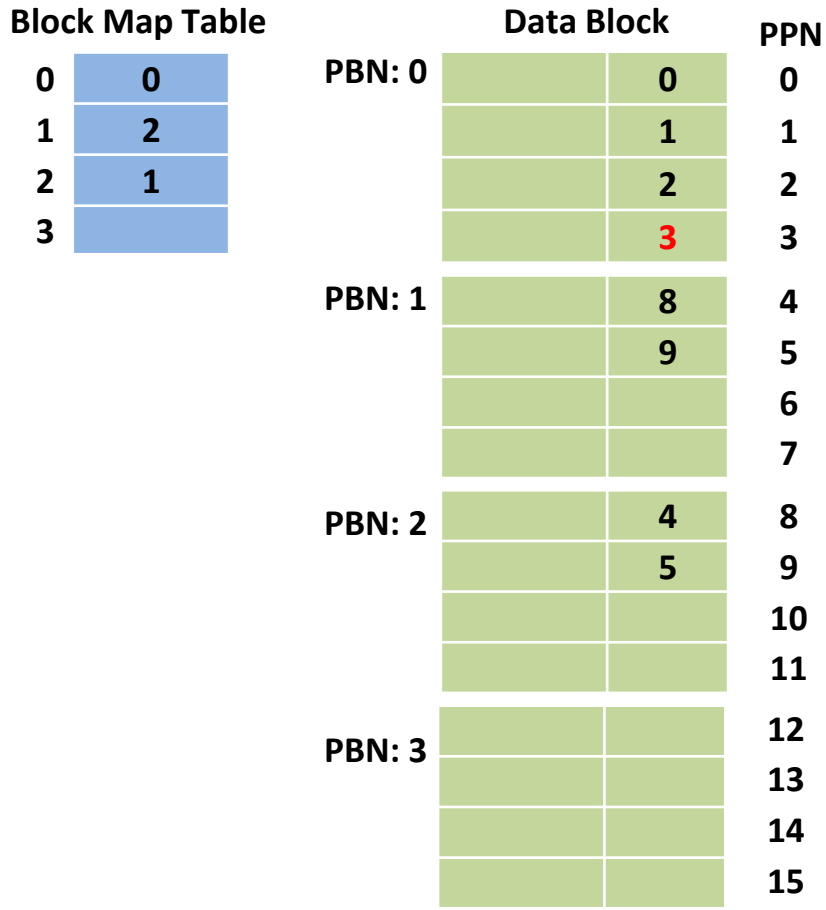
Example: Block Mapping

- 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 (update)



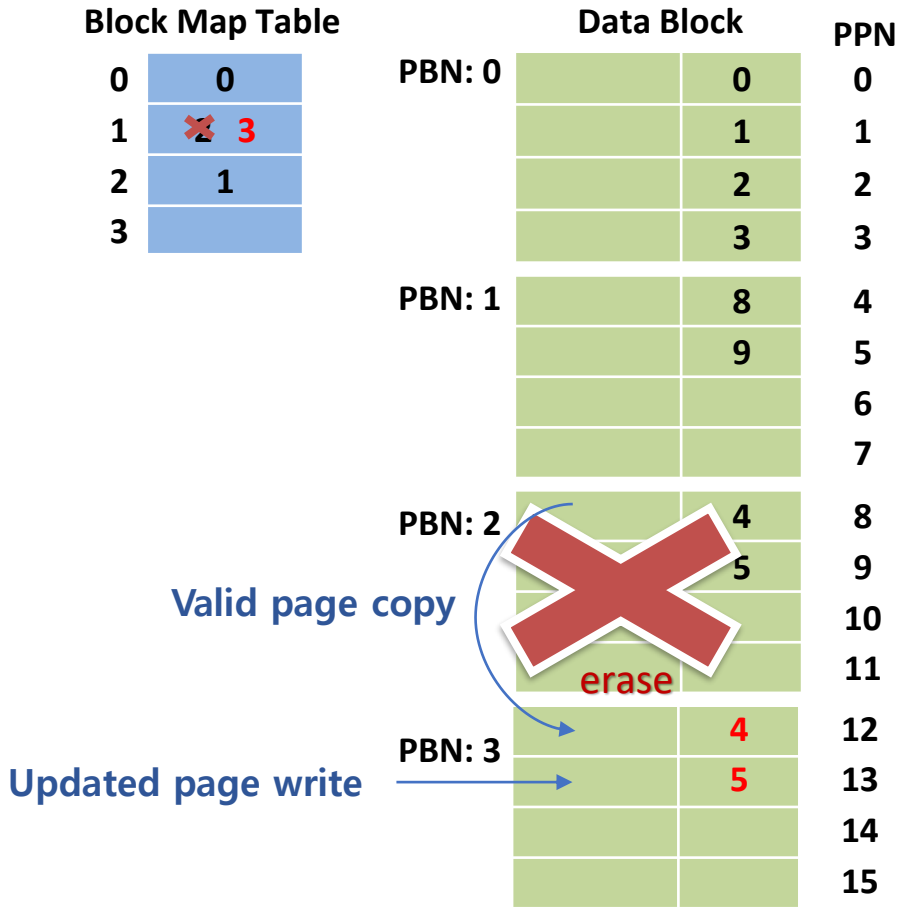
Example: Block Mapping

- 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 (update)



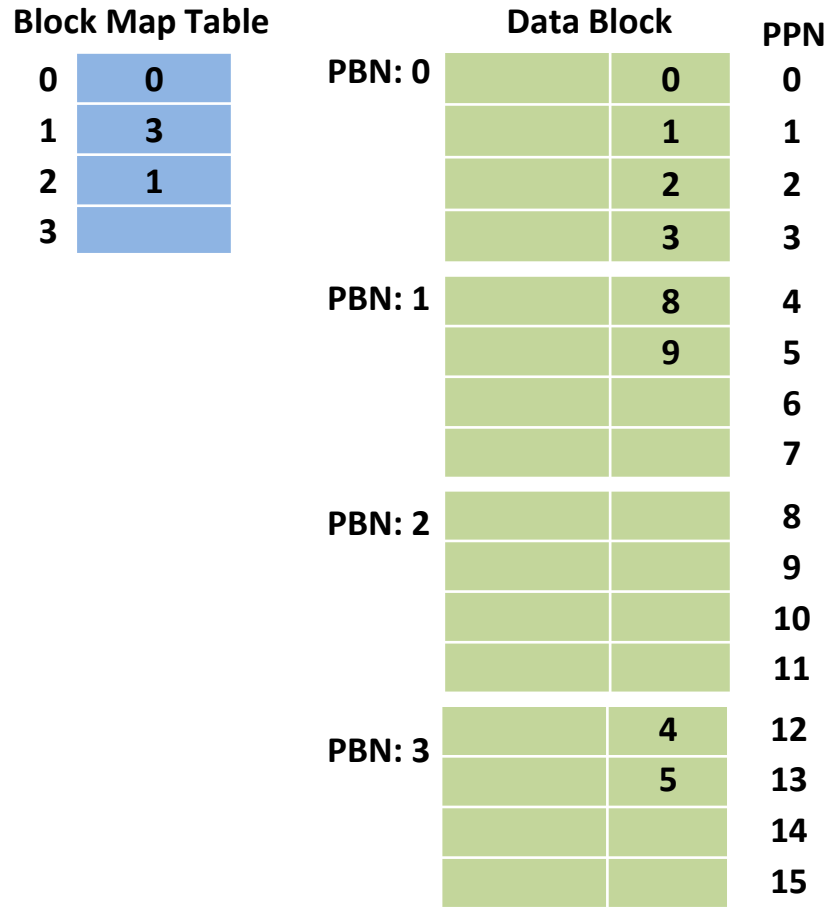
Example: Block Mapping

- 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 (update)



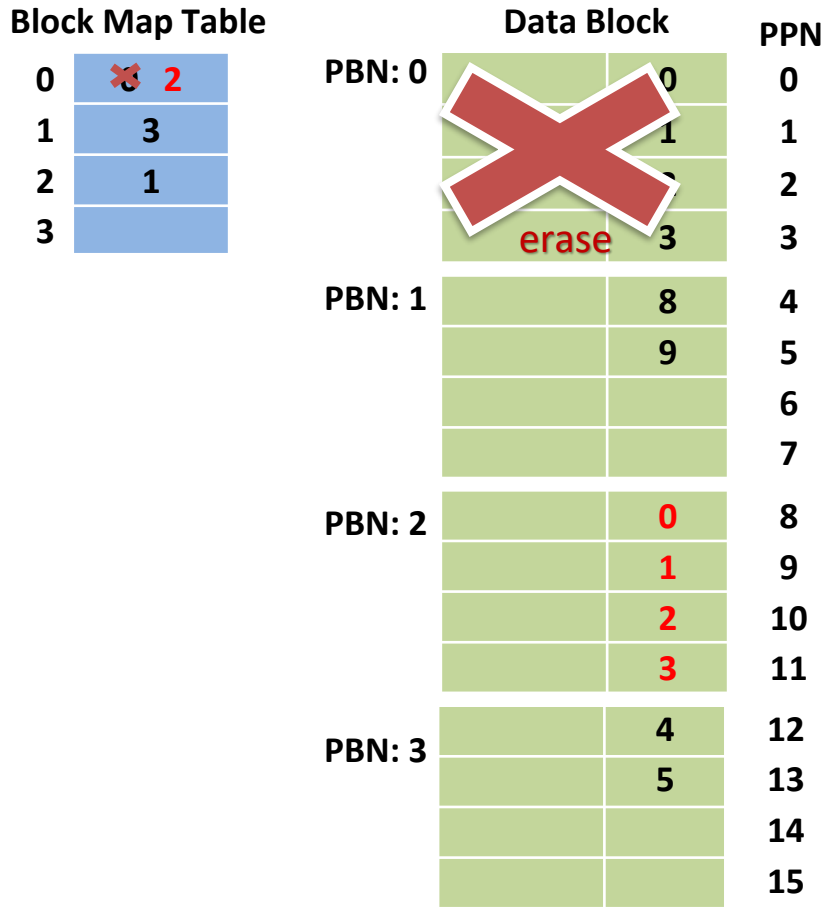
Example: Block Mapping

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



Example: Block Mapping

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



Block Mapping

- Pros
  - Small RAM usage
    - One mapping entry per block
  - Good performance for sequential writes

- Cons
  - Inefficient handling of small random writes
    - Even a single page update requires a block copy & erase

Hybrid Mapping FTL

Hybrid Mapping

- Goals
  - Small DRAM footprint
  - Efficient handling of small random writes

- Basic idea
  - Basic mapping scheme: block mapping
  - A small number of page-mapped log blocks
    - Updated pages are just written to the log block to avoid erase-before-write
    - When there remains no free log block, victim log block is merged with data block to reclaim a log block
    - Log blocks can be regarded as buffers for updated (hot) pages

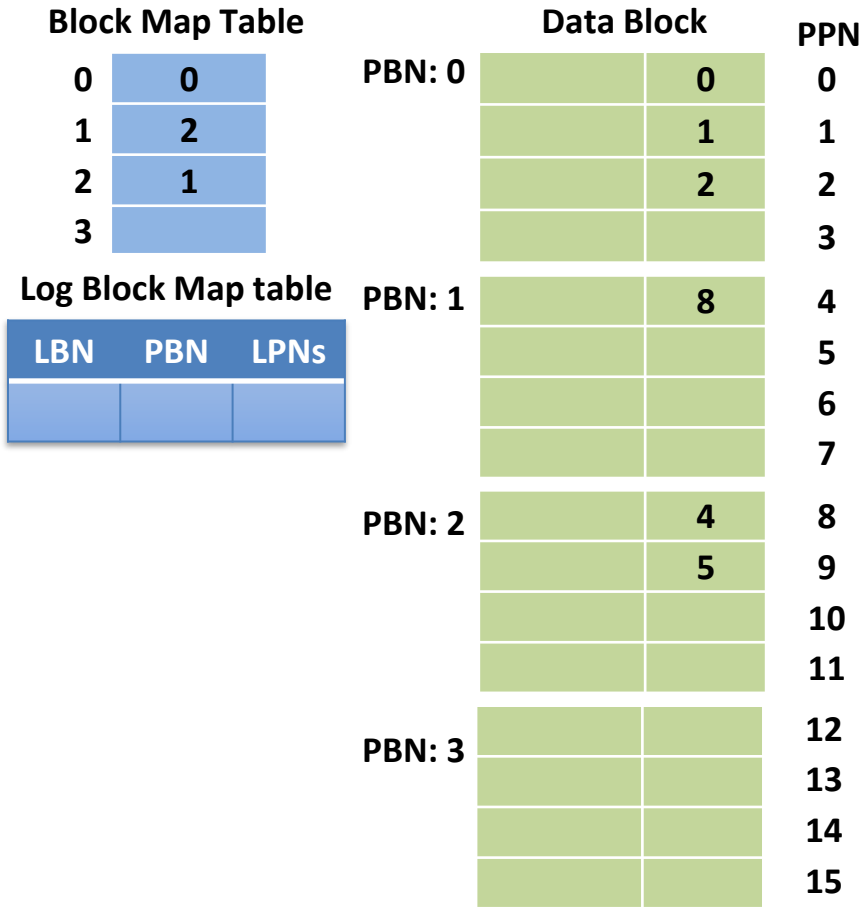
Hybrid Mapping

- Memory requirement
  - Most of blocks are block-mapped (data blocks)
  - Only a small fixed number of blocks (log blocks) are page-mapped
- Random write performance
  - Random writes are accommodated in the page-mapped log blocks

- Many schemes proposed so far
  - BAST, FAST, Superblock, LAST, KAST, SAST, …

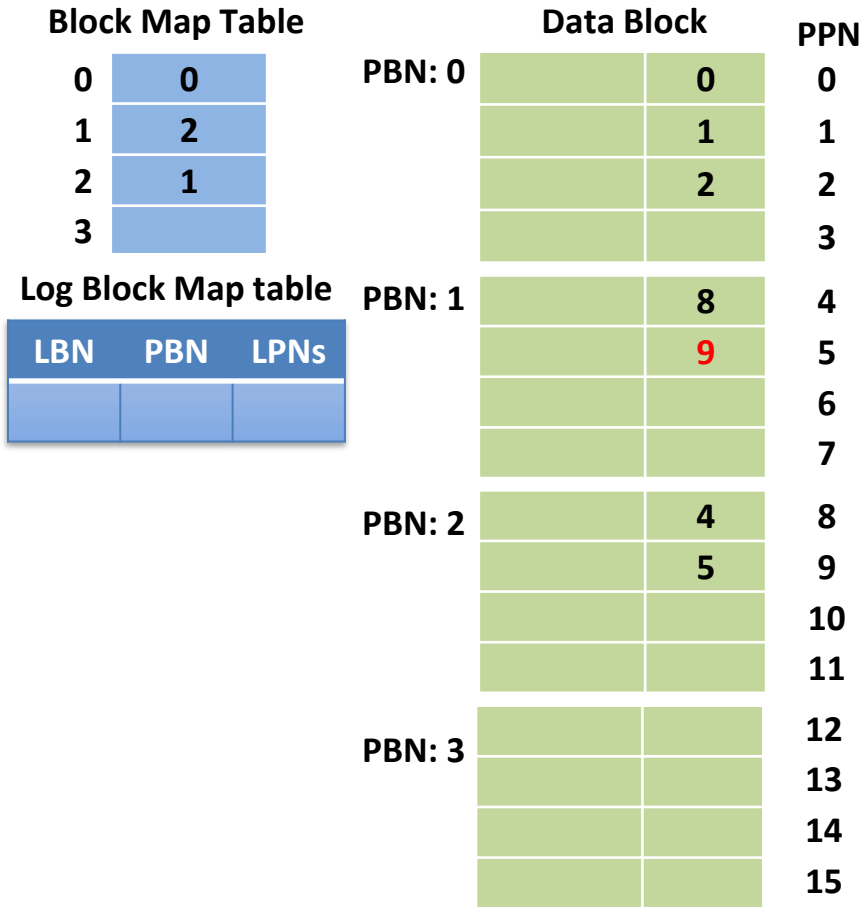
Example: Hybrid Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
  - # of log blocks = 1
- 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 (update)



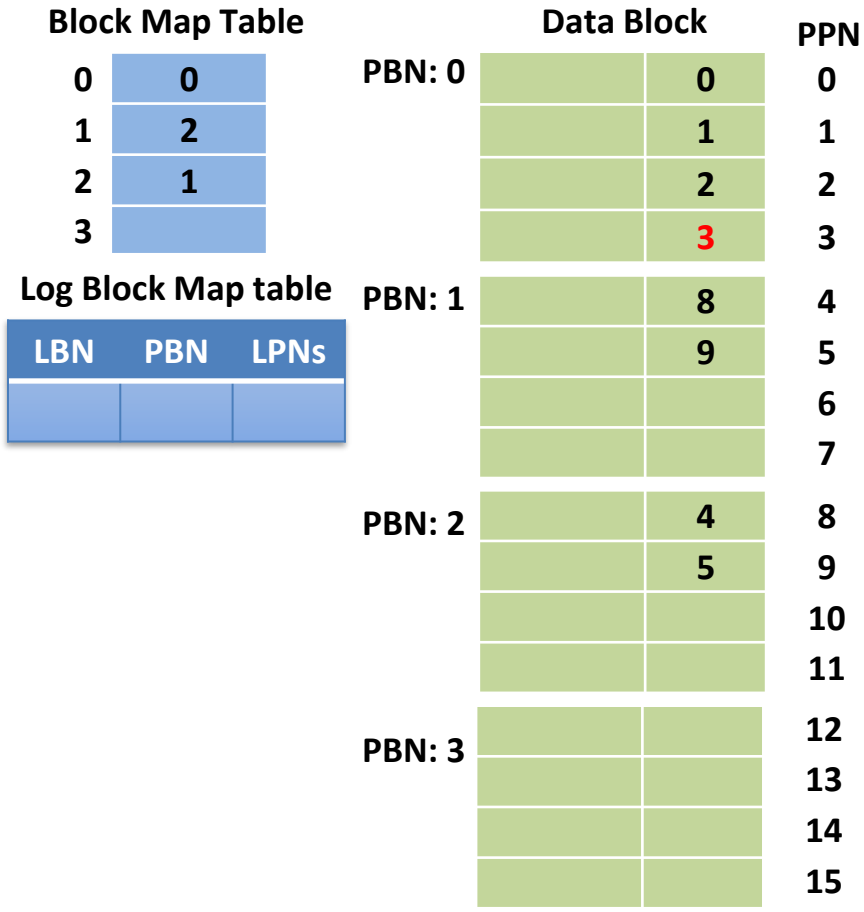
Example: Hybrid Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
  - # of log blocks = 1
- 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 (update)



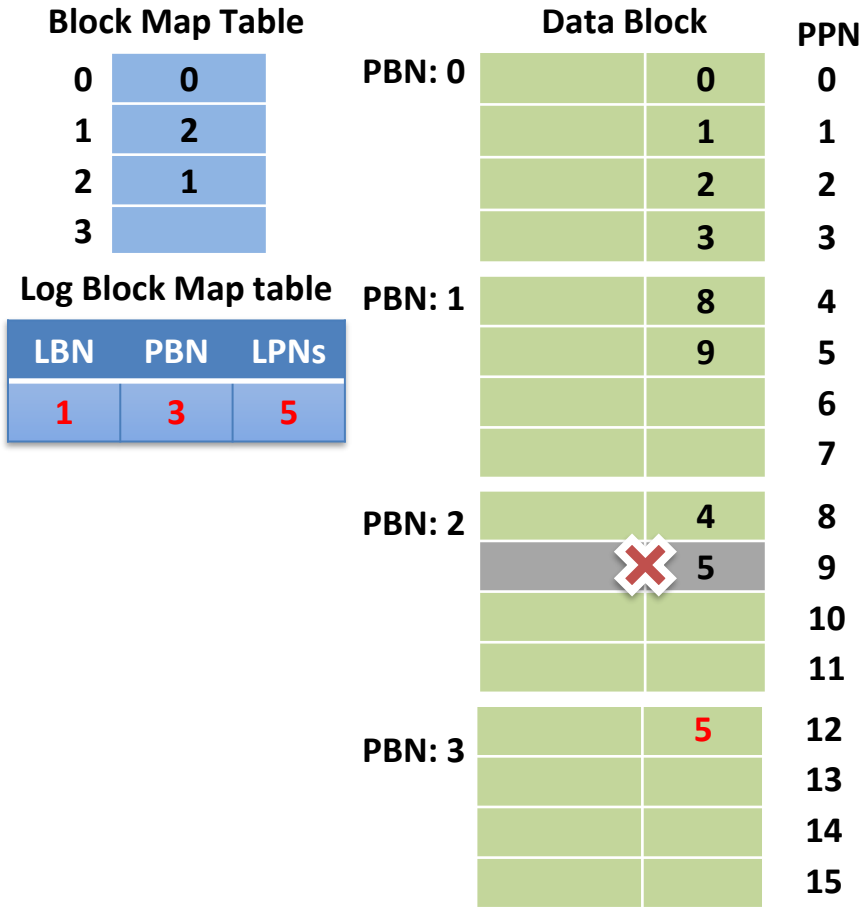
Example: Hybrid Mapping

- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
  - # of log blocks = 1
- 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 (update)



Example: Hybrid Mapping

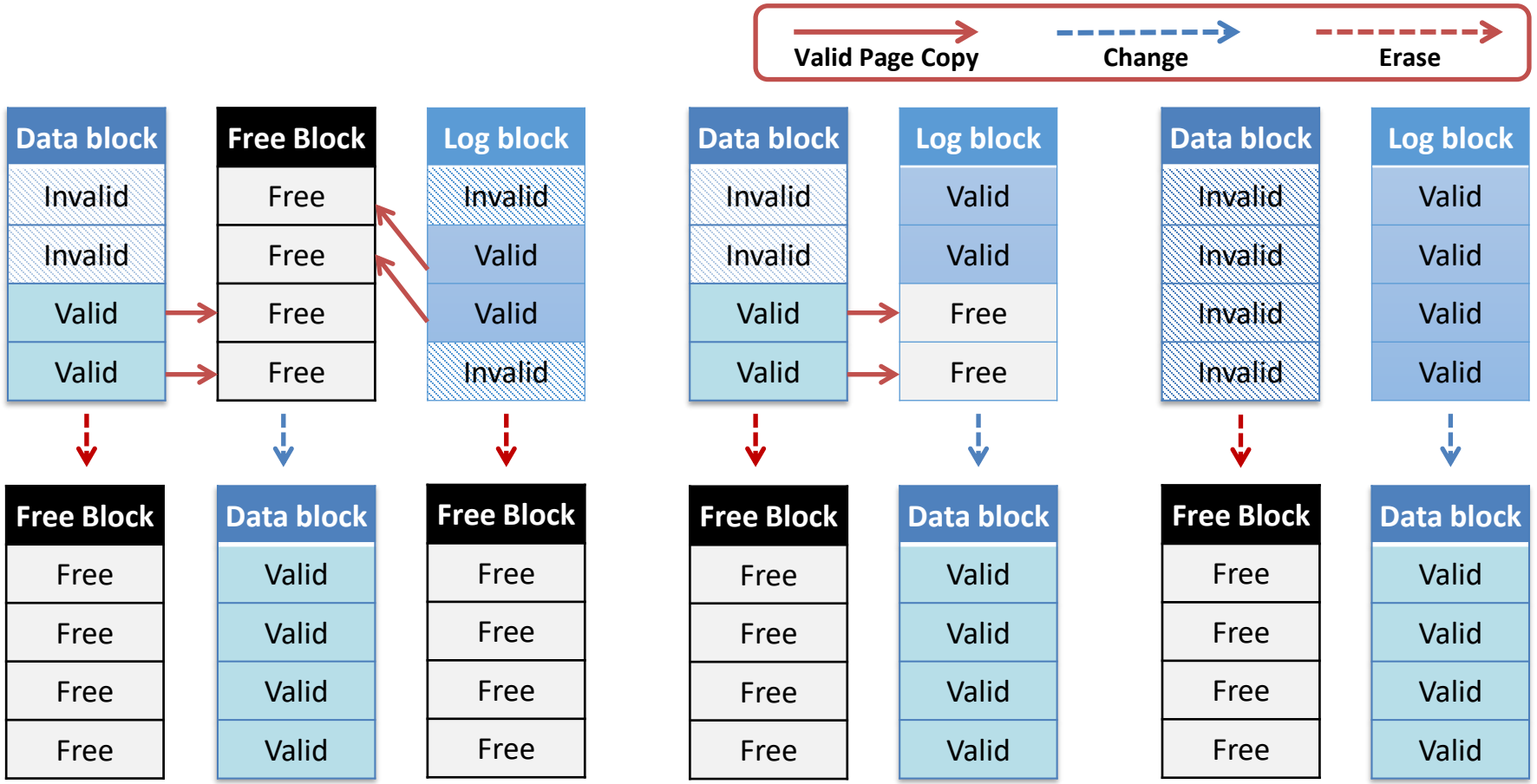
- Flash configuration
  - Page size: 4KB
  - # of pages / block = 4
  - # of log blocks = 1
- Current state
  - Written to page 0, 1, 2, 8, 4, 5
- New requests (in order)
  - Write to page 9
  - Write to page 3
  - Write to page 5 (update)



Design Considerations

- Relationship between data block and log block
  - 1 log block : 1 data block (≈ direct-mapped)
    - Only one log block is dedicated to a single data block
    - One log block is used to accommodate updates from the corresponding data block
  - 1 log block : N data blocks (≈ set-associative)
    - One log block is shared among a set of data blocks
  - M log blocks : N data blocks (≈ fully-associative)
    - All data blocks share all log blocks
    - One log block can accommodate updates from different data blocks
    - Updates from one data block can scatter over different log blocks

Types of Block Merges



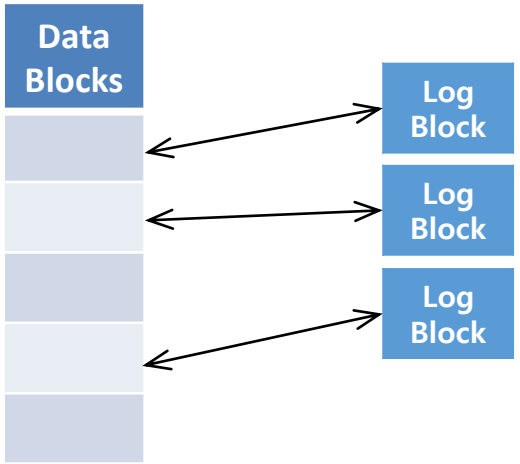
Full Merge
(two block erases,
Max. *Np* page copies)

Partial Merge
(one block erase,
Max. *Np* - 1 page copies)

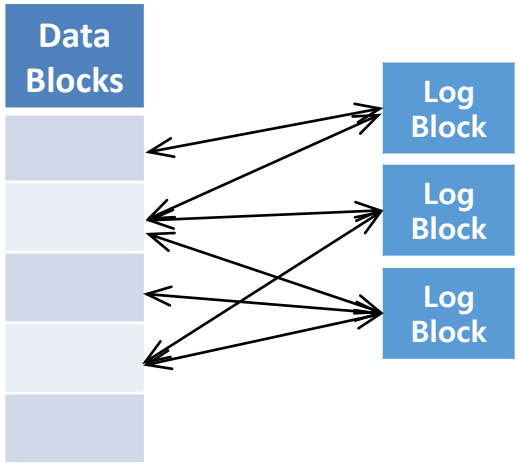
Switch Merge
(one block erase,
No page copy)

FAST

- Fully-Associative Sector Translation
  - Overcome the problems of BAST
    - Alleviate log block thrashing
    - Avoid frequent merge operations
  - N-to-M mapping between data blocks and log blocks
    - Fully associative approach in mapping logical pages onto log blocks



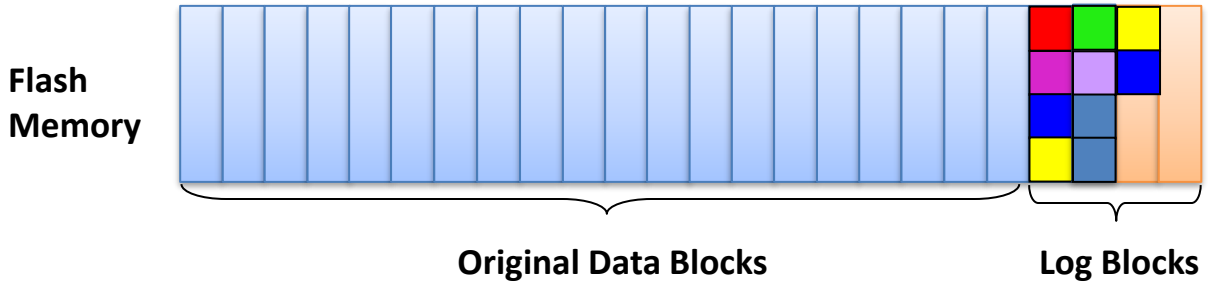
BAST (1:1 mapping)



FAST (N:M mapping)

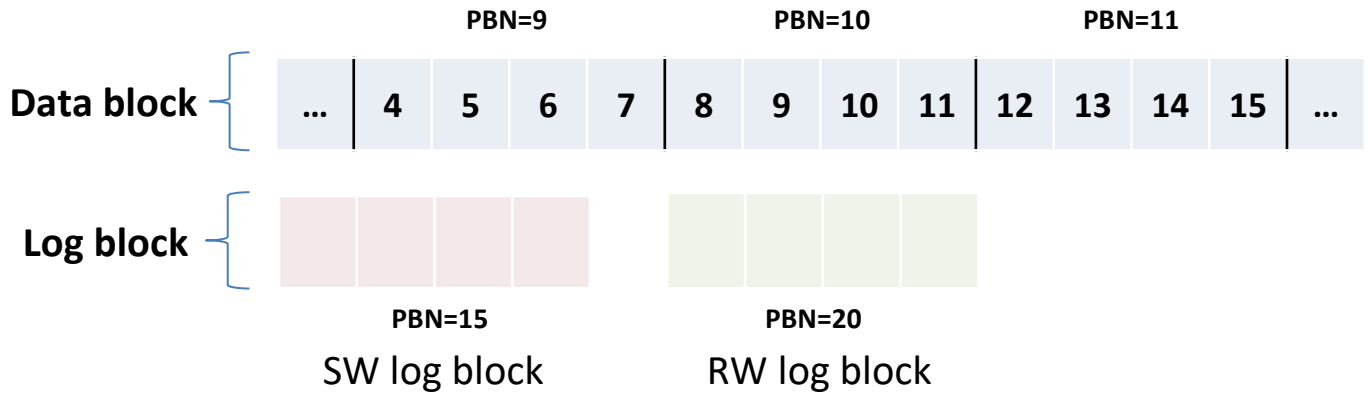
FAST Architecture

- Two types of log blocks
  - SW: Sequential Write log block (just one)
    - To increase the chance of switch merge for sequential writes
  - RW: Random Write log block
- Mapping table
  - Block-level mapping table (for data blocks)
  - Page-level mapping table (for SW)
  - Page-level mapping table (for RW)



Example: FAST

Write(4,…)
Write(5,…)
Write(6,…)
Write(11,…)
Write(7,…)
Write(9,…)
Write(14,…)
…



Block-level mapping table

LBN	PBN
0	
1	9
2	10
3	11

Page-level mapping table for RW

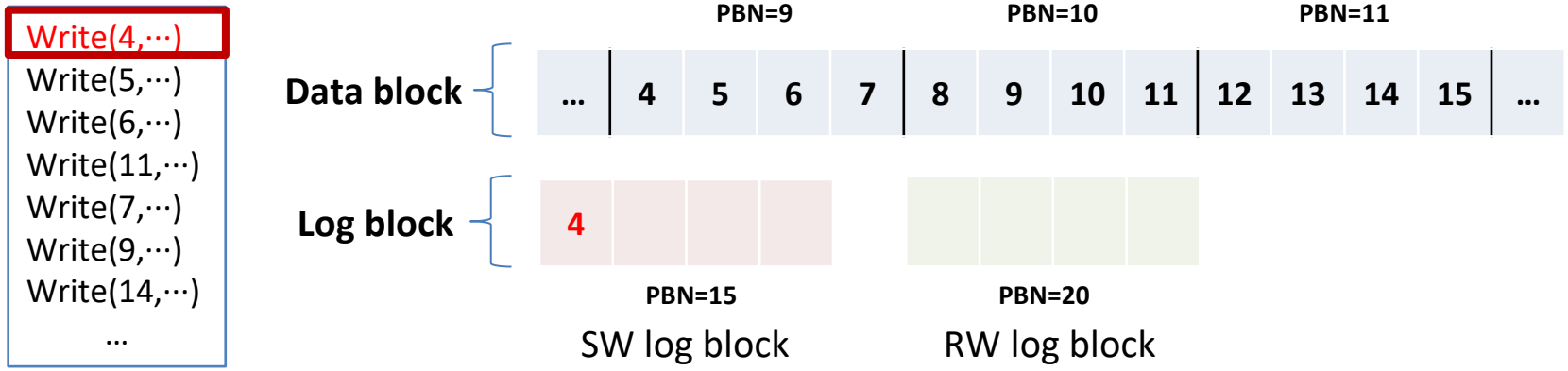
LBN	PBN	LPNs	Offset of RW log block

Page-level mapping table for SW

LBN	PBN	LPNs

Example: FAST

New sequential write



Block-level mapping table

LBN	PBN
0	
1	9
2	10
3	11

Page-level mapping table for RW

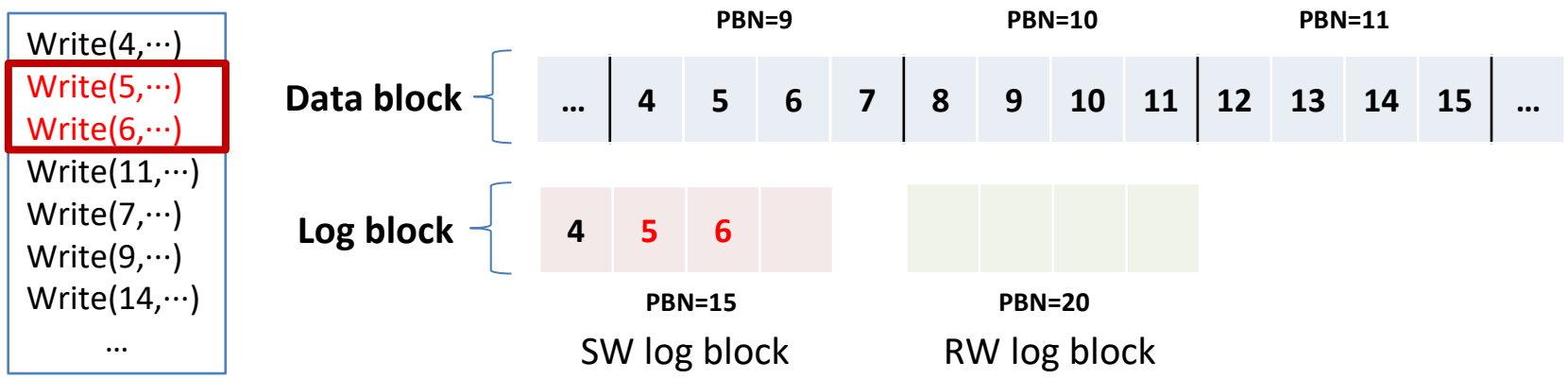
LBN	PBN	LPNs	Offset of RW log block

Page-level mapping table for SW

LBN	PBN	LPNs
1	15	4

Example: FAST

Subsequent sequential writes



Block-level mapping table

LBN	PBN
0	
1	9
2	10
3	11

Page-level mapping table for RW

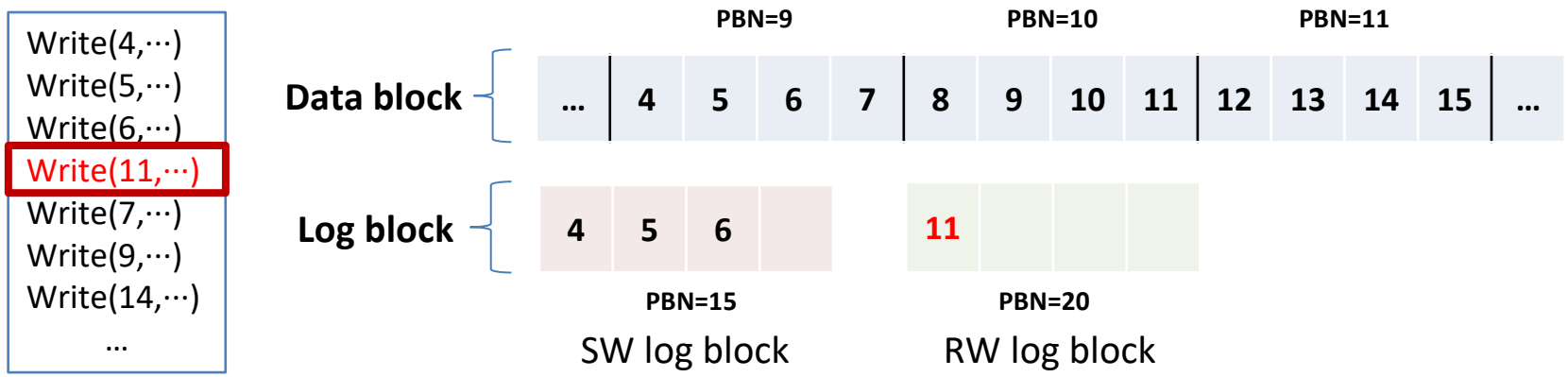
LBN	PBN	LPNs	Offset of RW log block

Page-level mapping table for SW

LBN	PBN	LPNs
1	15	4, 5, 6

Example: FAST

Random write



Block-level mapping table

LBN	PBN
0	
1	9
2	10
3	11

Page-level mapping table for RW

LBN	PBN	LPNs	Offset of RW log block
2	20	11	0

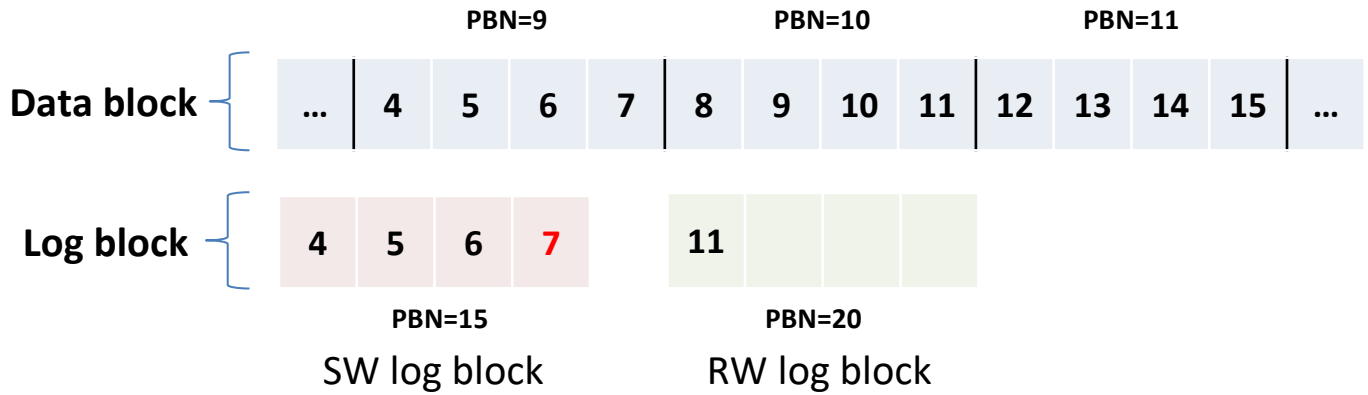
Page-level mapping table for SW

LBN	PBN	LPNs
1	15	4, 5, 6

Example: FAST

Subsequent sequential write

Write(4,…)
Write(5,…)
Write(6,…)
Write(11,…)
Write(7,…)
Write(9,…)
Write(14,…)
…



Block-level mapping table

LBN	PBN
0	
1	9
2	10
3	11

Page-level mapping table for RW

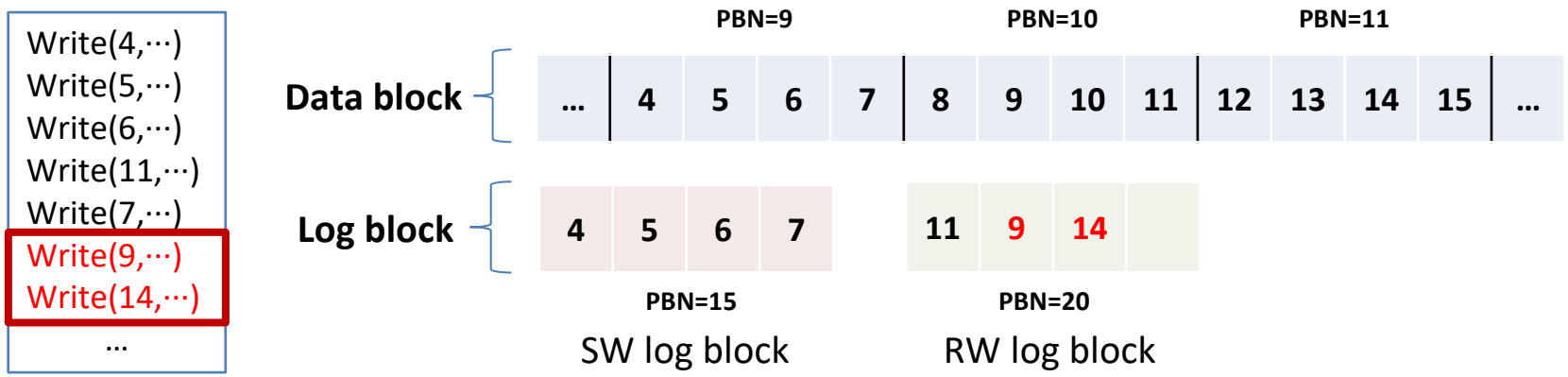
LBN	PBN	LPNs	Offset of RW log block
2	20	11	0

Page-level mapping table for SW

LBN	PBN	LPNs
1	15	4, 5, 6, 7

Example: FAST

Another random writes



Block-level mapping table

LBN	PBN
0	
1	9
2	10
3	11

Page-level mapping table for RW

LBN	PBN	LPNs	Offset of RW log block
2	20	11, 9	0, 1
3	20	14	2

Page-level mapping table for SW

LBN	PBN	LPNs
1	15	4, 5, 6, 7

Summary: FAST

- Pros
  - Improved log block utilization
  - Avoid unnecessary merge

- Cons
  - Increased merge time
  - Detecting sequential writes is hard
  - Looking up the page-level mapping table for RW log blocks