FRASH: Exploiting Storage Class Memory in Hierarchical File System

Youjip Won

Dept. of Electronics & Computer Engineering
Hanyang University
Seoul, Korea

Outlines

- Motivation
- Byte Addressable NVRAM and Flash
- Maintaining Flash as Storage Device: Log-Structured vs. FTL
- FRASH: Hybrid File System for Hierarchical Storage
- Performance Analysis
- Conclusion & Future Work

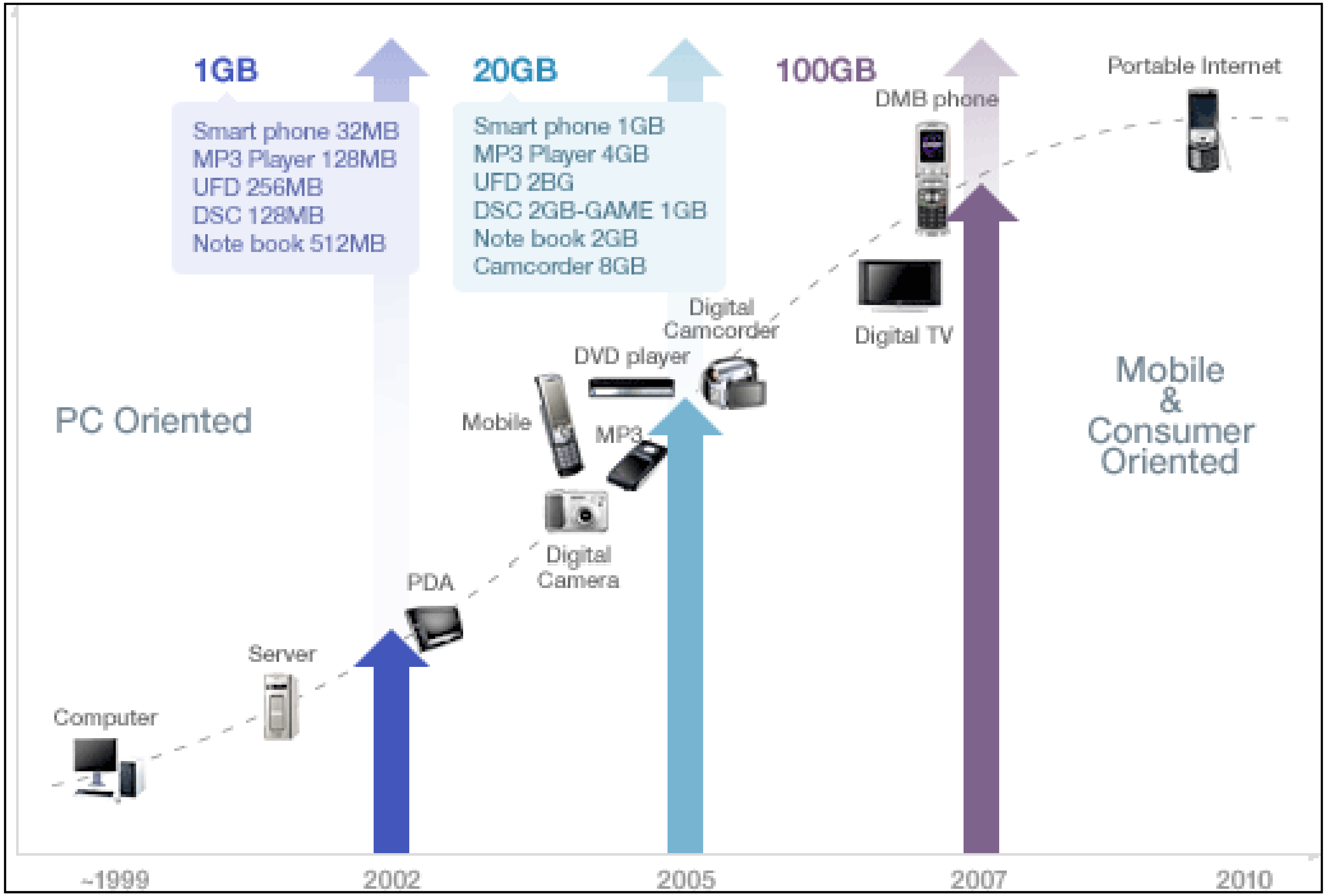
Background

- Advancement of Large Scale NAND Flash
- Advancement of Byte Addressable NVRAM

Does the current file system technology

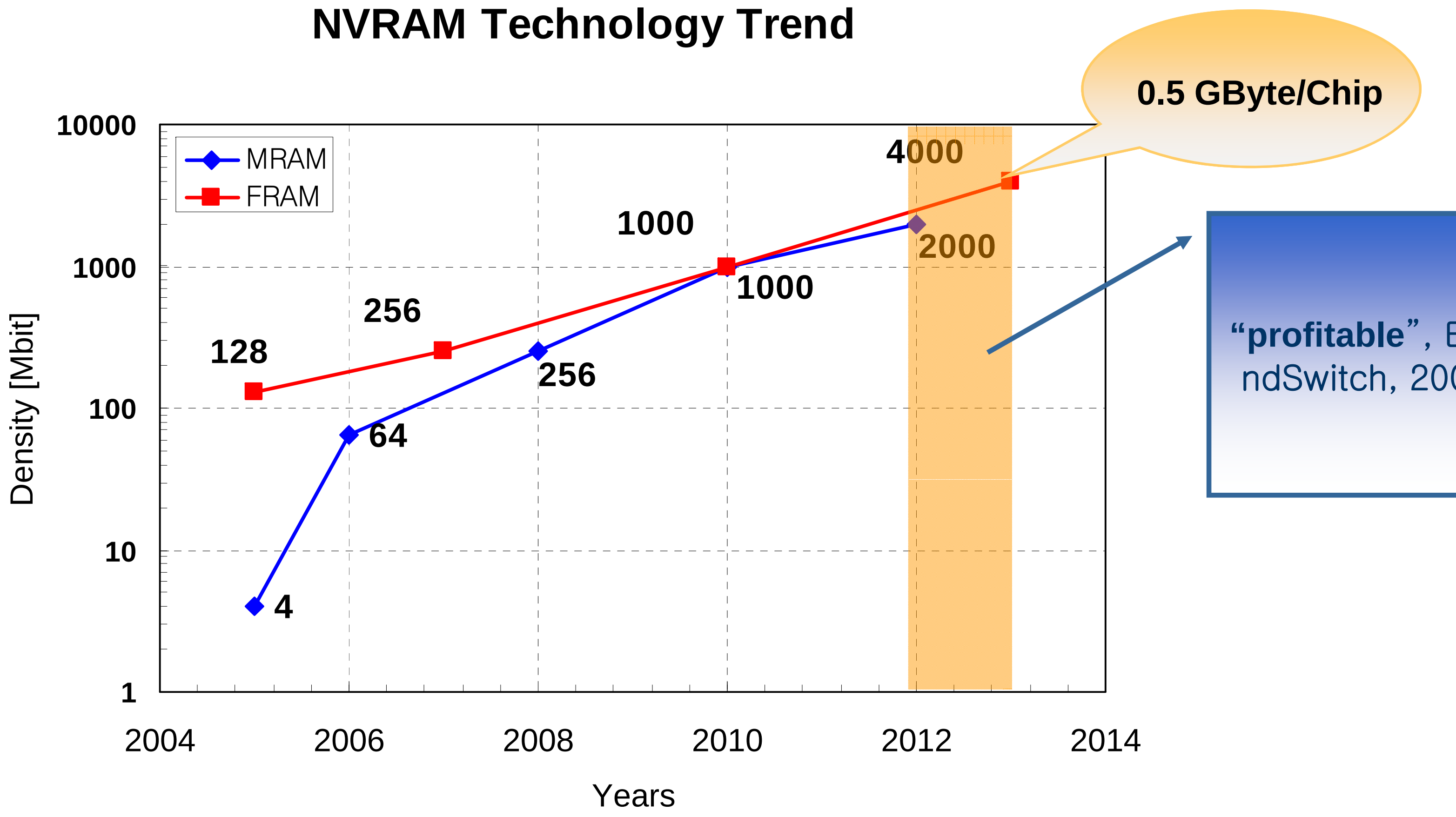
effectively exploit their physical characteristics?

NAND Flash Trend



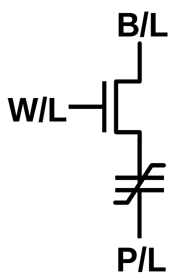
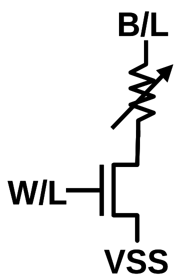
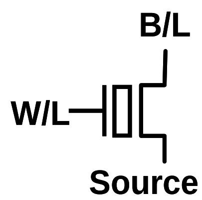
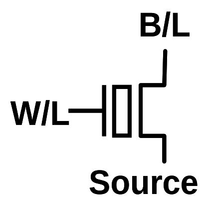
Byte-addressable NVRAM Technology Trend

Source: FRAM: Nikkei Elec., MRAM: NEDO(Japan)



"**profitable**", ByteandSwitch, 2006/6

NVRAMs

Items	FRAM	PRAM	NOR	NAND
Byte Addressable	Yes	Yes	Yes (read only)	No
Non-volatility	Yes	Yes	Yes	Yes
Read	85ns	62ns	85ns	16us
Write/Erase	85ns / none	300ns / none	6.5us / 700ms	200us / 2ms
Power Consumption	Low	High	High	High
Capacity	Low	Middle	Middle	High
Endurance	1E15	>1E7	100K	100K
Unit Cell				

Some fact on Solid State Disk

Device	Sequential	Random 8KB	Price $	Power	iops/$	iops/watt
SCSI 15k rpm	75 MBps	200 iops	500$	15 watt	0.5	13
SATA 10k rpm	60 MBps	100 iops	150$	8 watt	0.7	12
Flash - read	53 MBps	2,800 iops	400$	0.9 watt	7.0	3,100
Flash - write	36 MBps	27 iops	400$	0.9 watt	0.07	30

< source : "Flash Disk Opportunity for Server-Application", Microsoft Research >

excerpt from S. Kang, "NVRAM for Write Buffer in SSD", NVRAMOS 2007, Jeju, Korea

Limitation

- FLASH

  Slow write or mount delay!

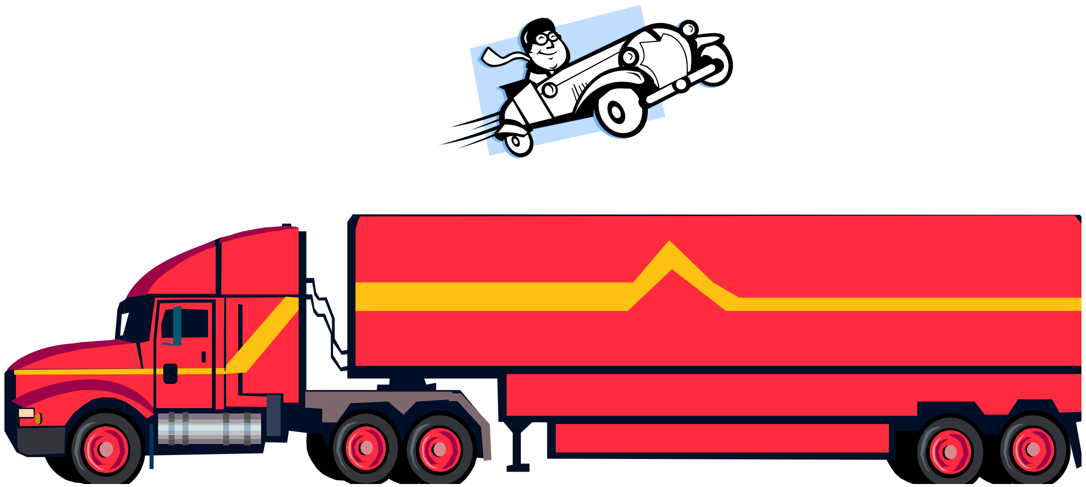
  - Page write/block erase → slow write performance
- Byte-addressable NVRAM

  Small and expensive!

  - FRAM: still needs more density
  - PRAM: write speed is slow.
  - MRAM: power consumption

Objective

- New file system for hybrid storage of
  - Byte addressable NVRAM
  - NAND Flash



Related works

- Booting time acceleration
  - snapshotboot: longer unmount time
  - RFFS: mount time is subject to flash device size
  - MNFS: large block size
  - yaffs2/3
- NVRAM: MRAMFS, HERMES, PRIMS, CONQUEST
- Memory file system: RAMFS, TMPFS

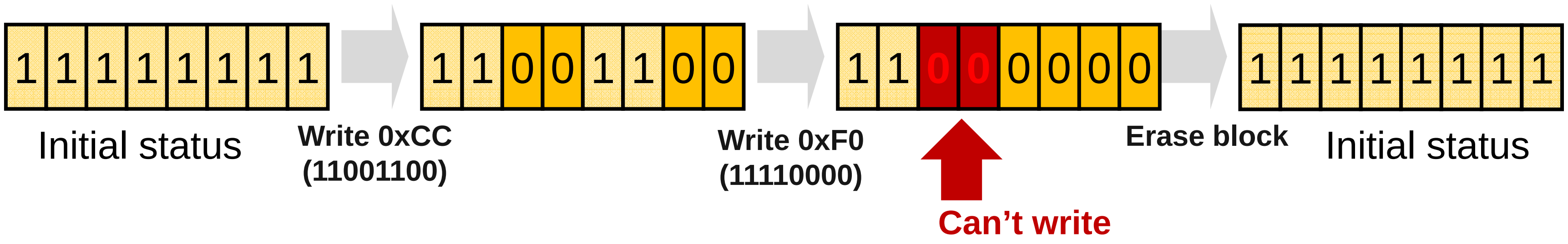
Objective

New File System for byte addressable NVRAM and Flash

- Faster mount time
- Robust against crash
- Faster I/O

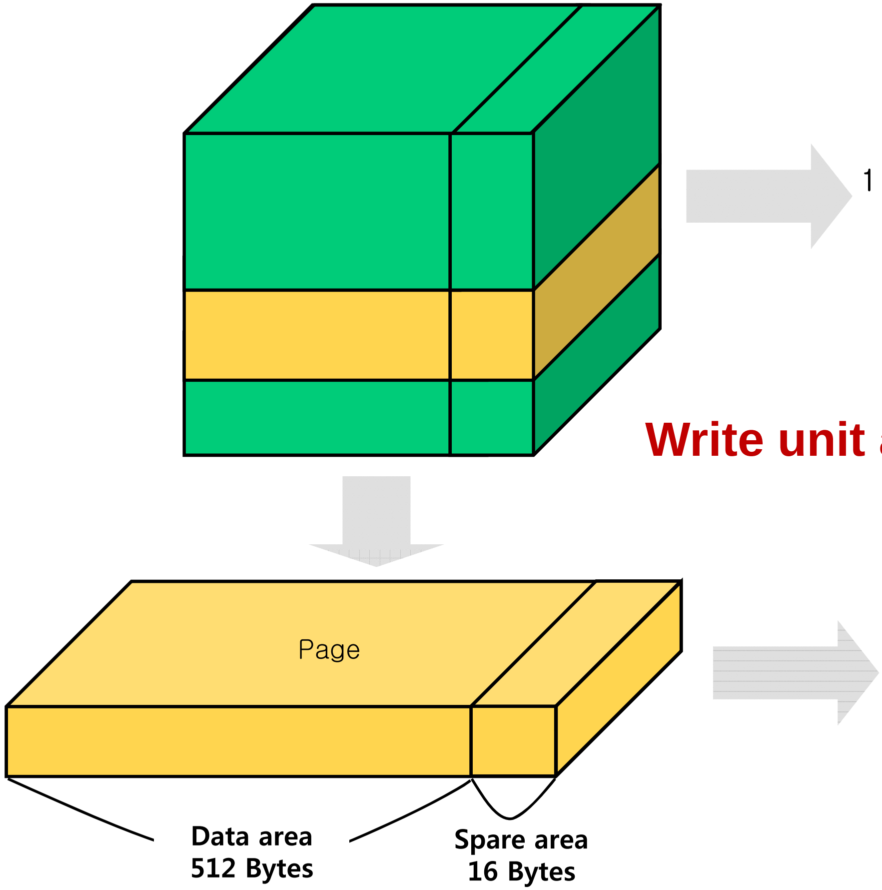
Limitation of Flash Memory: Write/Erase

- Page write/block erase
  - Erase before write on dirty page.
  - Write unit and erase unit are different.



- Wear-out
  - an upper bound on the number of write/erase cycle of a flash memory block.

NAND Flash



1 Block = 528 Bytes X 32 Pages
= 16 K + 512 Bytes
Erase Unit

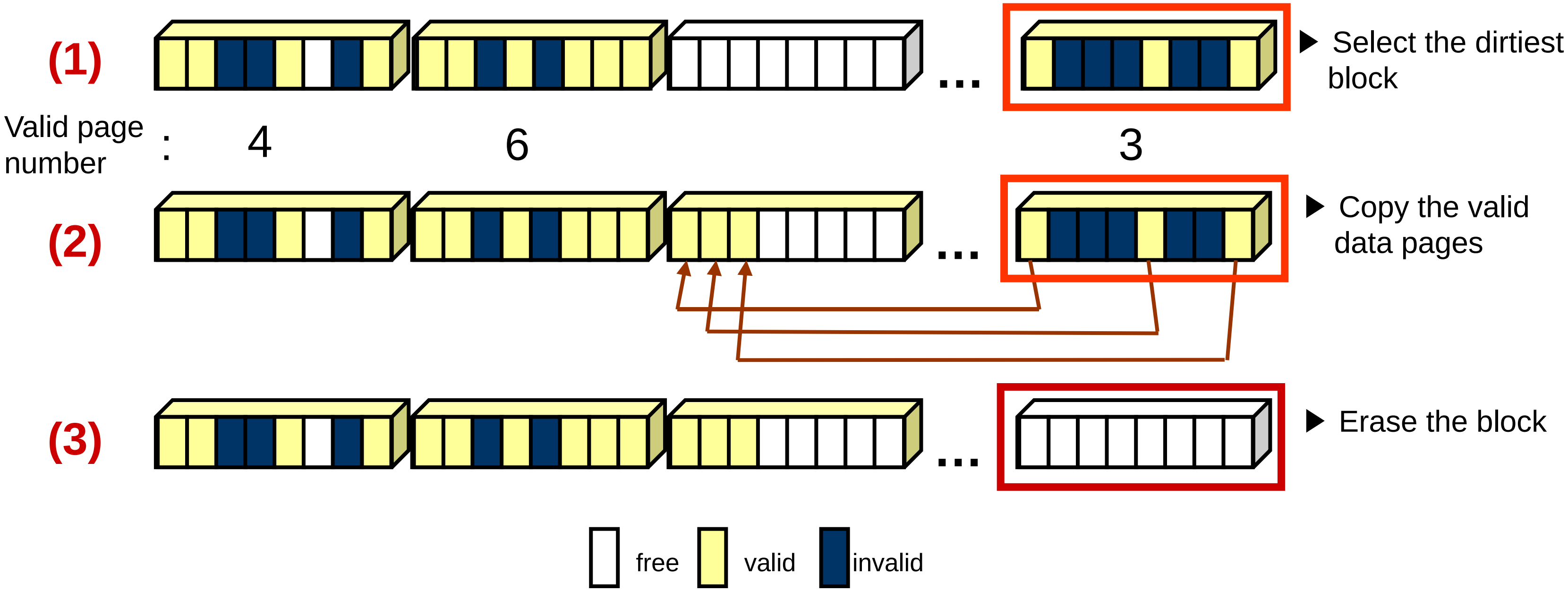
Write unit and erase unit are different!!

1 Page = 512 Bytes + 16 Bytes
= 528 Bytes
Write Unit

Characteristics of Flash Memory

- Out-place update: Erase-before-write
- Basic Operations
  - Read : Page (512byte)
  - Write (program) : Page
  - Erase : Block (1Page × 32)
- Asymmetric cost
  - Read : 20 ㎲
  - Write : 200 ㎲
  - Erase : 2 ㎳

Limitations of Flash Memory: Garbage collection

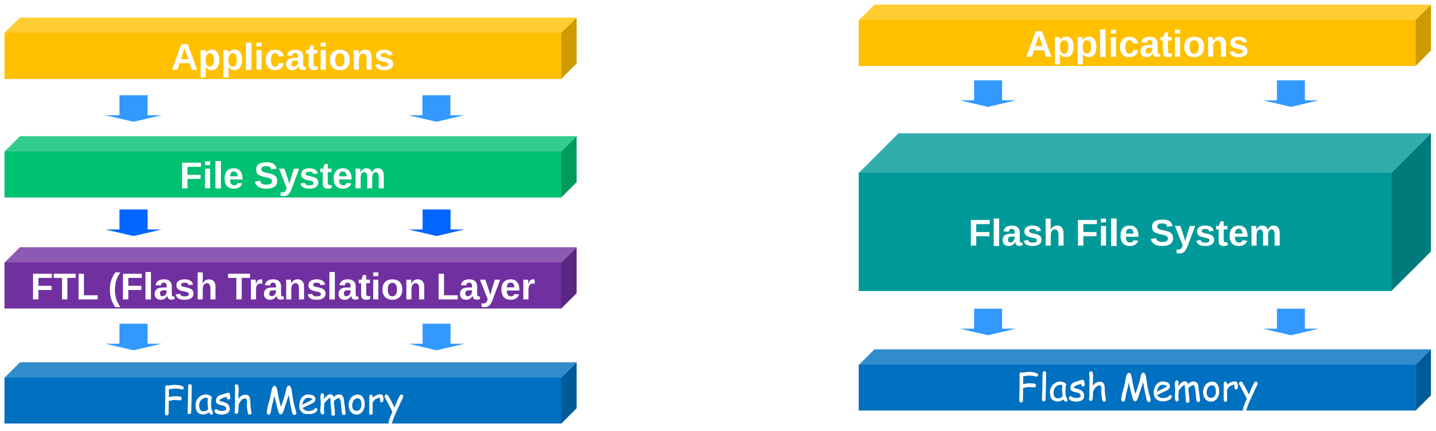


NAND Flash File System

- Hide the erase operation from the upper layer.
- Efficient Garbage collection
- Maintain performance

Two approaches for NAND Flash File System

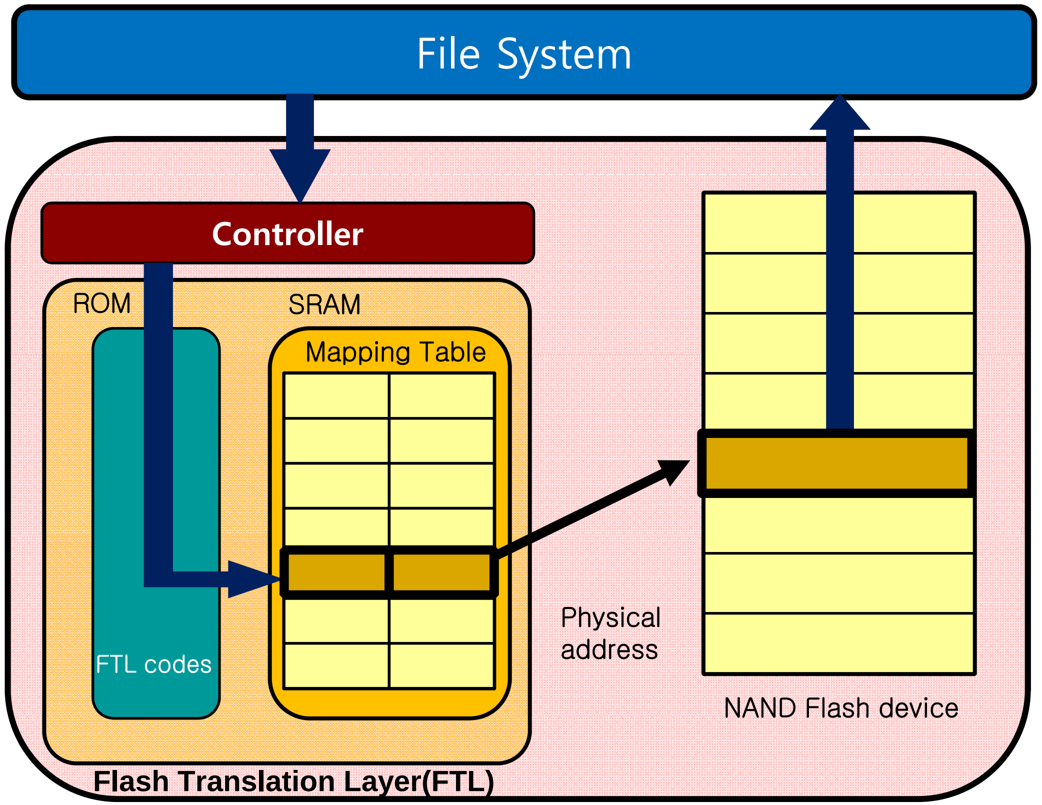
- Flash Translation layer(FTL): Samsung, Intel,...
- Log Structured File System: Android(Google)



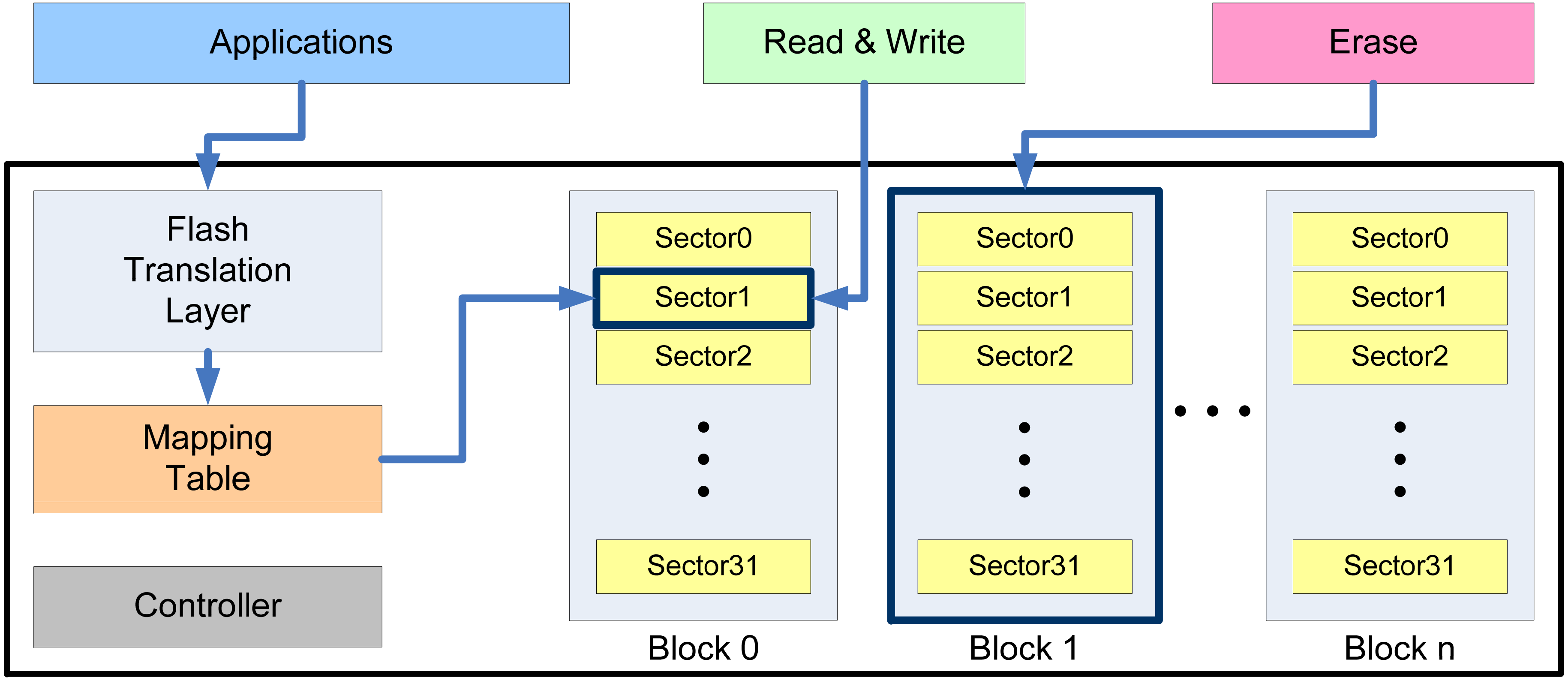
FTL

Flash File System

Flash Translation Layer (FTL)



Flash Translation Layer(FTL)



excerpt from D. Lee, “An Efficient Buffer Management Scheme for Implementing a B-Tree on NAND Flash Memory”, NVRAMOS 2007, Jeju, Korea

Flash Translation Layer (FTL)

- Strength
  - Can use conventional file systems

- Weakness
  - Encumbered by patents
  - Software FTL on Linux has bad performance.
  - Hardware FTL has power consumption problem.

Log Structured File System

- Out-place update
  - All pending writes are buffered in memory into a single segment
  - Flush the segment into the disk as a log in order to use disk full bandwidth

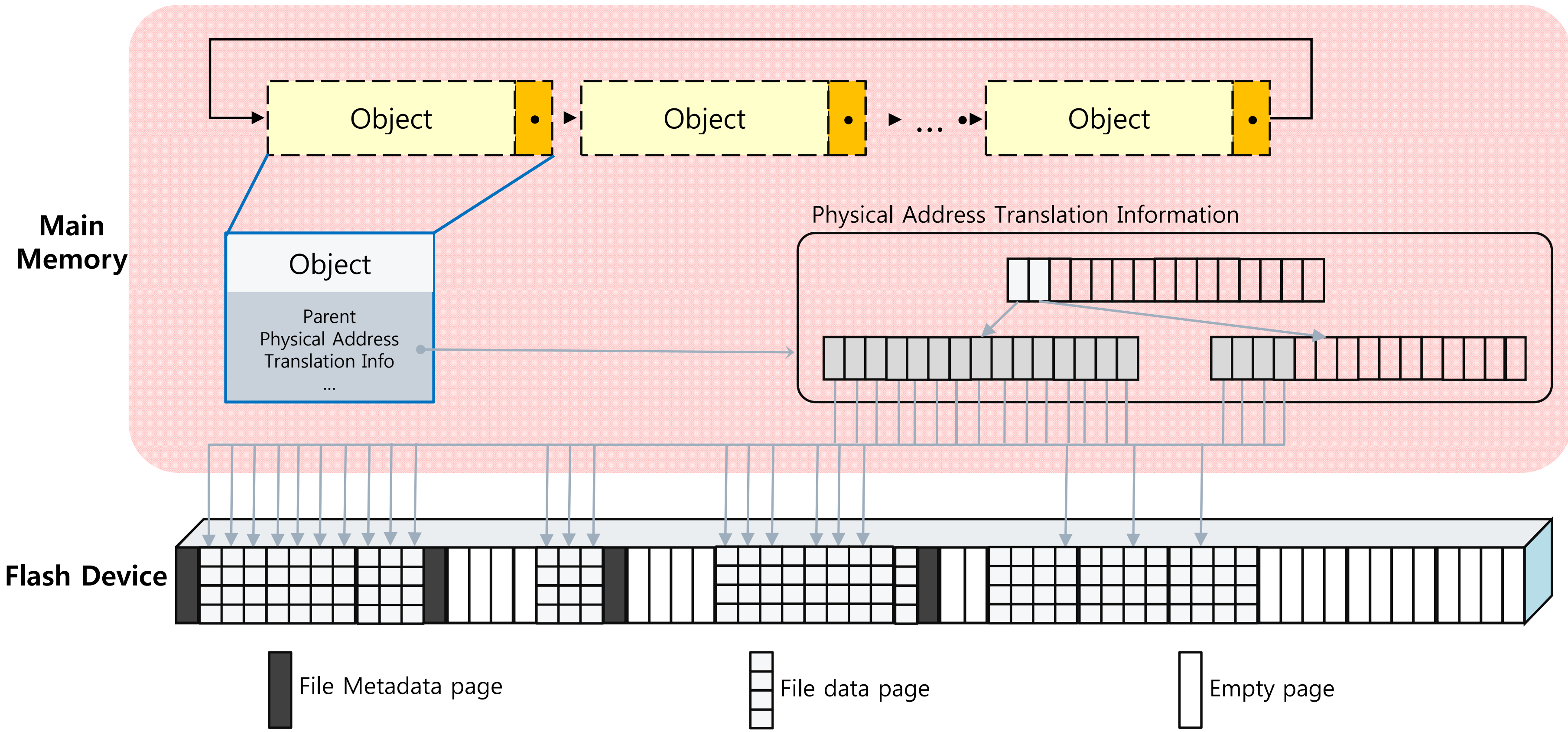
Long mount delay

- Need a map to find file metadata
  - Because all file metadata are scattered all over the log
- Need a space management mechanism
  - Segment cleaner (garbage collector)

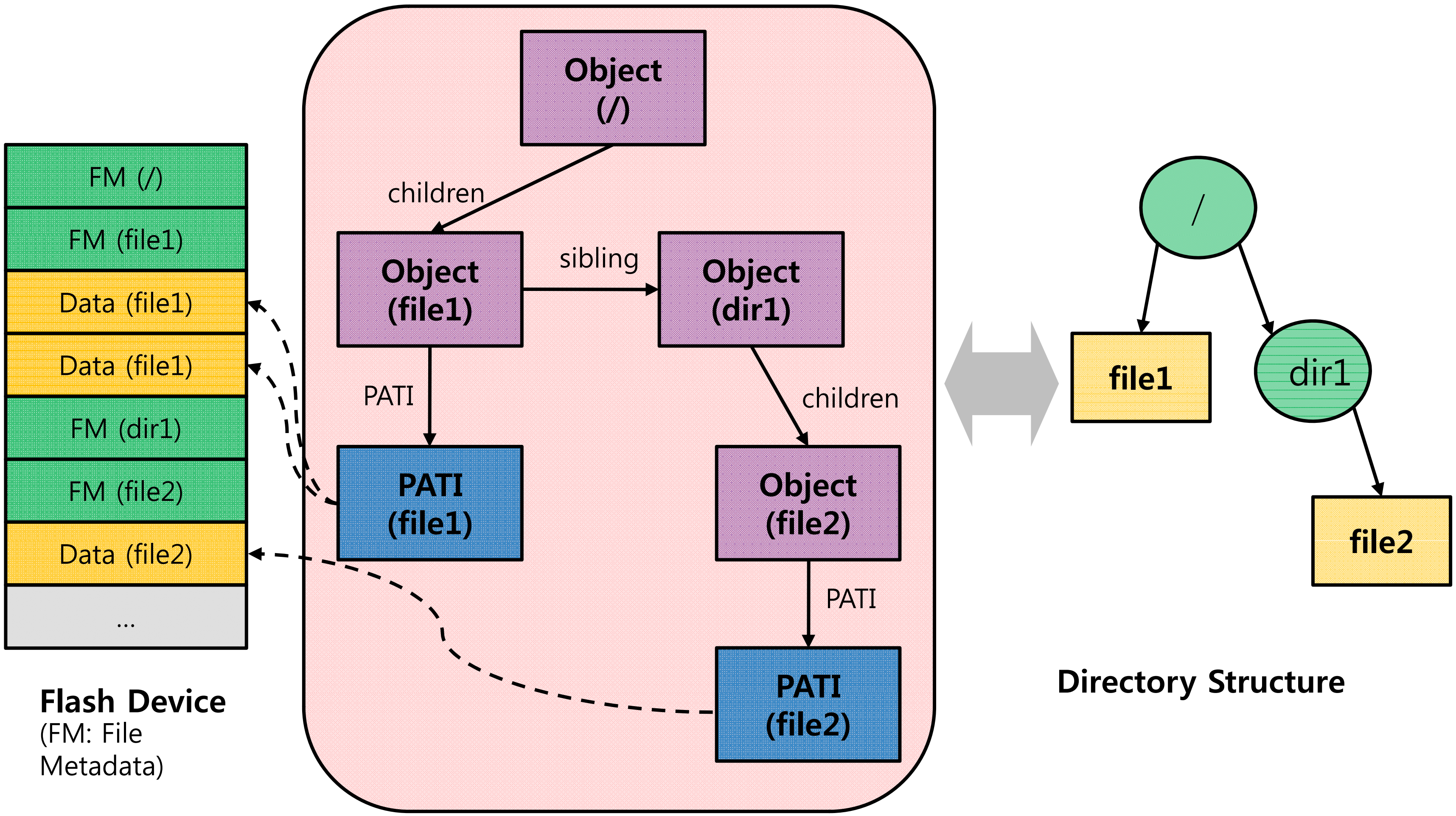
Native file system approach terminology

- Object: File/directory
- PATI (Physical Address Translation Information)
  - logical address → physical address
- File metadata(File metadata, e.g. inode)
  - Object type, Name, File size, Etc
- Page metadata(PM)
  - Block status – Information about bad block
  - Data status – Information about invalid page
  - Page ECC, Page information tuple

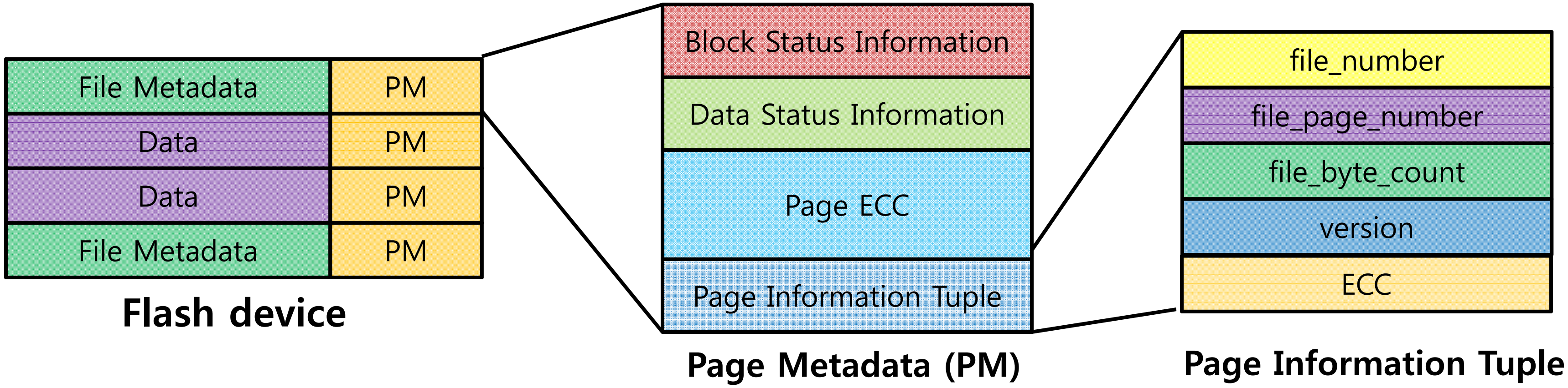
Data Structures in native File system for FLASH



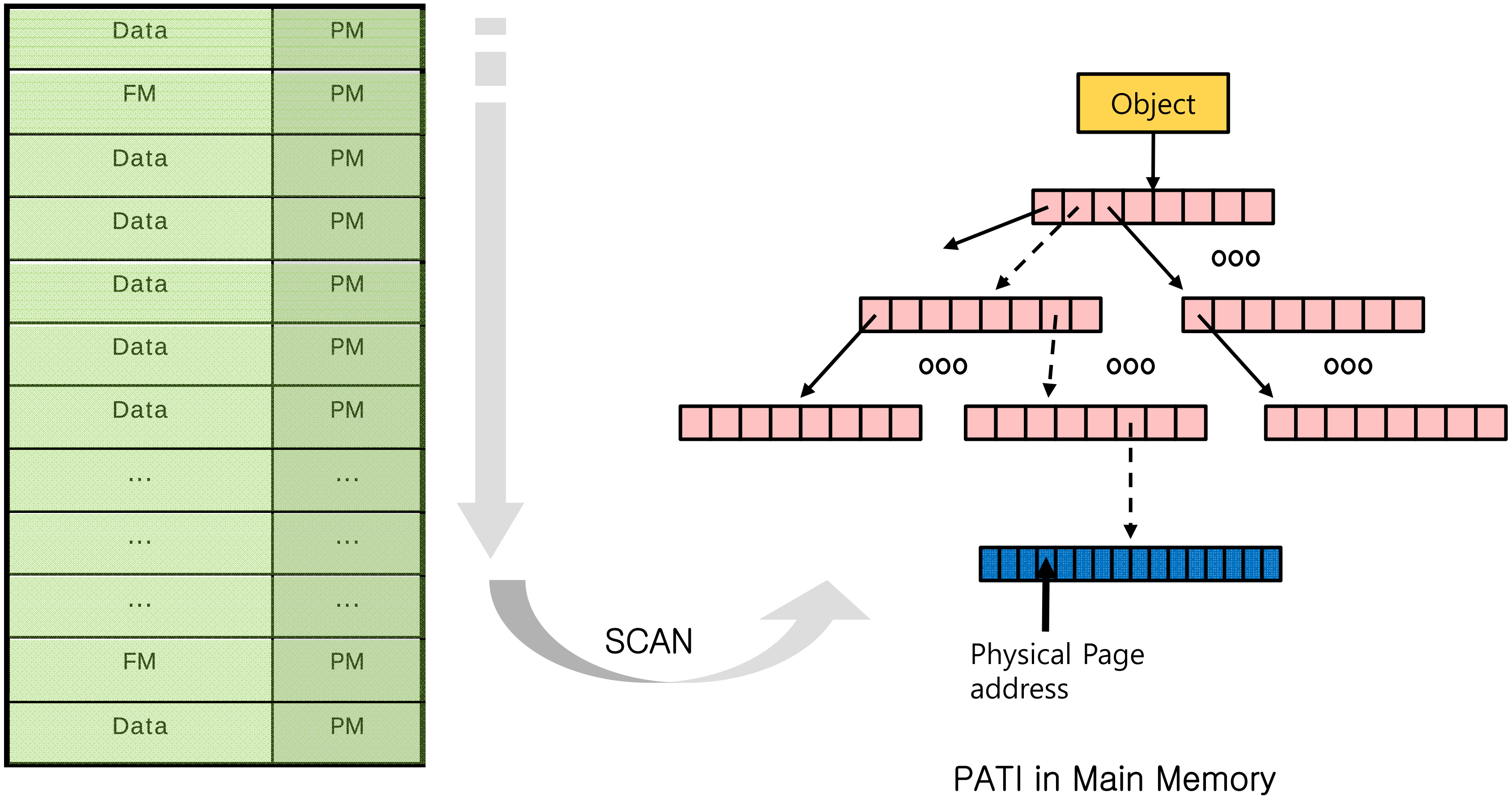
Data Structures in main memory



Data Structures in Flash Device



Mount Operation: disk → in-core



Objective

- Use byte-addressable RAM and FLASH
- Exploit the physical characteristics of them.!!!

- Better file system

FRASH: File system for FRAM and Flash

Design of Hierarchical file system for Hybrid storage

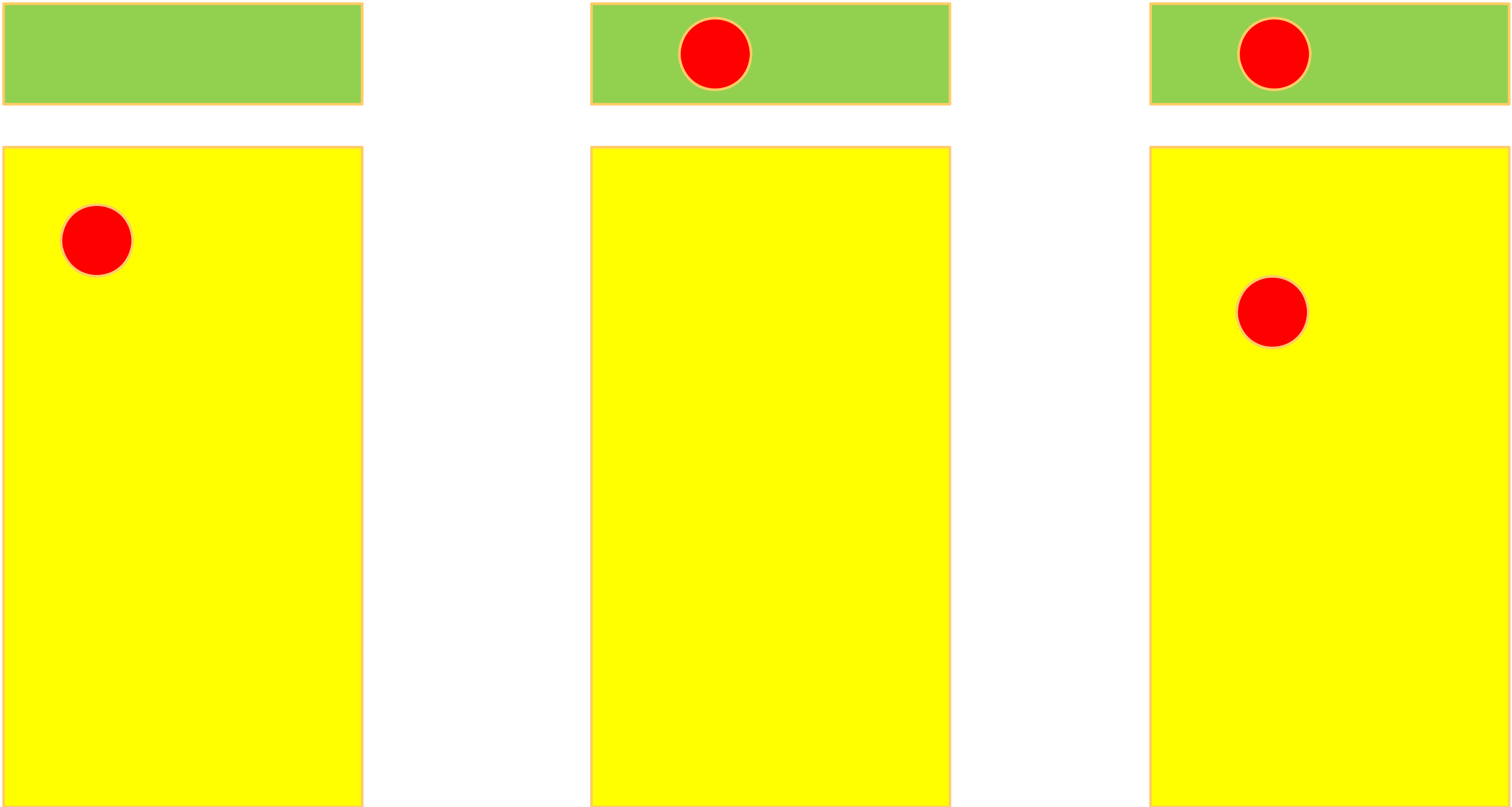
- Design choice

  - Location of each file system component

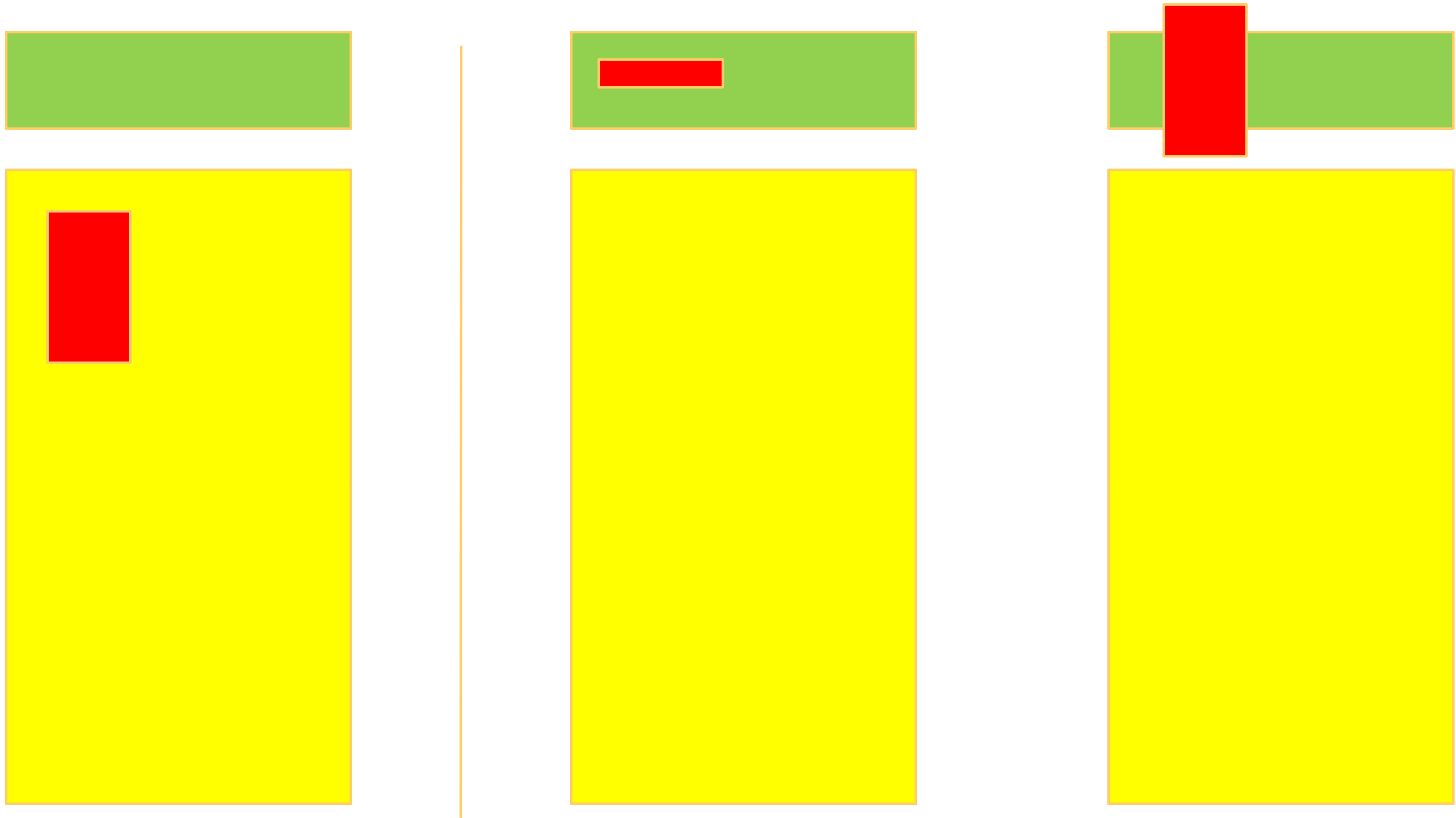
  - Device-friendly data structure of file system component

  - View on the byte-addressable NVRAM
    - Block device vs. RAM

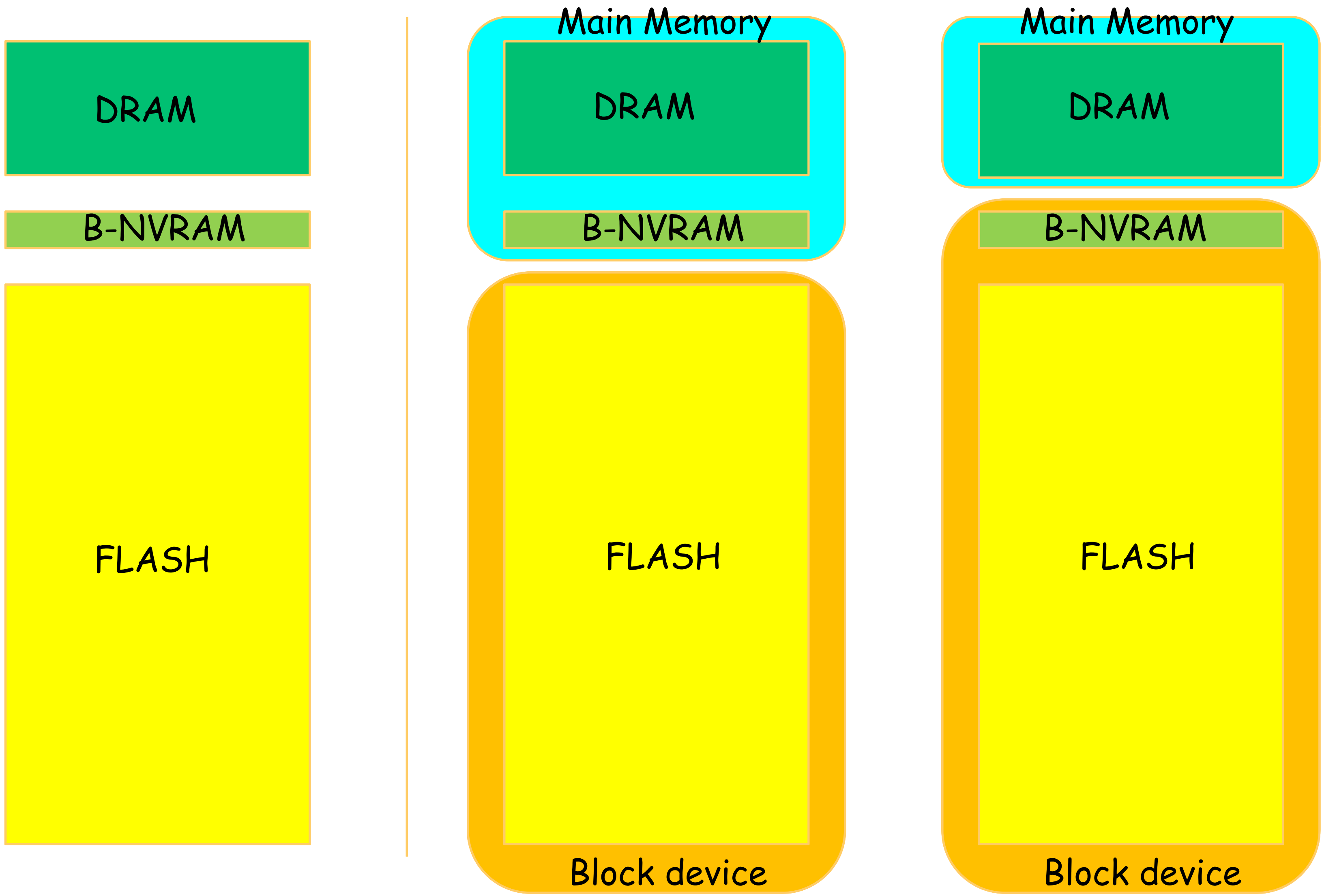
Location of file system component



Device Friendly Data Structure



B-NVRAM: memory vs. storage



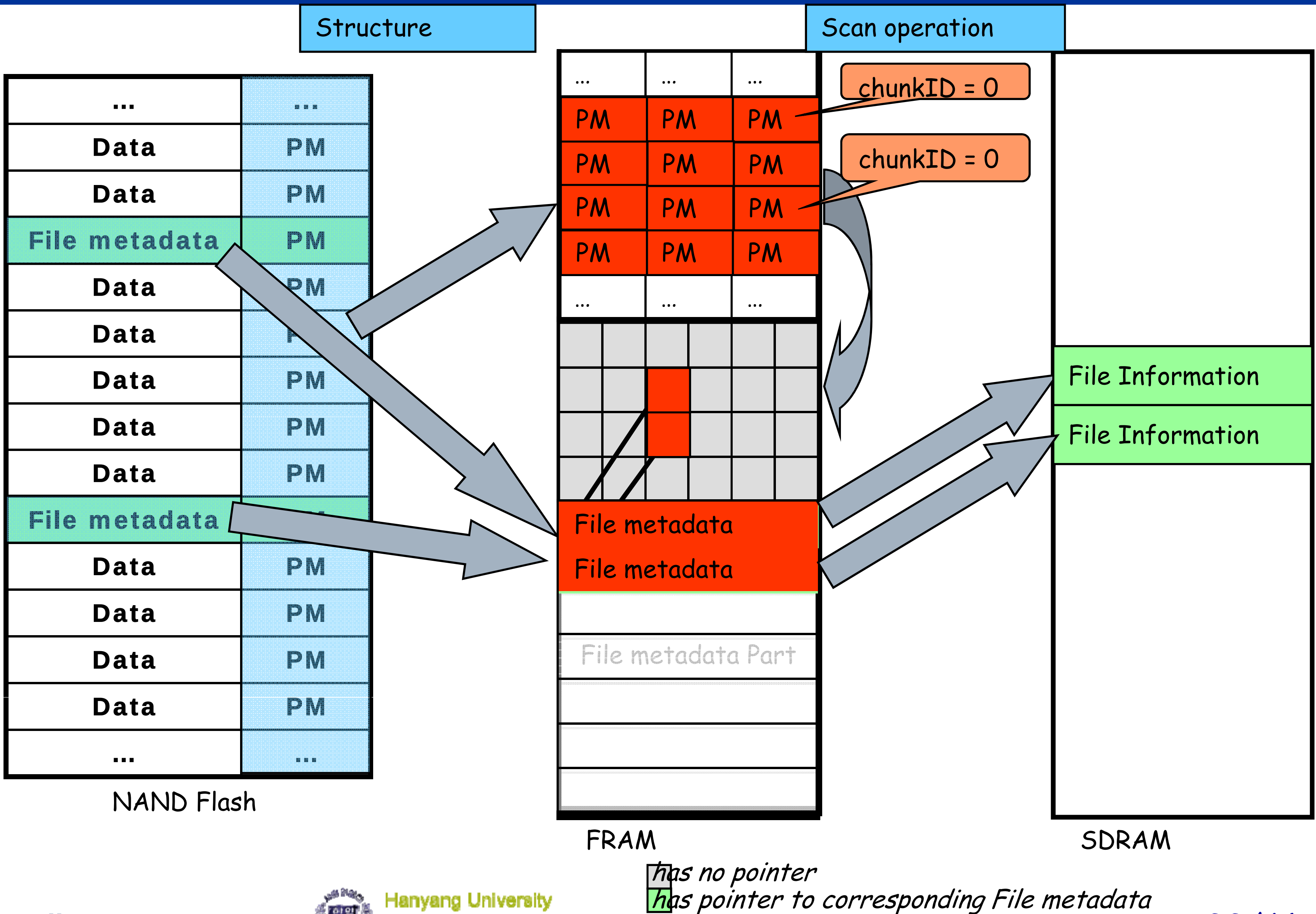
Design Choice 1: Metadata at NVRAM

- Objective: Reduce the file system mount delay

- File metadata and page metadata in b-NVRAM
  - Avoid flash scan overhead in file system mount
  - Data in NAND FLASH → NVRAM

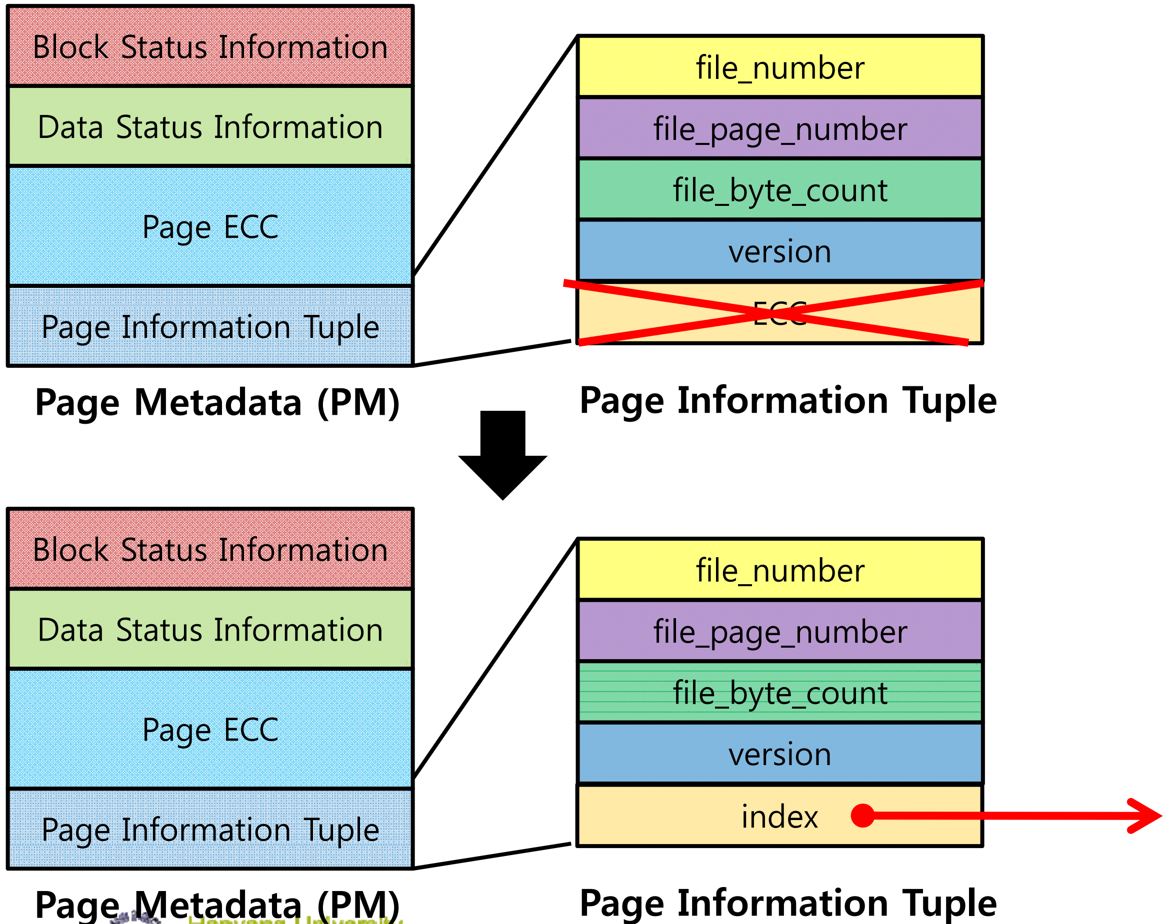
- Issue
  - Synchronization overhead between different storage hierarchy(byte-addressable NVRAM and NAND Flash)

Metadata at NVRAM(without modification)

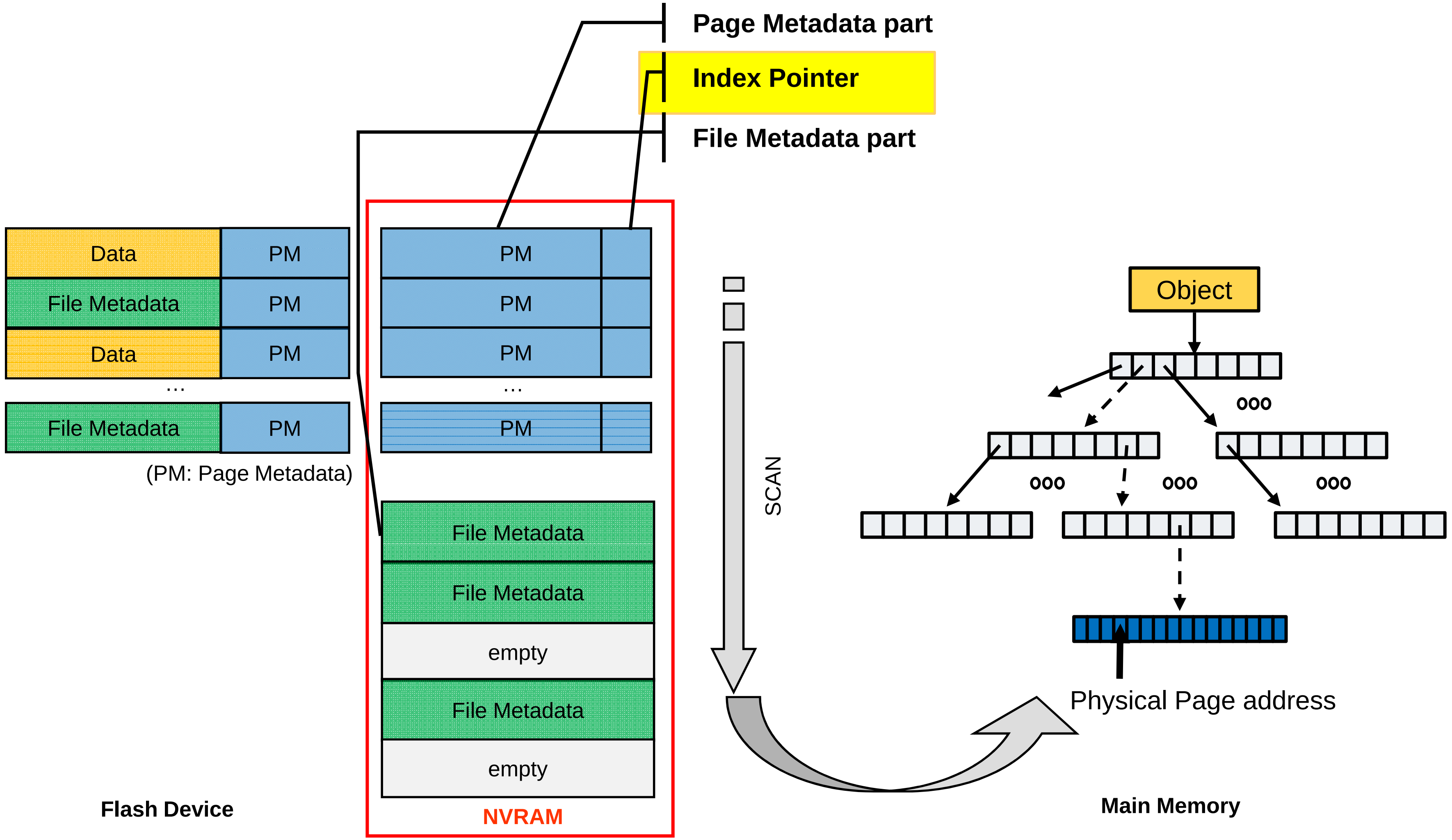


Issues

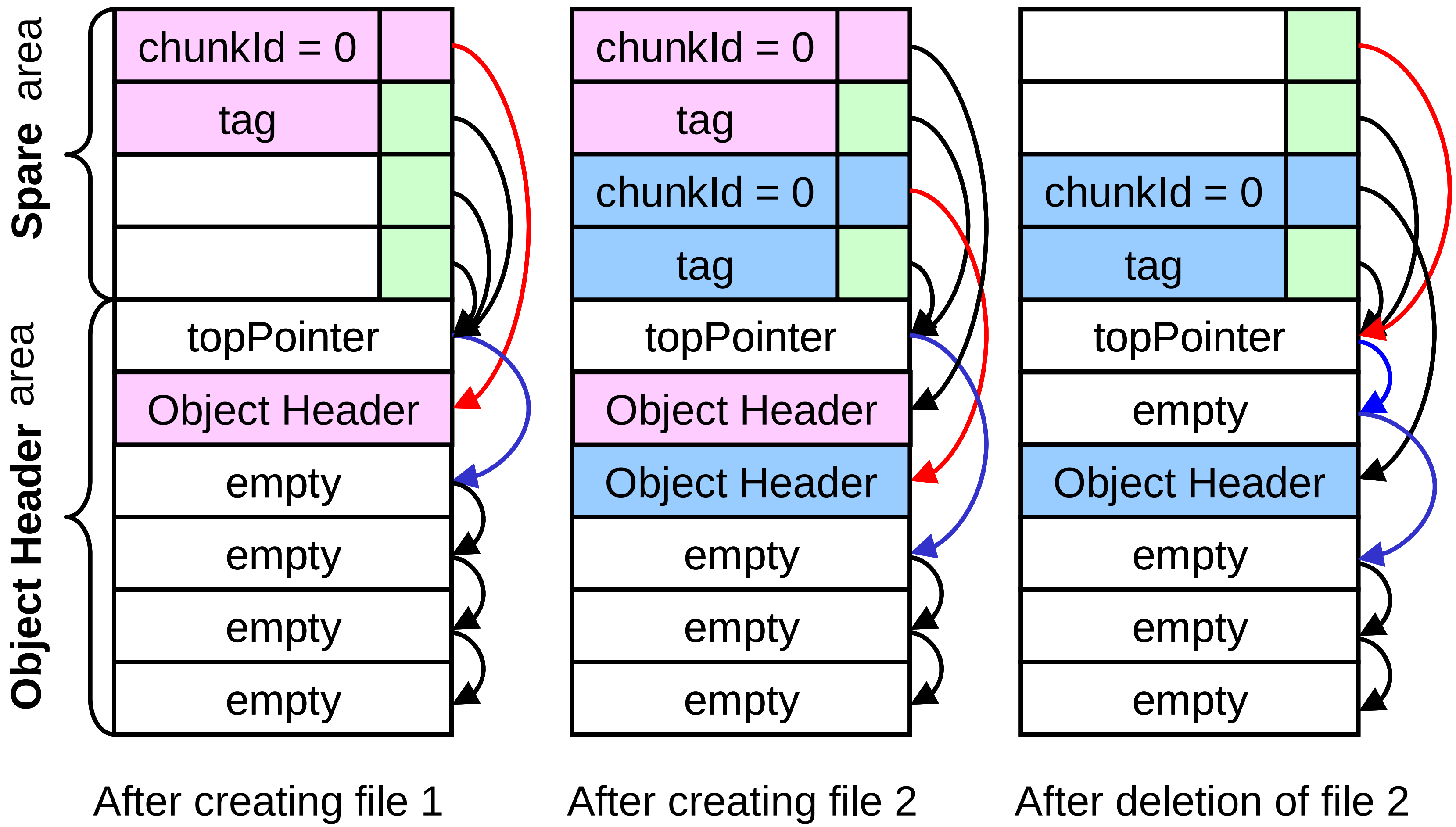
- We do not need ECC for data at FRAM.
- Remove one level of indirection for accessing FRAM data.



Metadata at NVRAM(without ECC in PM)



B-NVRAM data structure



Design Choice 2: right Data Structure

- Why disk data structure in b-NVRAM?

- File system mount
  - Scan the metadata(FRAM, NAND , or whatever),
  - Parse it and translate it into RAM-friendly structure a.k.a. in-core data structure(file system mount).

What is the right data structure for b-NVRAM?

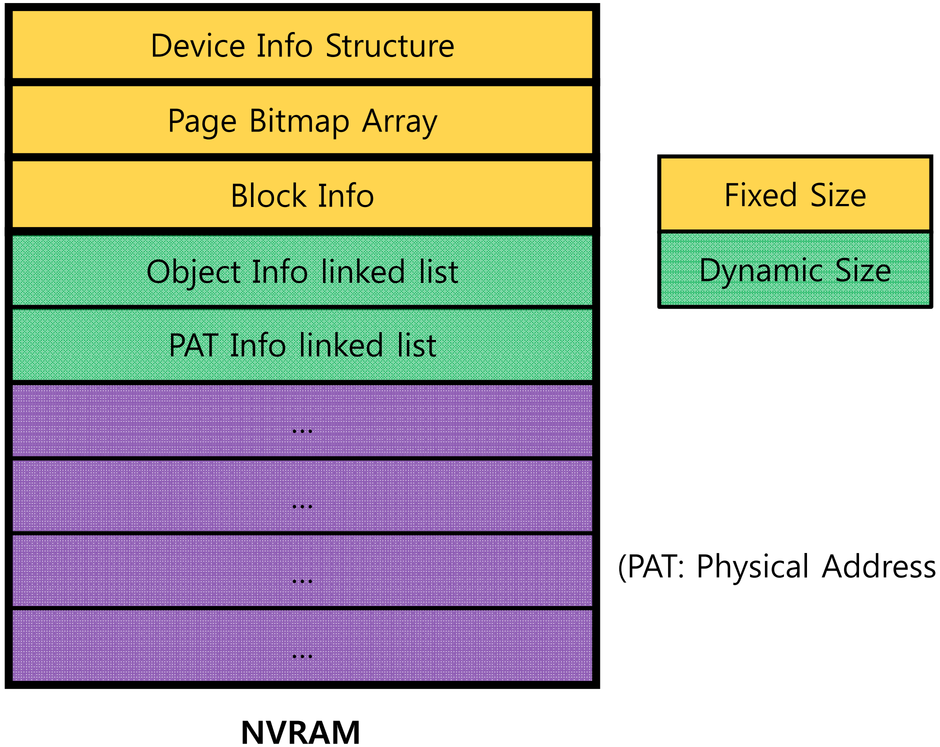
Exploiting Byte-addressability

Use RAM-like representation of file system in byte addressable NVRAM.

- File system accesses byte addressable NVRAM directly instead of accessing RAM.
- Issues
  - Synchronization problem. Flash memory still has metadata.
  - Performance degradation: B-NVRAM is slower than DRAM.

RAM-like Data Structure

- Device Info Structure – Partition status information
- Page Bitmap Array – Information about page in-use or not
- Block Info – Block Status Information



(PAT: Physical Address Translation)

Design Choice 3: Block Device vs. RAM

- B-NVRAM: shall we see it as Block device or RAM?

Both of them?

In fact, neither of them!!!

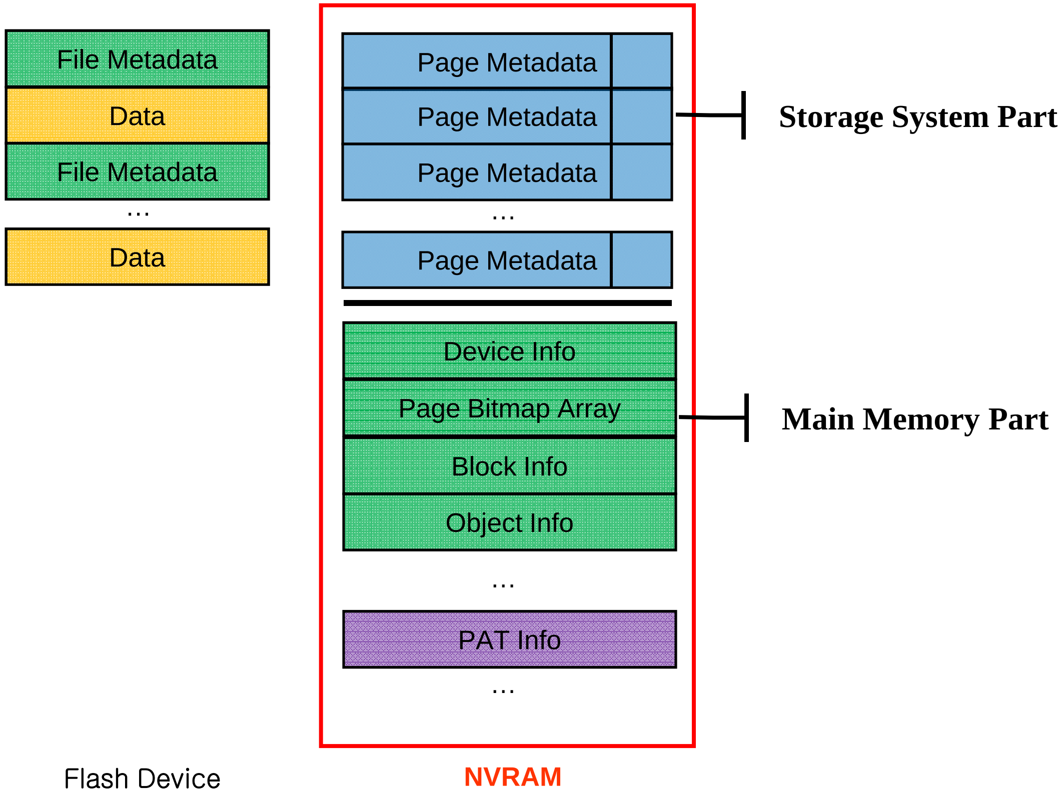
Final Design

- All page metadata and file metadata is moved to FRAM from flash memory.
  - Flash memory no more have file metadata and page metadata.
  - File system does not need to access flash memory when metadata operation is executed.

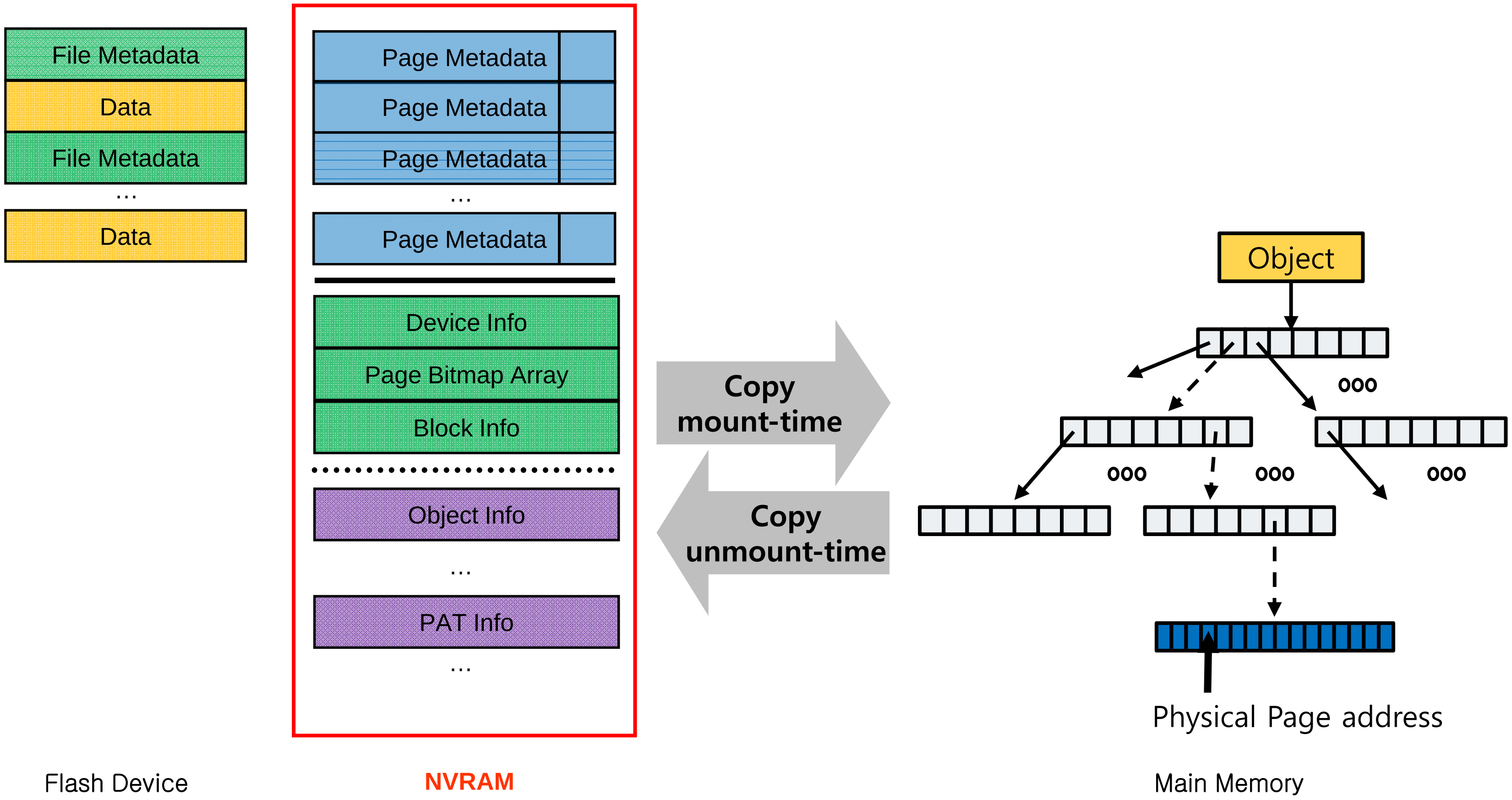
Final Design

- Exploit RAM!
  - Maintain all data structures used by kernel in FRAM
  - Copy(not mount) to DRAM at Mount time.
  - DRAM data is copied to FRAM at Unmount time
- Synchronization between FRAM and DRAM is coarse.
  - No defense and recovery method is implemented in crash condition.

Final Design



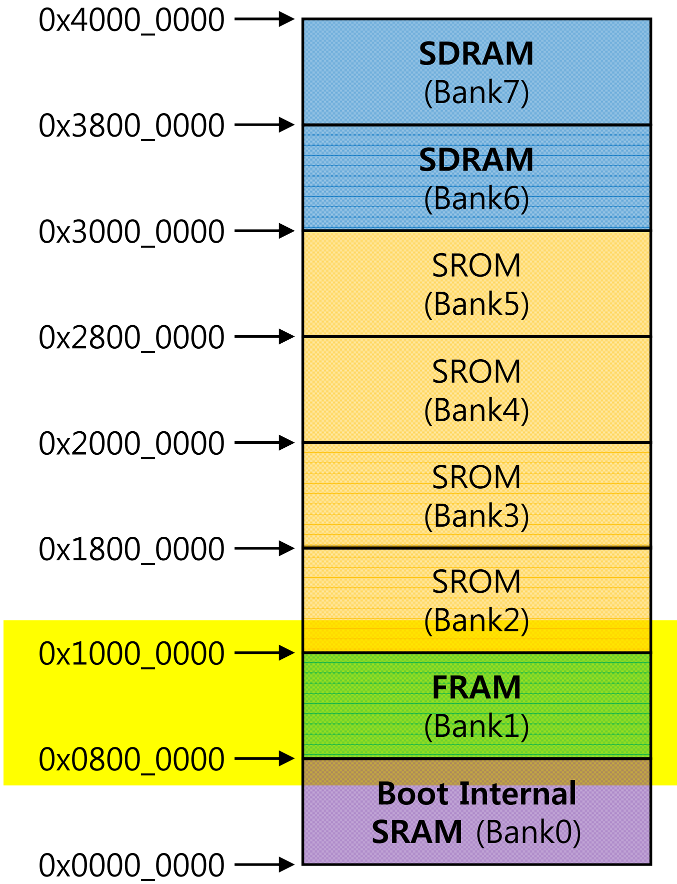
Mount Operation at Final Design



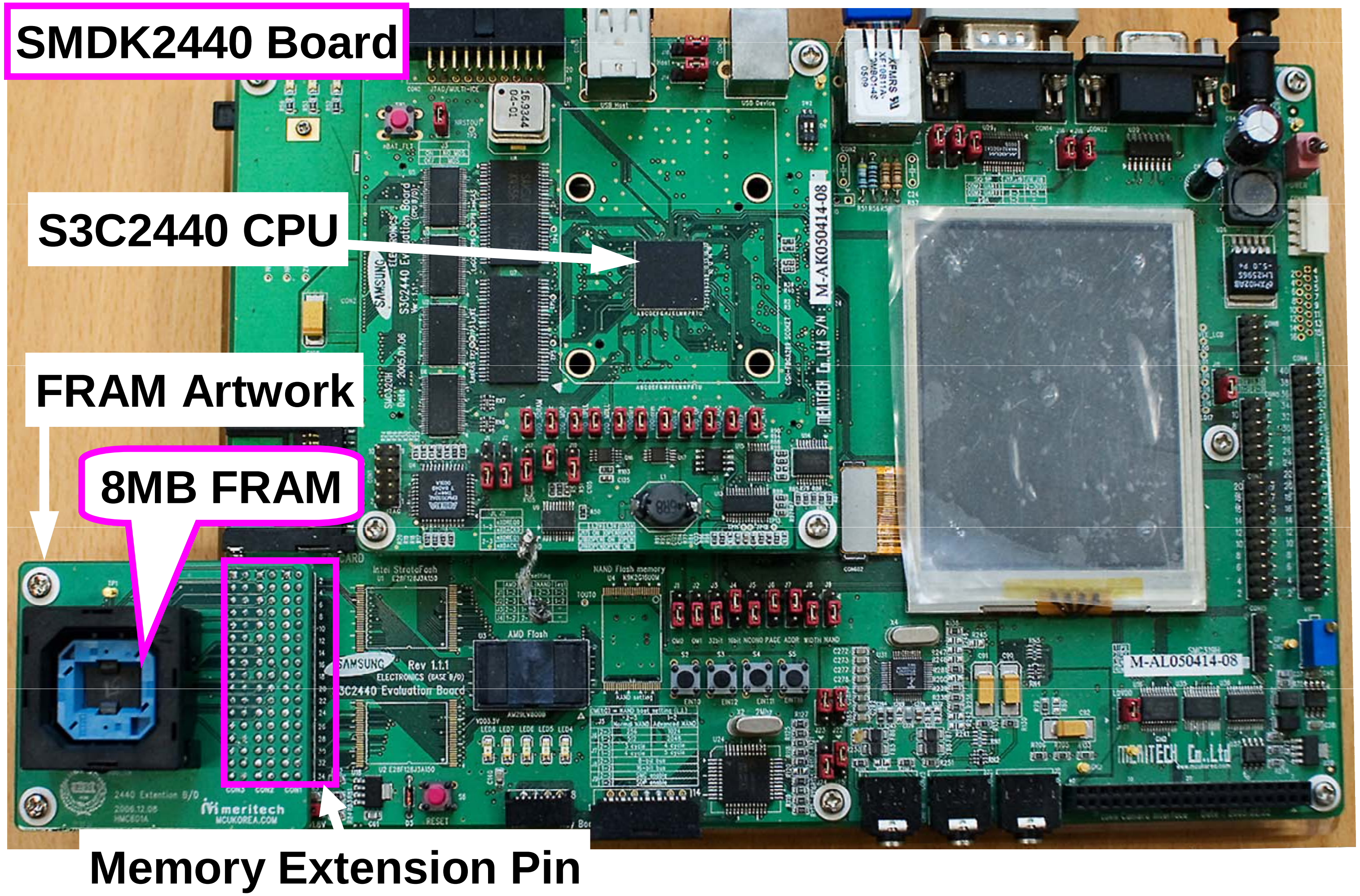
Implementation

- Hardware: SMDK 2440
  - Core clock: 400 MH
  - Memory bus: 100 MHz
- OS: Linux 2.6 kernel
- Hierarchical Storage
  - 64 Mbit FRAM, 5.6 MHz(180 ns)
  - 128 Mbyte NAND Flash(Smart Media Card)

Memory Map in SMDK2440

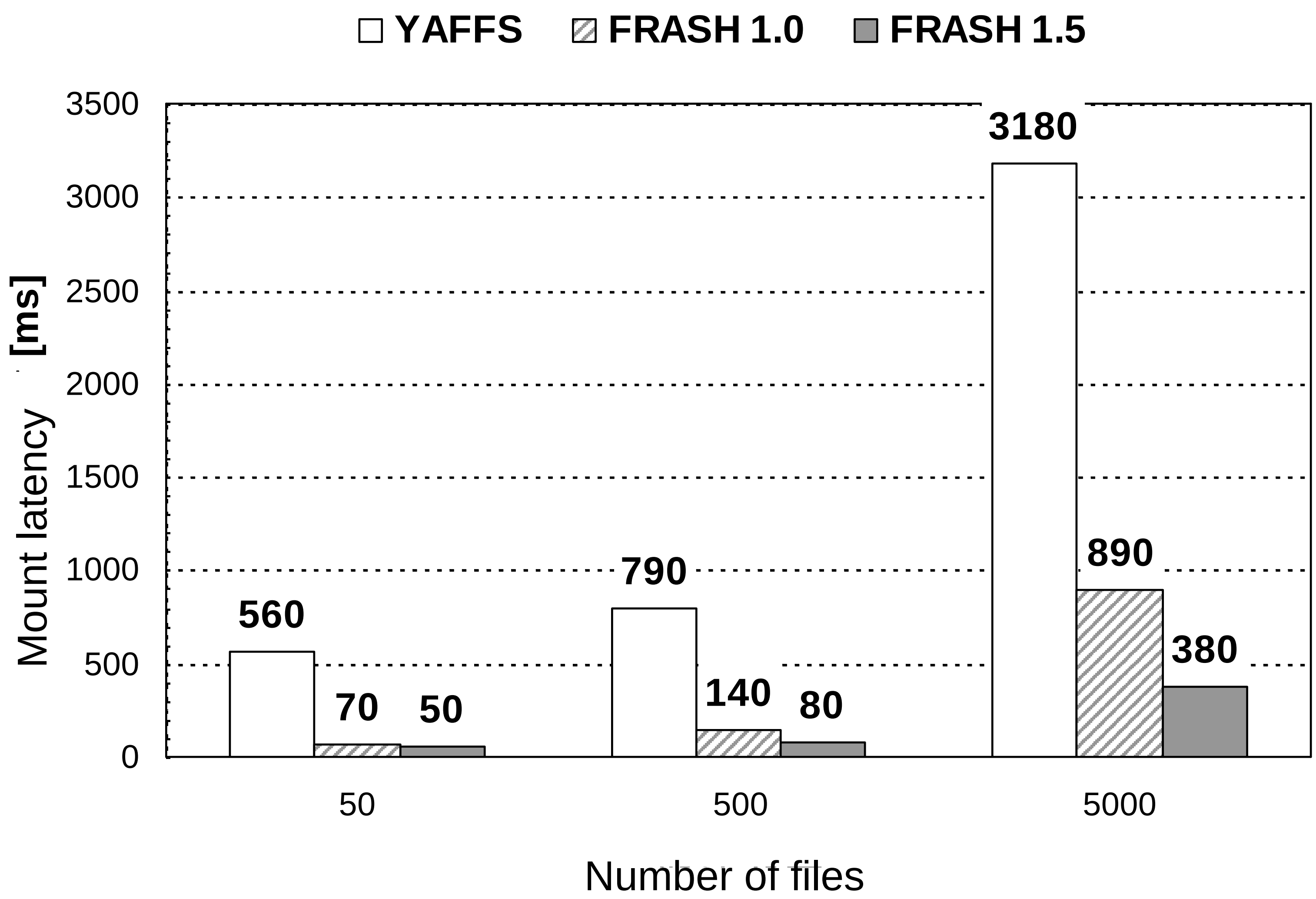


Memory Map in SMDK2440

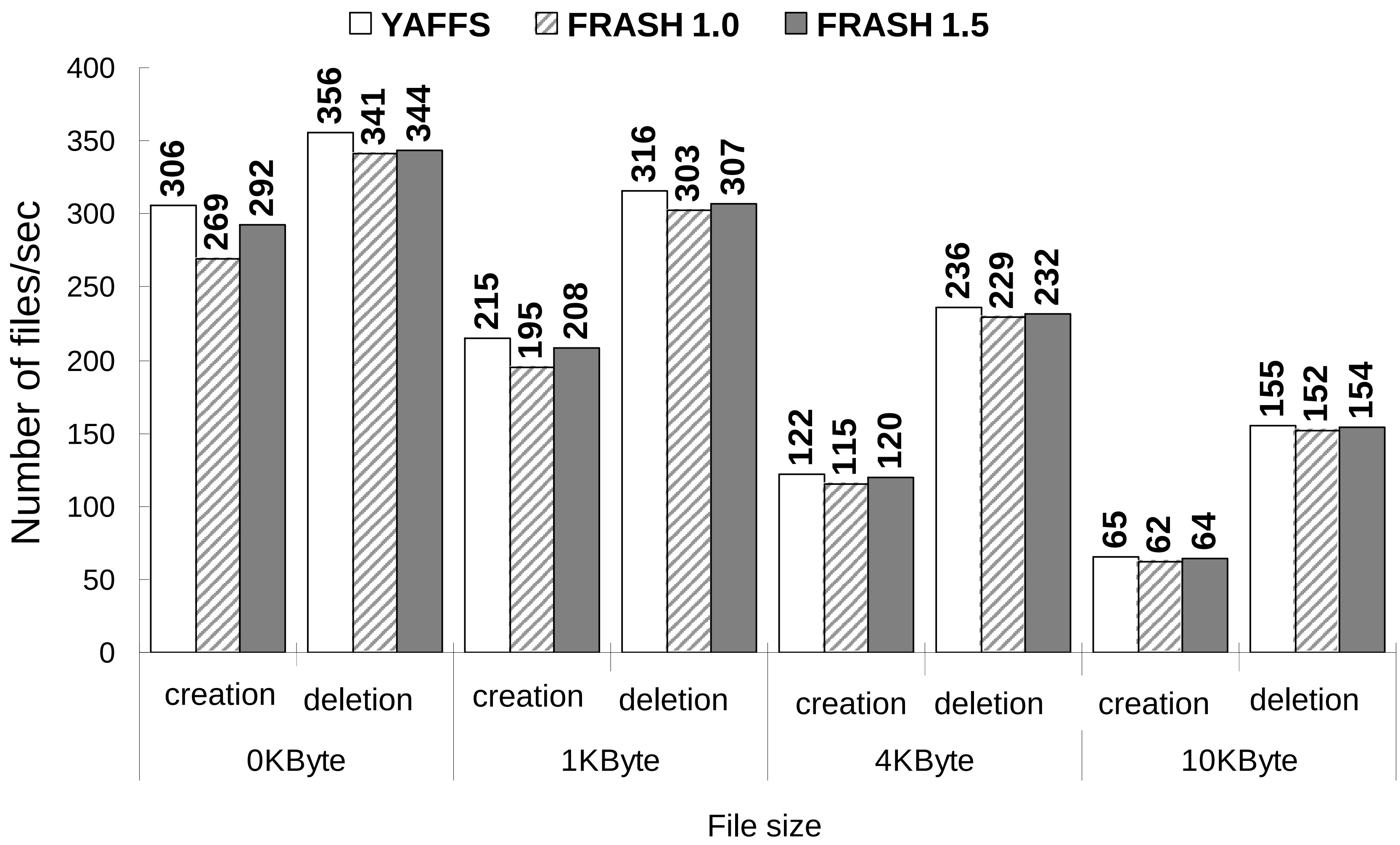


- Effect of Maintaining Metadata at FRAM
  - With and without ECC
  - Redundancy issue

Effect of Removing ECC and level of indirection

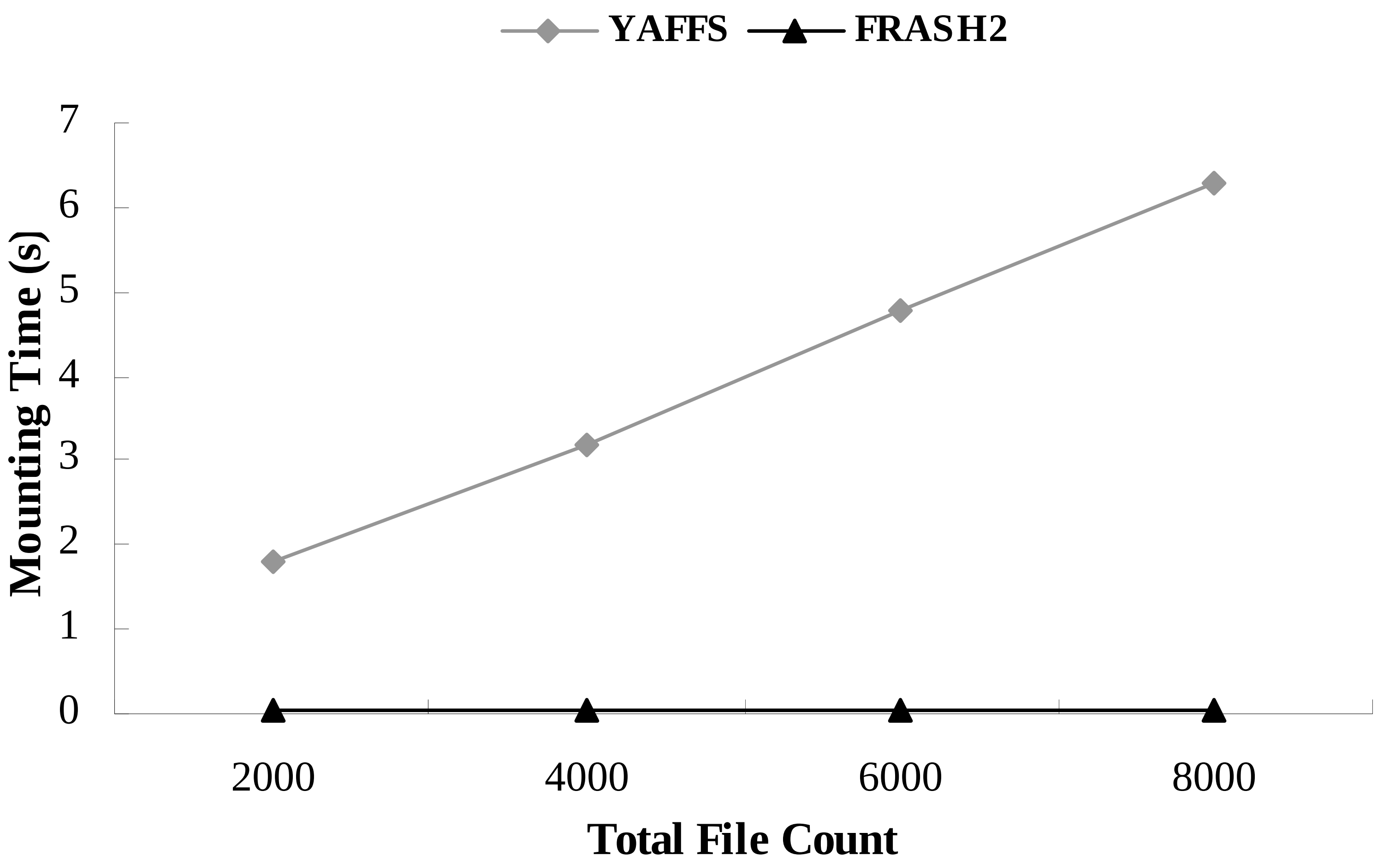


Effect of Removing ECC and level of indirection: metadata update

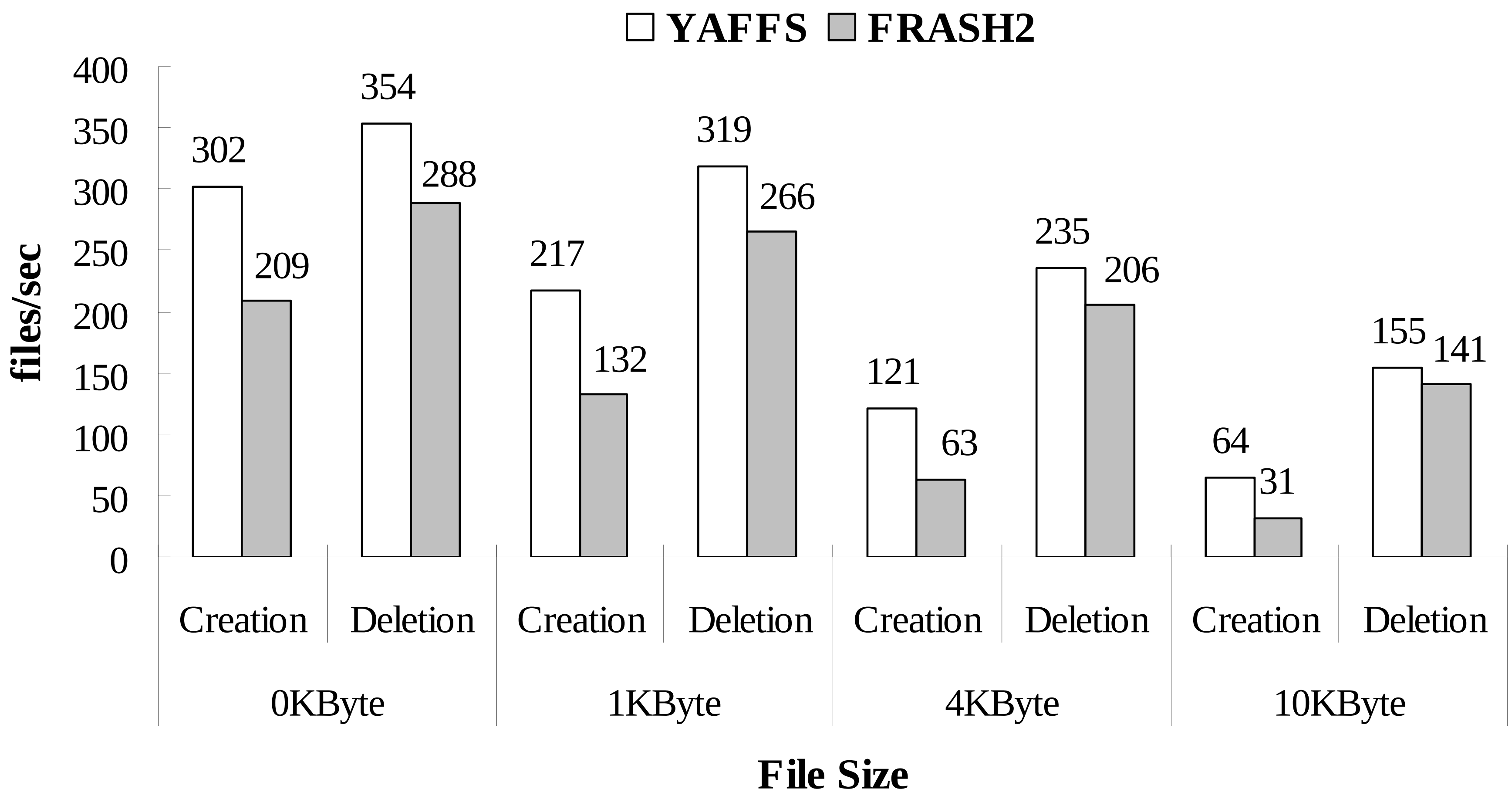


- RAM-friendly data structure
  - RAM-like data structure
  - Updating FRAM, not DRAM

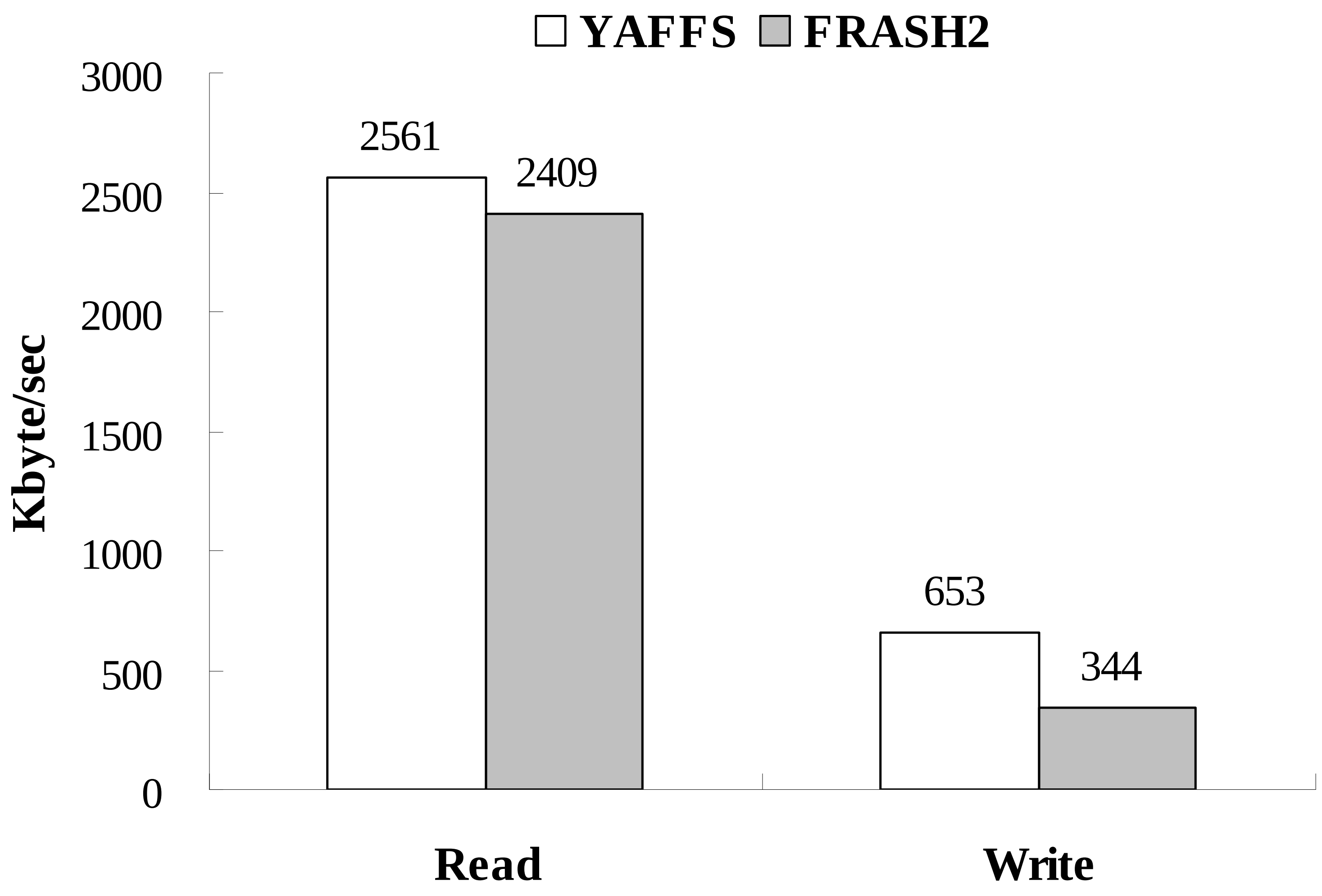
Mount latency



Metadata update

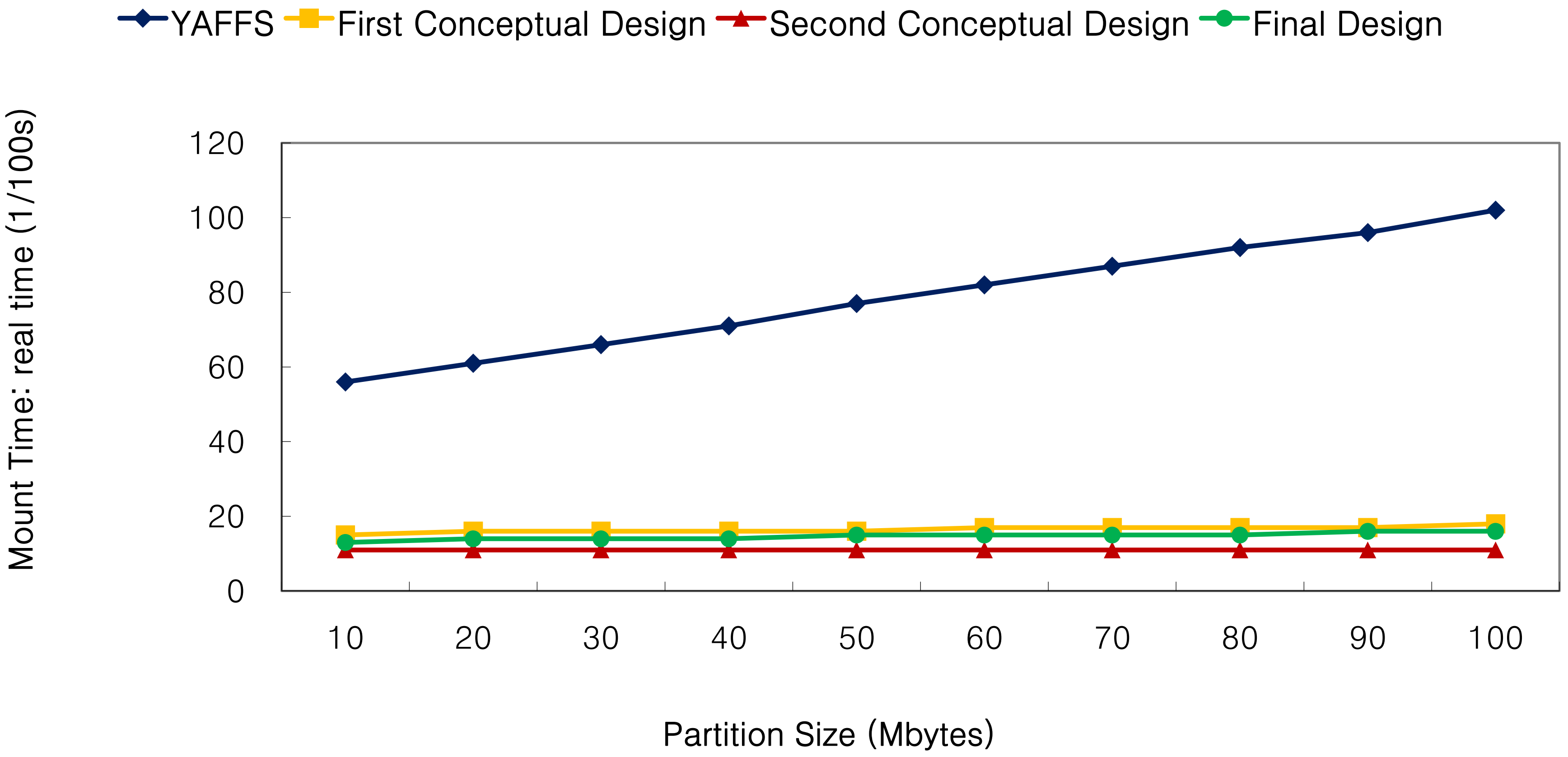


Data I/O

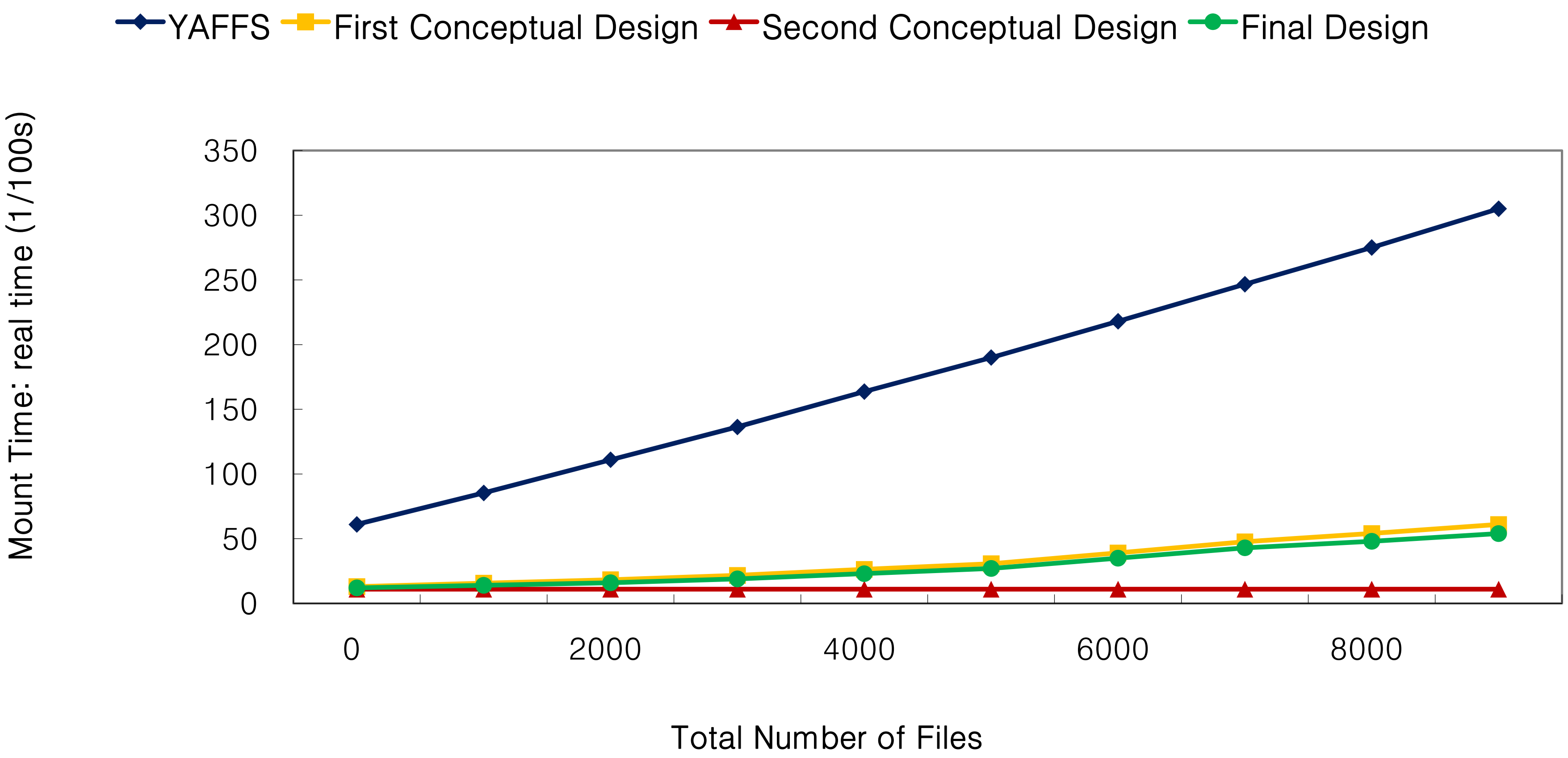

YAFFS
FRASH2
Kbyte/sec
3000
2500
2000
1500
1000
500
0
2561
2409
653
344
Read
Write

- Overall
  - Yaffs
  - Choice 1: Metadata in b-NVRAM, no ECC in b-NVRAM
  - Choice 2: Direct access on b-NVRAM
  - Choice 3: Adaptive Layer Selection

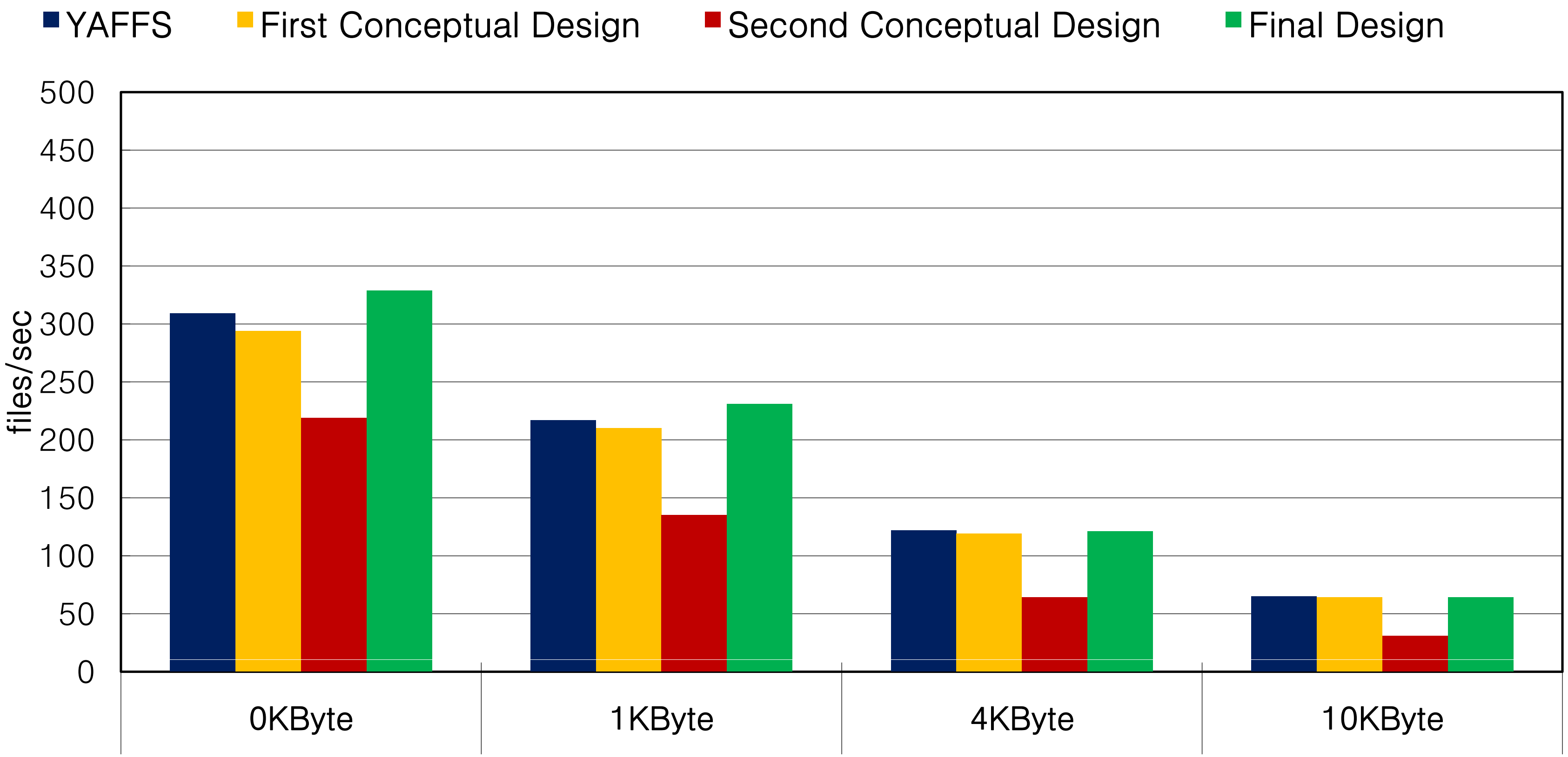
Mount Delay with different partition size



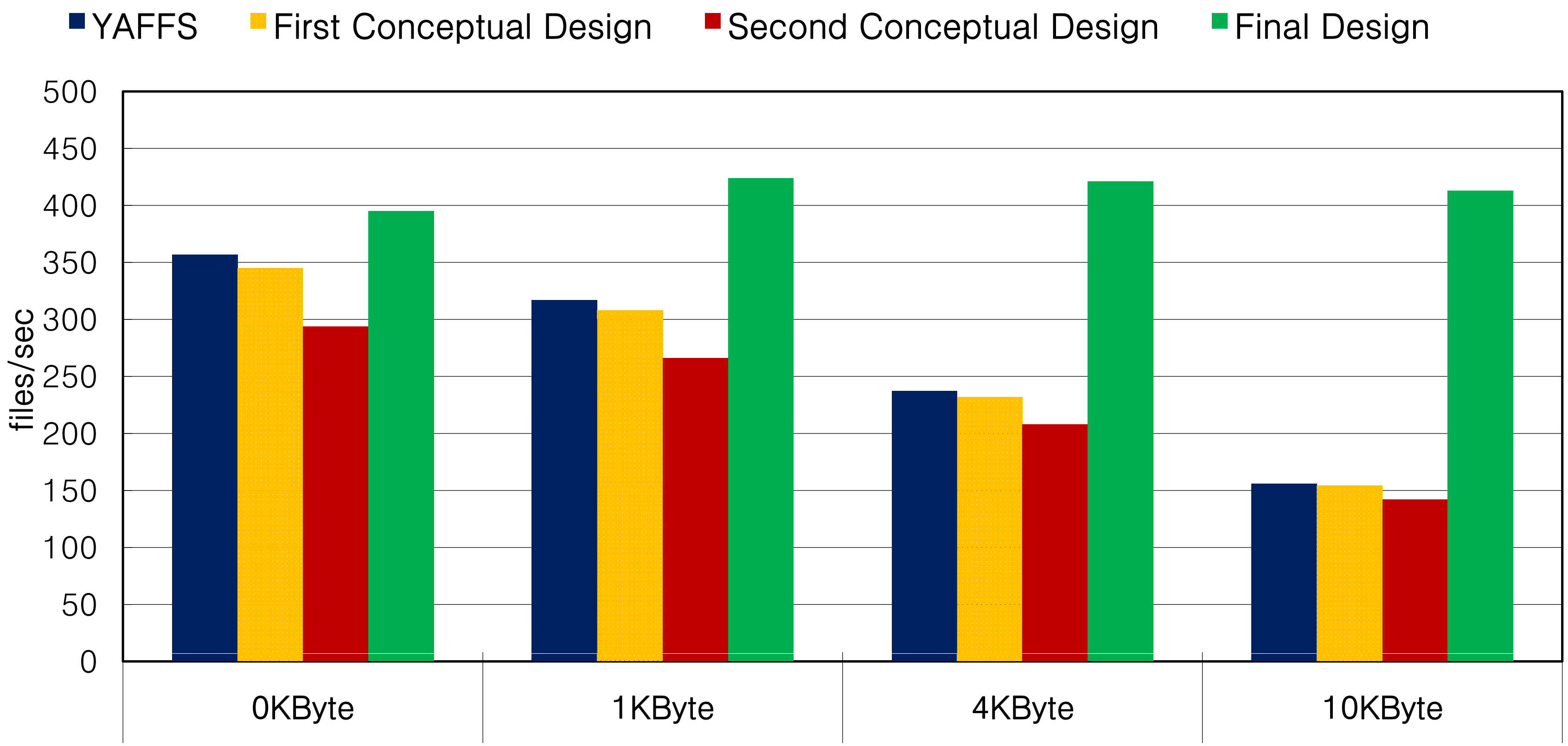
Mount Delay with different number of files



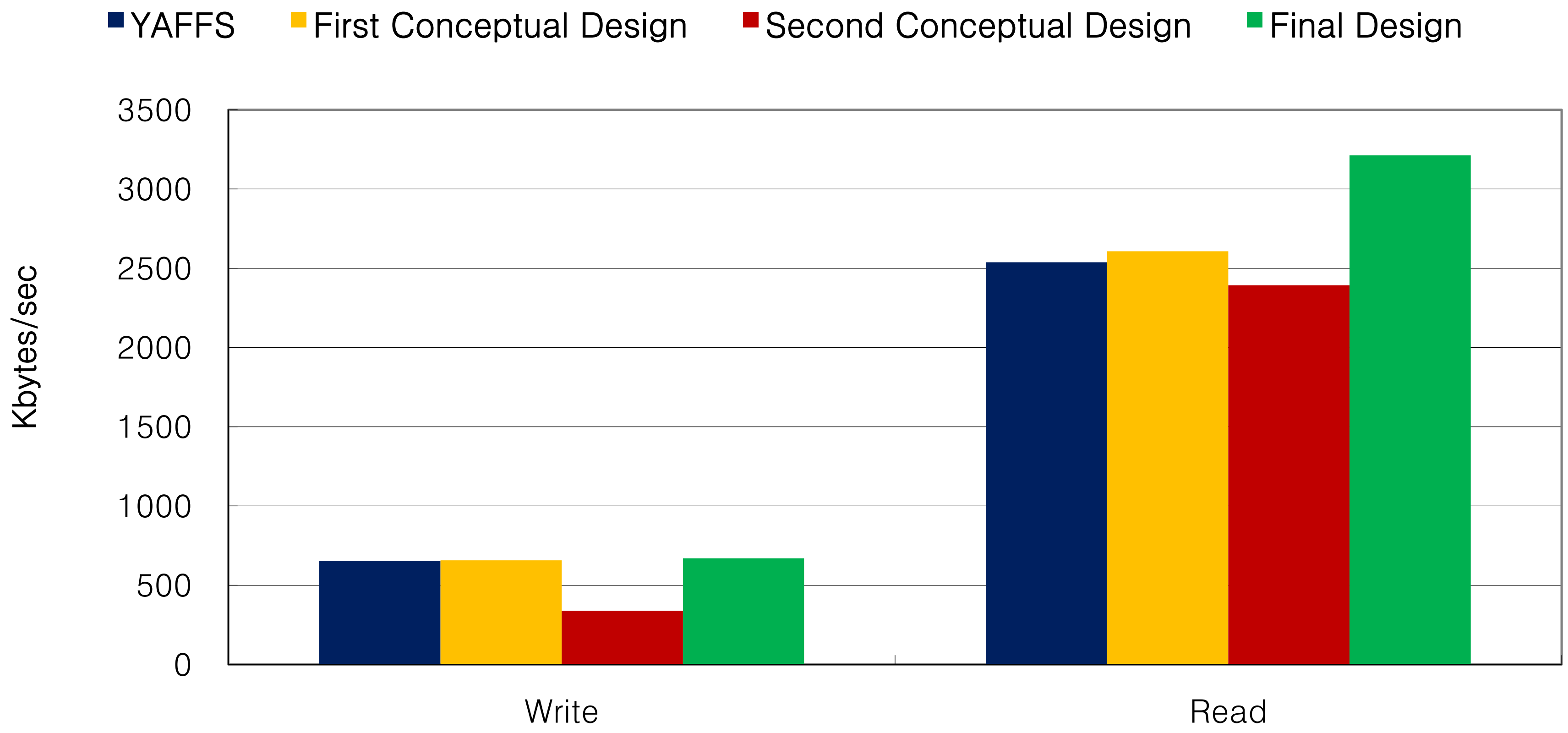
File Creation (LMBENCH)



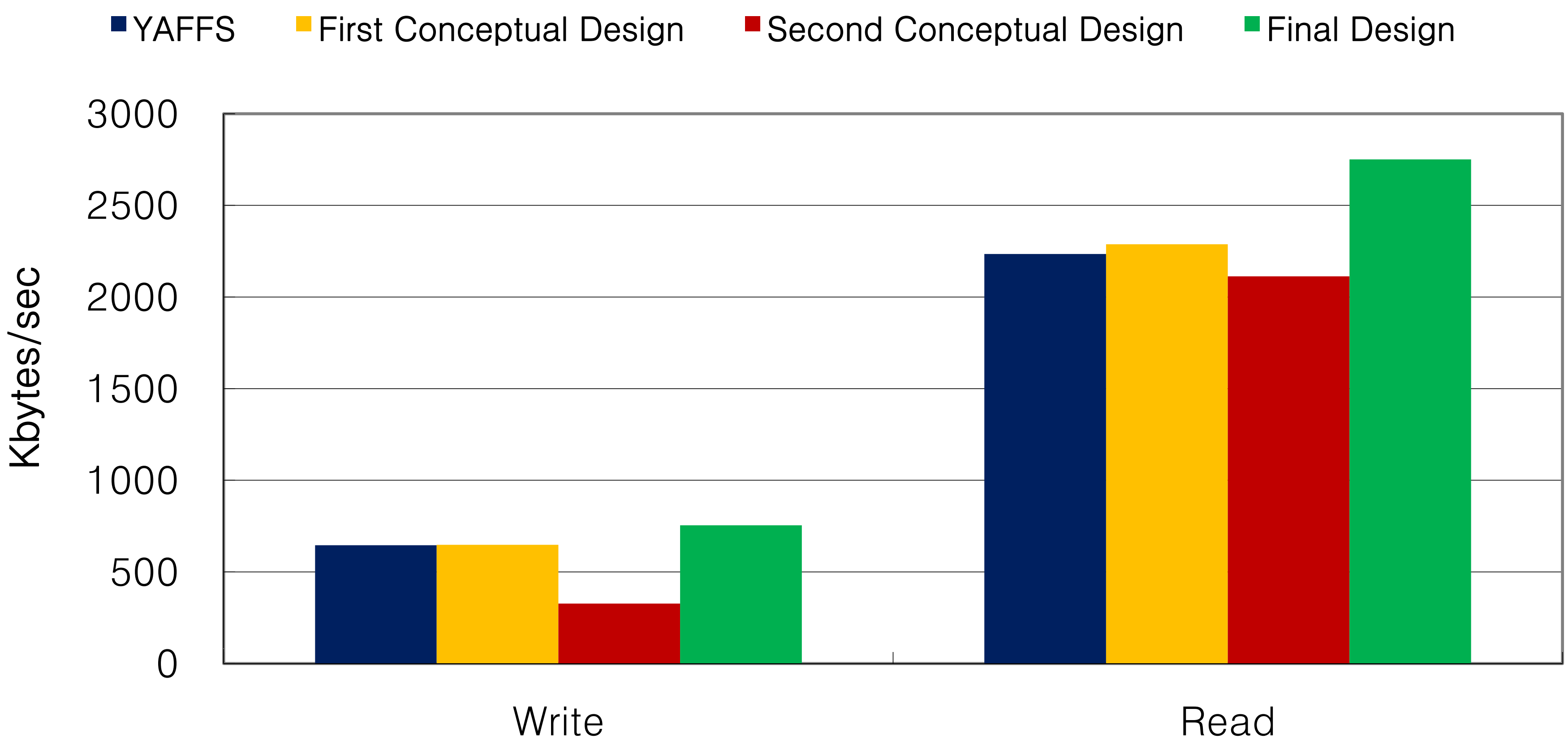
File Deletion (LMBENCH)



Sequential Read/Write (LMBENCH)



Sequential Read/Write (Iozone)



Conclusion

- Log Structured Approach for Flash
  - Long Mount Delay
- FRASH
  - Hierarchical File System with Flash and NVRAM
    - Fast scan operation
    - Elimination of scan operation
  - Faster Mount Time
  - Better Performance

Q&A

Thank you.