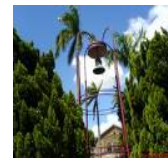
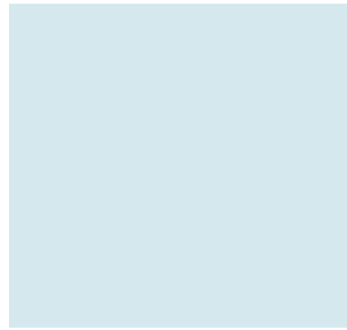
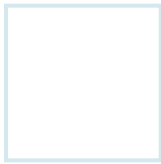


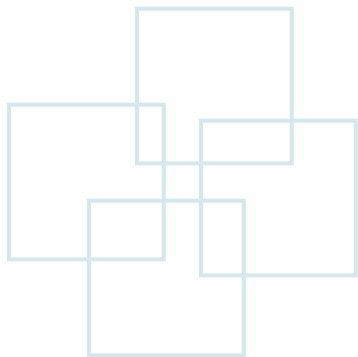
臺灣大學



A File-System-Aware FTL Design for Flash Memory Storage Systems

Po-Liang Wu, Yuan-Hao Chang, Po-Chun
Huang, and Tei-Wei Kuo

National Taiwan University



臺灣大學



Outline

- Introduction
- File Systems Observations
- The Proposed Flash Translation Layer Mechanism
- Experiments
- Conclusion

臺灣大學



Outline

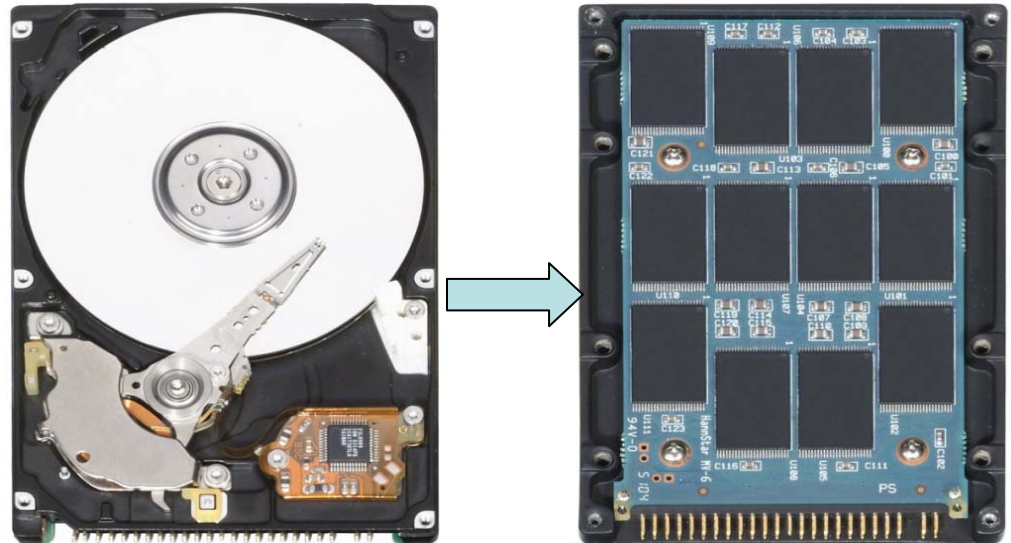
- Introduction
- File Systems Observations
- The Proposed Flash Translation Layer Mechanism
- Experiments
- Conclusion

臺灣大學



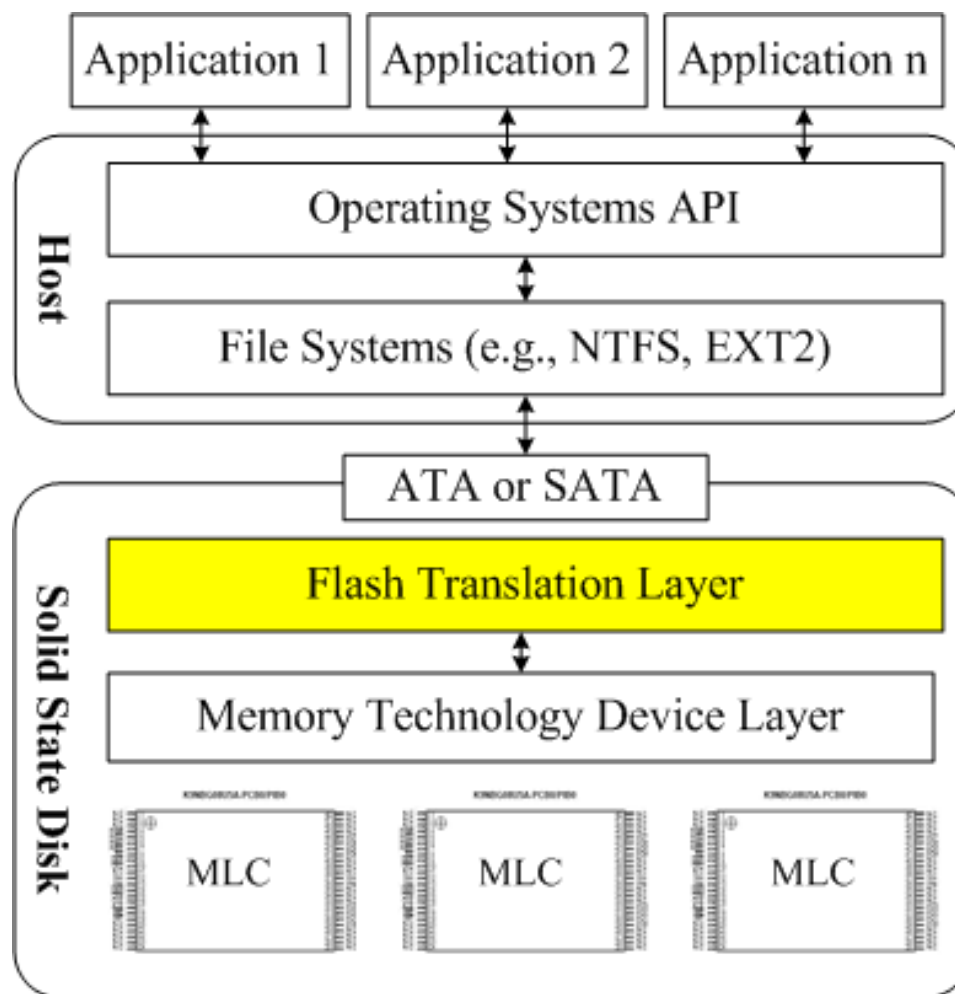
Introduction

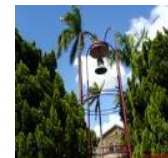
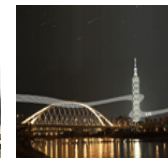
- Solid State Disks (SSDs) just get more and more attention for the replacement of traditional hard disks.
- Flash memory characteristics
 - Erase before write
 - Bulk erase





System Architecture





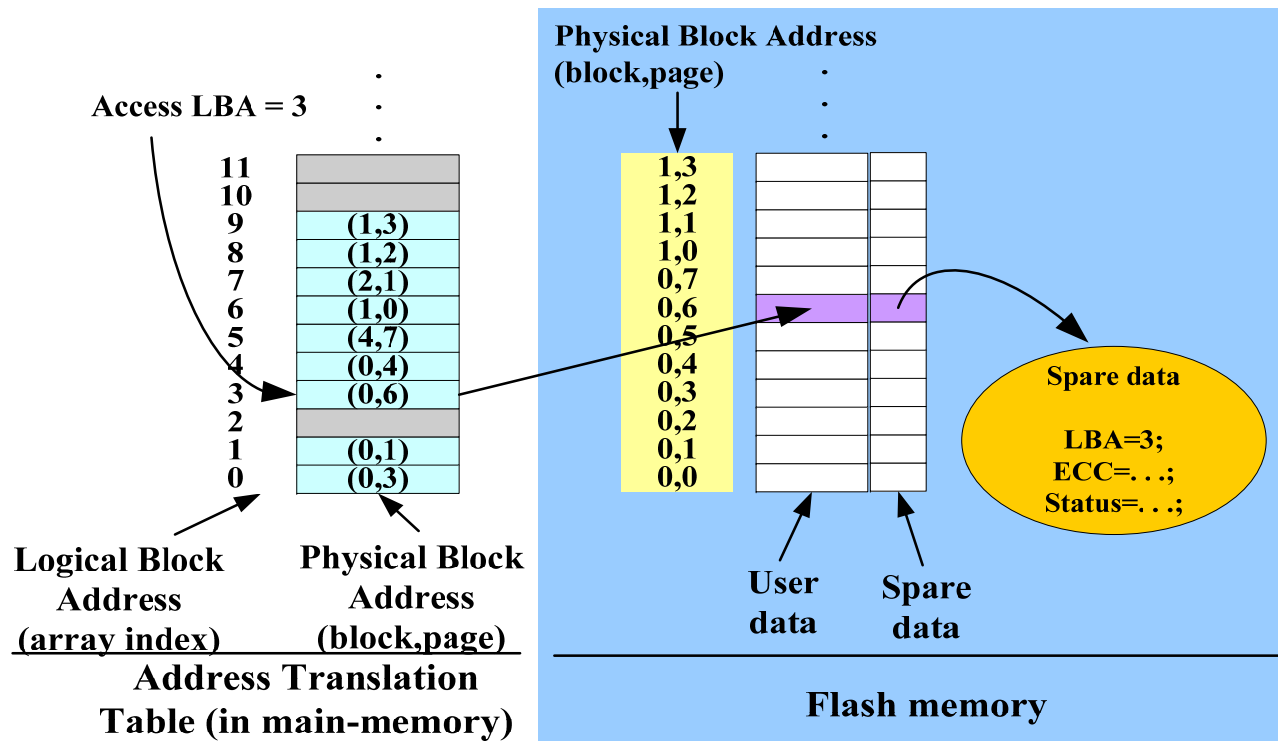
Flash Translation Layer

- Purpose of Flash Translation Layer
 - Model the underlying flash device as a block device.
 - Translate logical block addresses (LBAs) to physical block addresses (PBAs).
- Implementations of Flash Translation Layer
 - FTL: page-level mapping, significant memory consumption for mapping table
 - BL: block-level mapping, low performance
 - NFTL: block-level mapping with replacement block



FTL mechanism

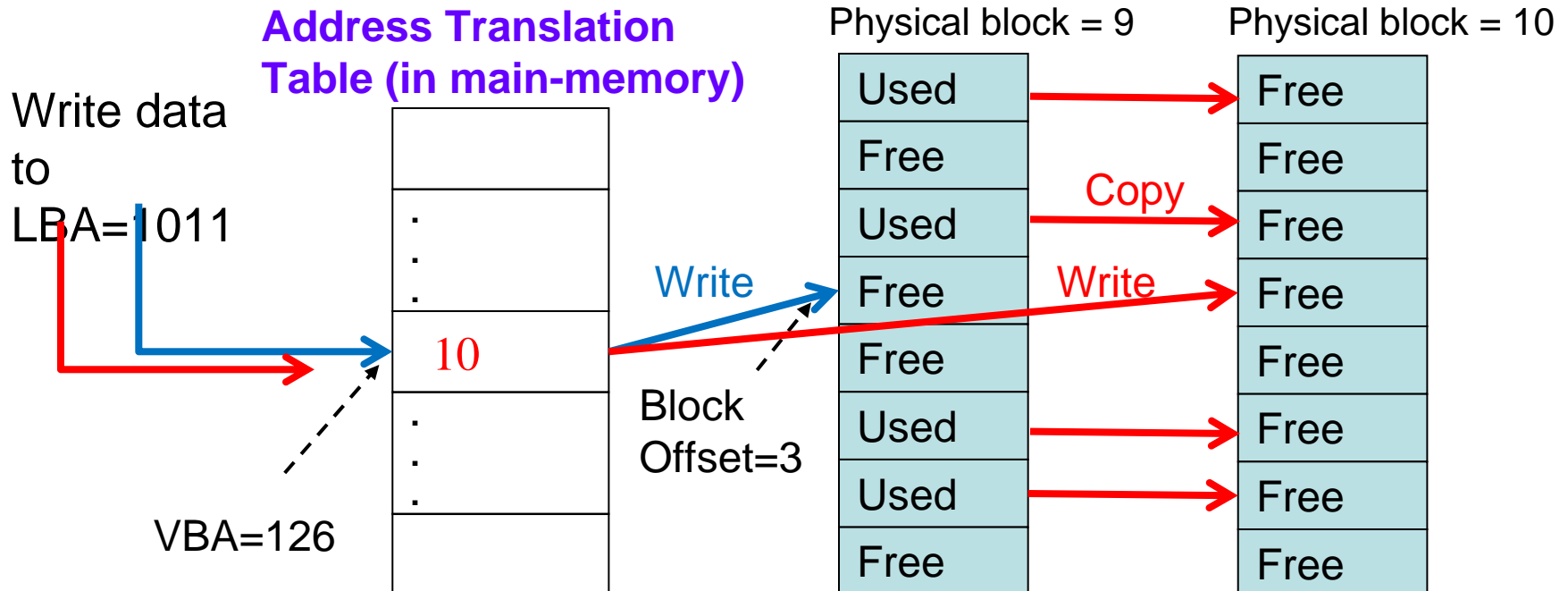
- FTL adopts a page-level address translation mechanism.
 - The main problem of FTL is on **large memory space requirements** for storing the address translation information.





BL Mechanism

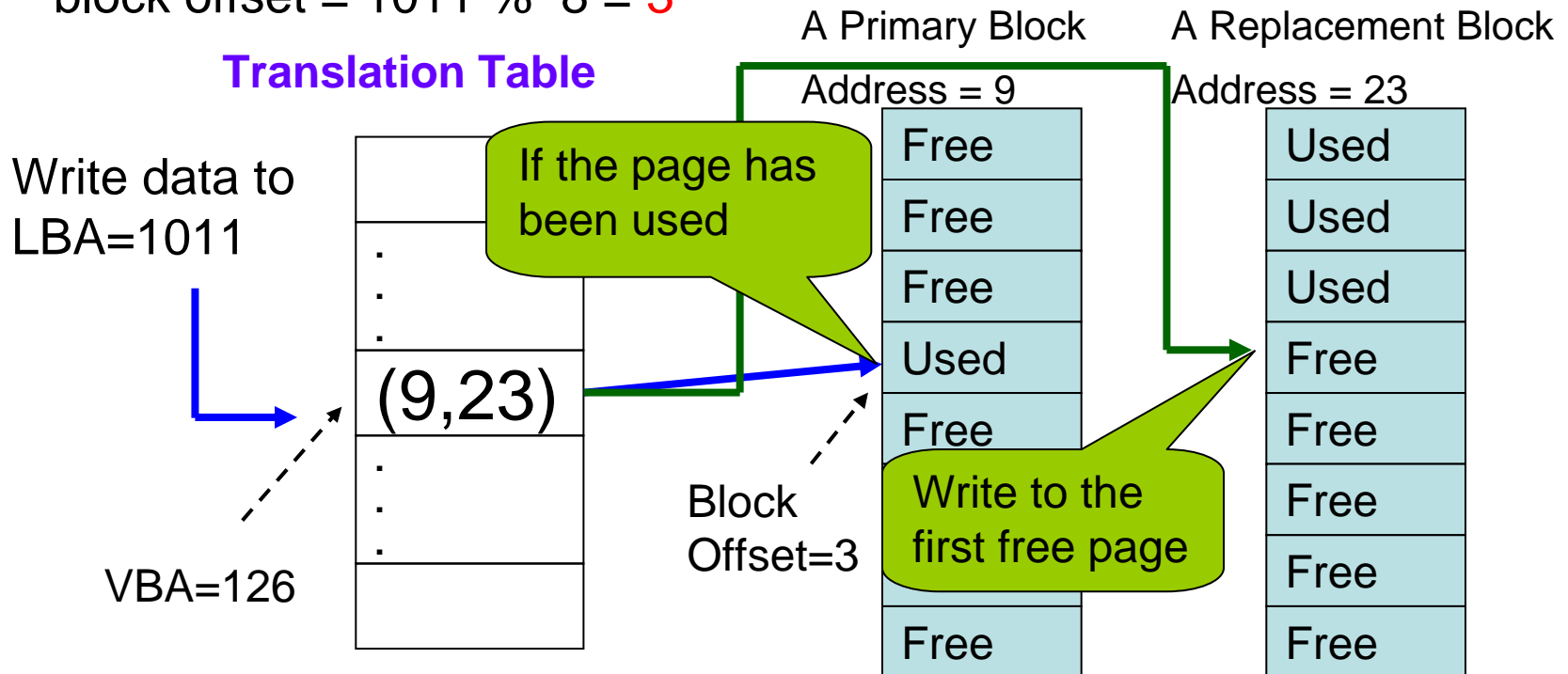
- BL adopts a block-level address translation mechanism.
 - The main problem of BL is on **low space utilization** and **significant write overhead**
 - e.g., LBA=1011 => virtual block address (VBA) = $1011 / 8 = 126$ and block offset = $1011 \% 8 = 3$





NFTL Mechanism

- A logical address under NFTL is divided into a virtual block address and a block offset.
 - e.g., LBA=1011 => virtual block address (VBA) = $1011 / 8 = 126$ and block offset = $1011 \% 8 = 3$





Motivation

- Problems of flash management mechanisms
 - Small mapping granularity: infeasibility, large memory consumption
 - Large mapping granularity: low performance, and many live copies
- Limitations of MLC NAND flash memory
 - Sequential writes in a block
 - Partial programming prohibition
- Characteristics of file systems
 - It is originally designed for hard disks
 - It has many small-size updates on Metadata

臺灣大學



Outline

- Introduction
- File Systems Observations
 - Ext2
 - NTFS
- The Proposed Flash Translation Layer Mechanism
- Experiments
- Conclusion

臺灣大學



File System Observations

- The I/O requests from the file system can be separated into **Metadata** and **Userdata** requests.
- Purpose of Metadata
 - General information of file systems
 - Specific attributes of each file and directory
 - Logging or journaling information
- Metadata occupy a small portion of file system, but they are accessed frequently.

臺灣大學



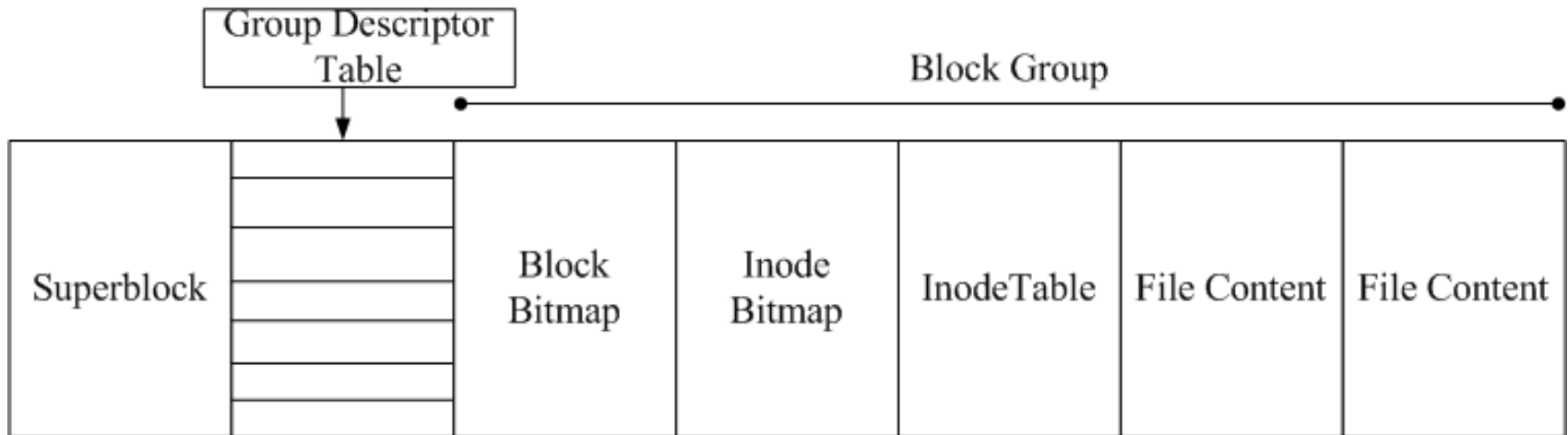
File System Observations – Ext2

- The whole area is divided into several block groups, and a block group contains several blocks.
- The Metadata of Ext2:
 - **Superblock**: Contain general information of file system
 - **Group descriptor table**: Contain information of each block group
 - **Inode bitmap and block bitmap**: Maintain the allocation
 - **Inode table**: Store all the Inodes in the a group



File System Observations – Ext2

- Ext2 layout





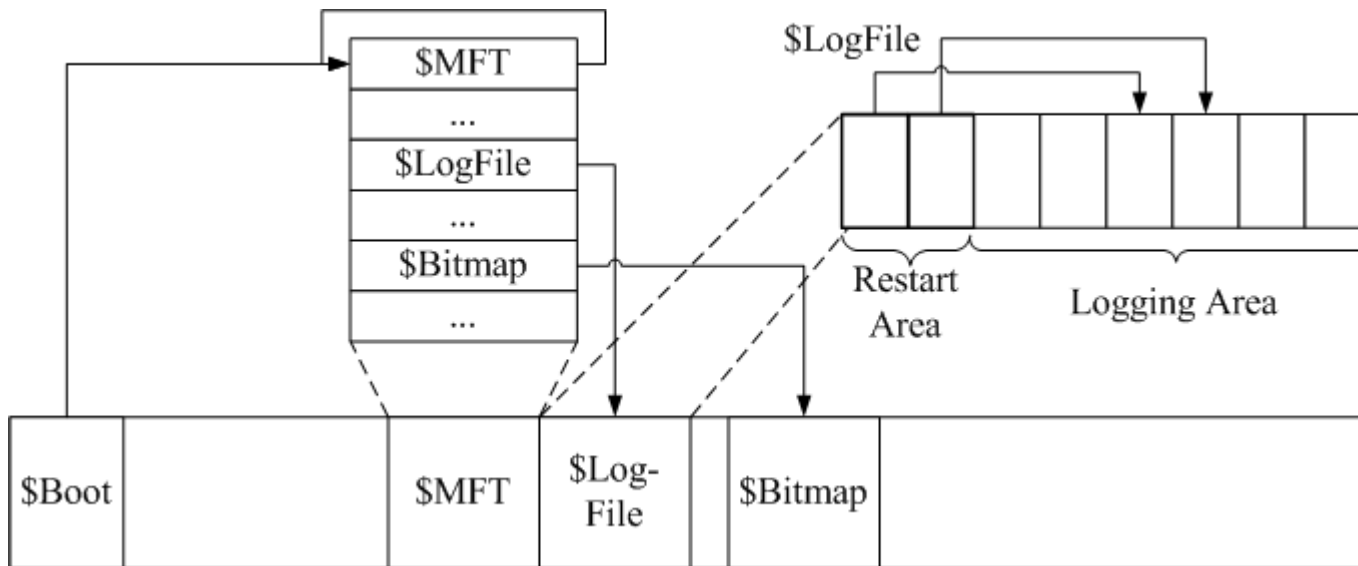
File System Observations – NTFS

- NTFS models everything in the file system as a file, including file system metadata.
- NTFS uses **metadata files**, whose file names start with \$, to store the file system's administrative data.
- \$MFT (**Master File Table**) maintains one entry for each file or directory to store file attributes.
- The Metadata files are allocated at the first 16 MFT entries.



File System Observations – NTFS

- NTFS layout





File System Observations – Traces

Ext2 Trace

Requests (R/W)	Data Type
R	Group Descriptor Table
R	Inode Bitmap
W	Superblock
W	Group Descriptor Table
W	Inode Bitmap
W	Inode Table
R	Block Bitmap
W	File Content
W	Directory
W	Superblock
W	Group Descriptor Table
W	Inode Table
W	Block Bitmap

NTFS Trace

Requests (R/W)	Data Type
R	\$MFT
R	\$Bitmap
W	\$Bitmap
W	\$MFT
W	\$LogFile
W	\$LogFile
W	\$LogFile
W	File Content
W	Index
W	Index
W	\$LogFile
W	\$LogFile
W	\$LogFile
W	\$MFT
W	\$MFT
W	\$LogFile
W	\$LogFile

臺灣大學



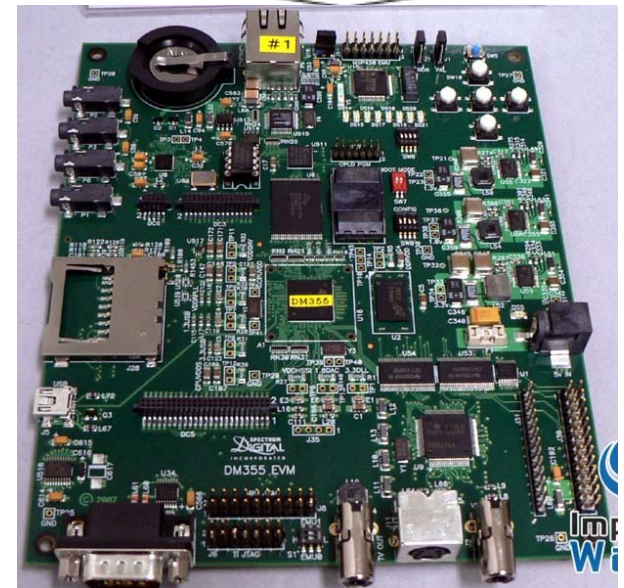
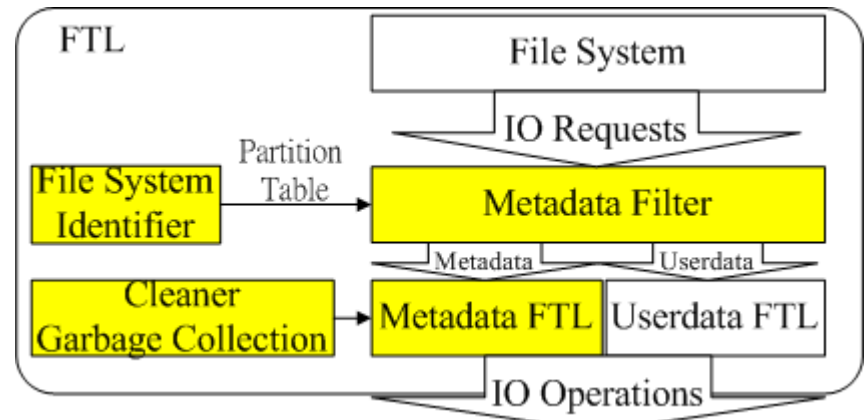
Outline

- Introduction
- File Systems Observations
- The Proposed Flash Translation Layer Mechanism
 - Metadata Filter
 - Metadata FTL
 - Block Set and Summary Page
- Experiments
- Conclusion



The Proposed FTL Mechanism

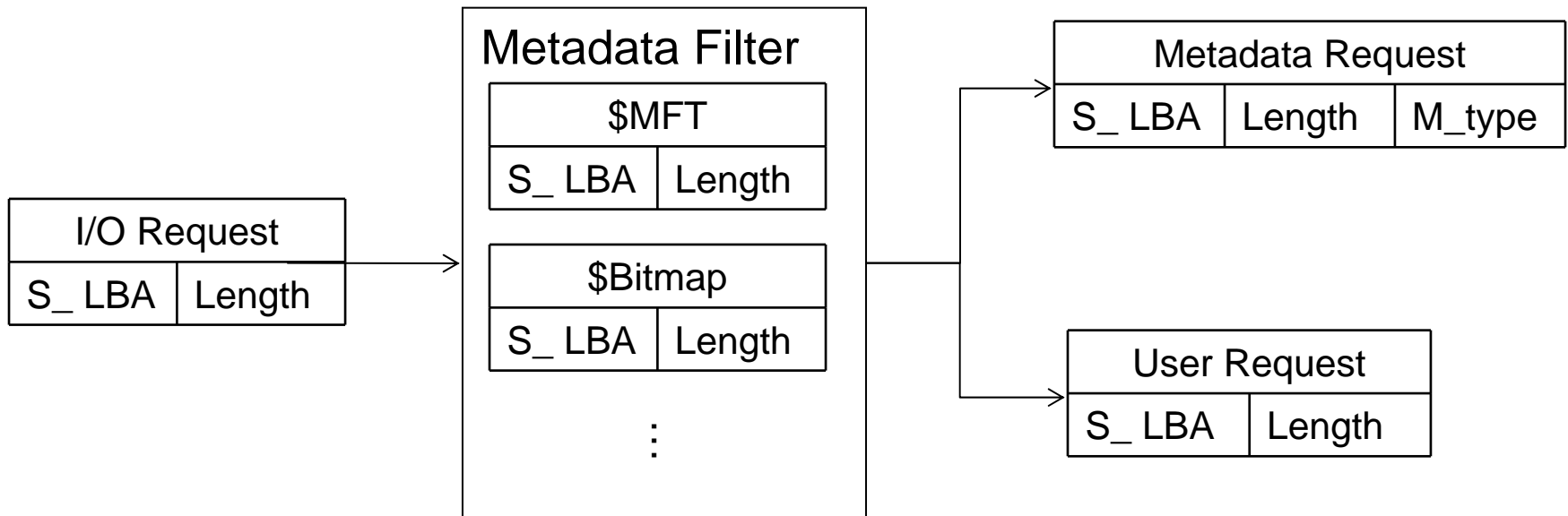
- **Metadata Filter**: Separate Metadata requests and Userdata requests
- **File System Identifier**: Determine the layout of the file systems
- **Metadata FTL**: Manage the Metadata location and translate to IO Operations
- **Cleaner**: Do garbage collection to reclaim invalid pages





Metadata Filter

- Maintain a **run-length** structure for each Metadata or Metadata file.
- According to the file system layout and the address of I/O requests, the Metadata filter dispatches the request to Metadata or Userdata FTL.



臺灣大學



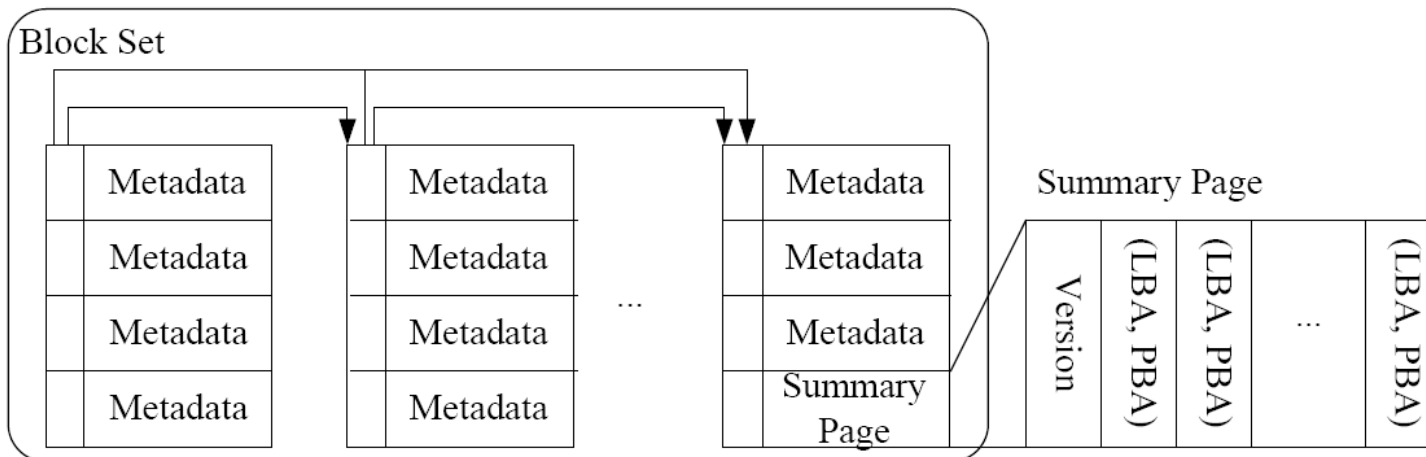
Metadata FTL

- Log-based writing mechanism
 - Cooperate with MLC NAND flash memory
 - Improve file system reliability
- Page-level mapping
 - Reduce address translation overhead
 - Improve file system performance



Block Set and Summary Page

- Block Set
 - Write interleavingly in the block set to exploit parallelism
- Summary page
 - Locate at the last page of the block set
 - Stores version number and mapping information



臺灣大學



System Recovery

- After system crashes, the mapping information in the RAM is lost, and some data may also corrupt.
- Recovery mechanism:
 - Scan the summary page of each block to restore mapping information.
 - If the data is corrupted, use the previous version.

臺灣大學



Cleaner

- Cleaner is triggered when the number of free pages is lower than the threshold.
- Adopt a round-robin garbage collection mechanism
 - The block sets in the head are likely with more invalid pages
 - After live data copying, the summary page must be written to maintain data consistency.

臺灣大學



Outline

- Introduction
- File Systems Observations
- The Proposed Flash Translation Layer Mechanism
- **Experiments**
- Conclusion

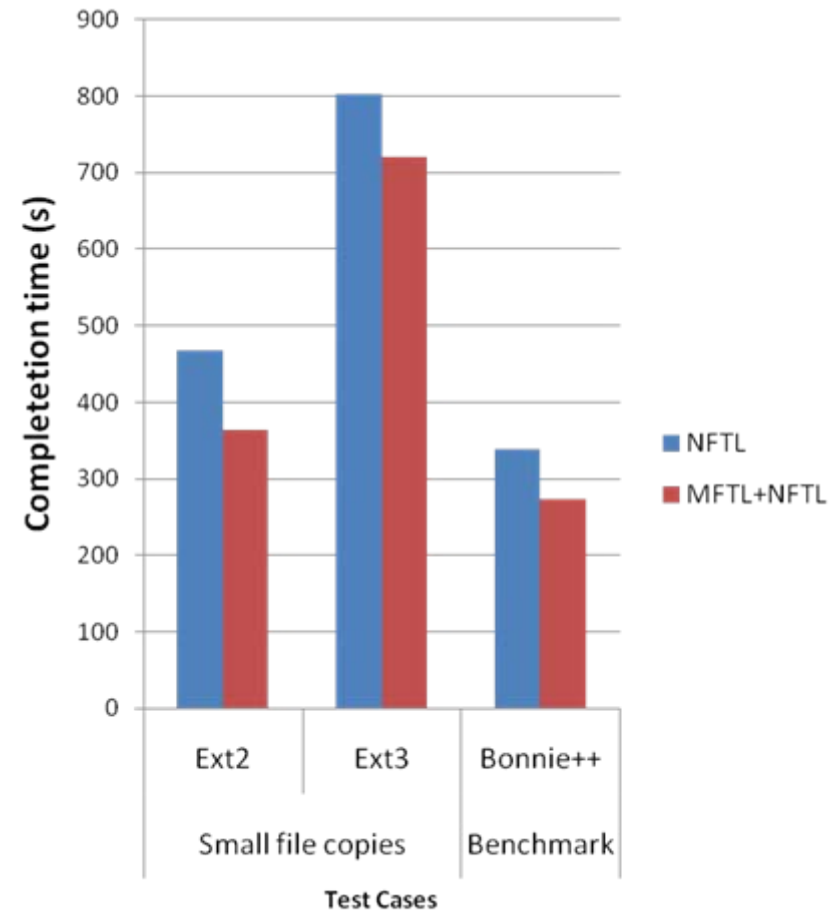
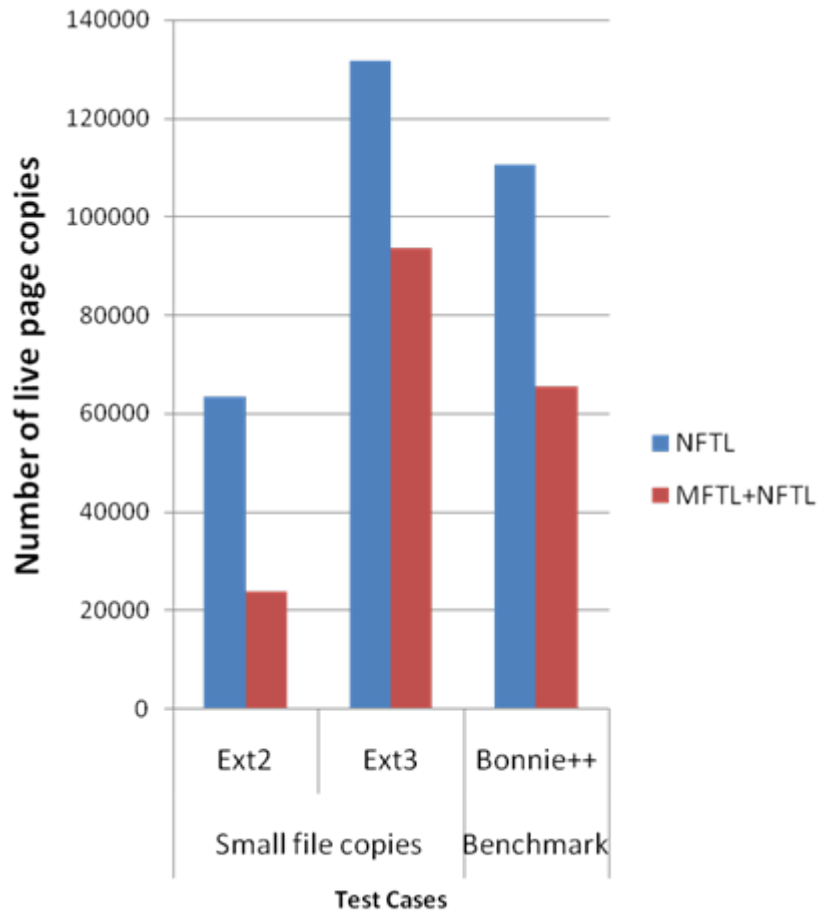


Experiments

- Linux file system
 - Experiments on a DaVinci Evaluation Board with realistic transmission patterns and benchmark.
 - Bonnie++: Sequential/Random files creation and deletion
 - The on-board NAND flash memory is 64 MB with small block SLC flash chip.
- Windows file system
 - Simulation over a 7-day trace with a 20GB hard disk.
 - The trace of daily activities, web surfing, email access, movie playing, etc.
 - Large block SLC and MLC flash chips.

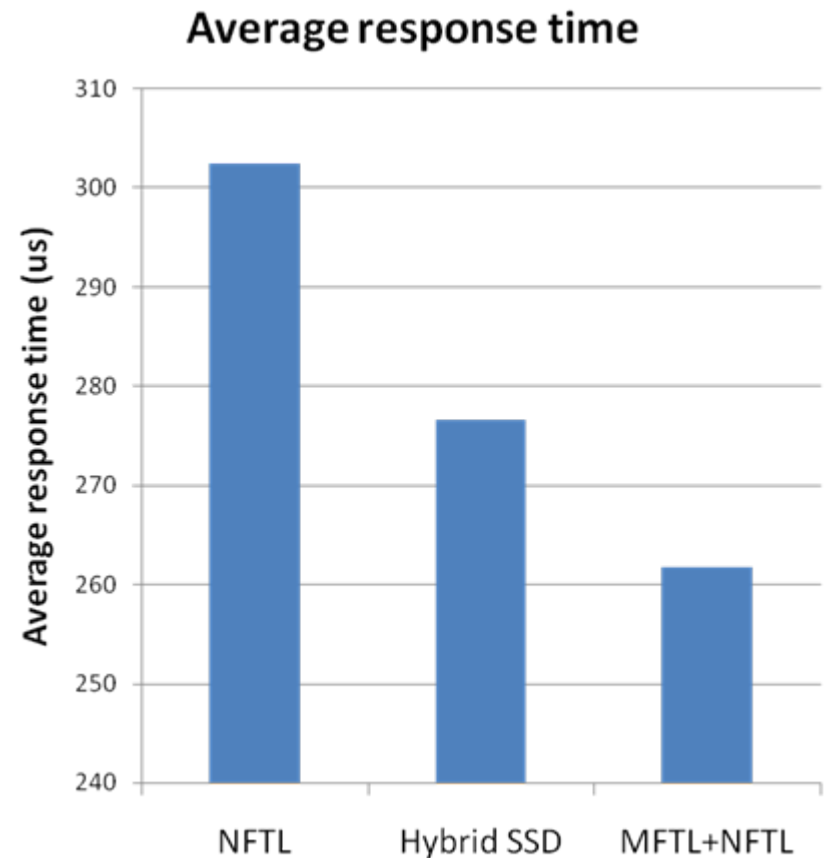
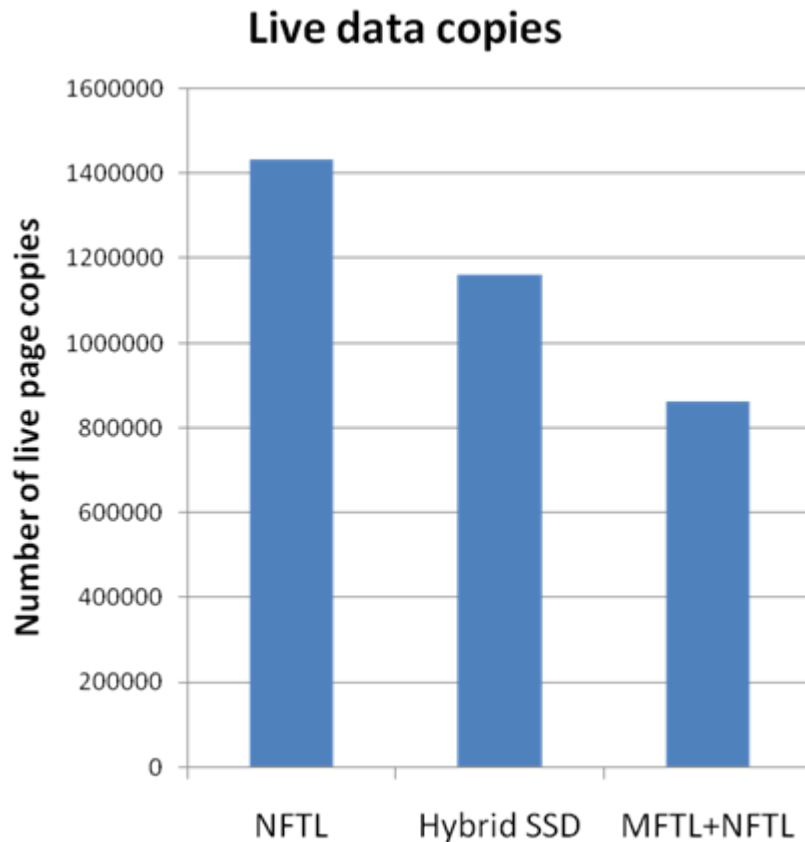


Experiments Results – Linux File System





Experiments Results – Windows File System





Performance Improvement

	Ext2		Ext3	NTFS
Improvement	Bonnie++	File-copy	File-copy	Traces
FTL	0%	0%	0%	0%
NFTL	19%	22%	10%	14%
BL	37%	49%	24%	13%

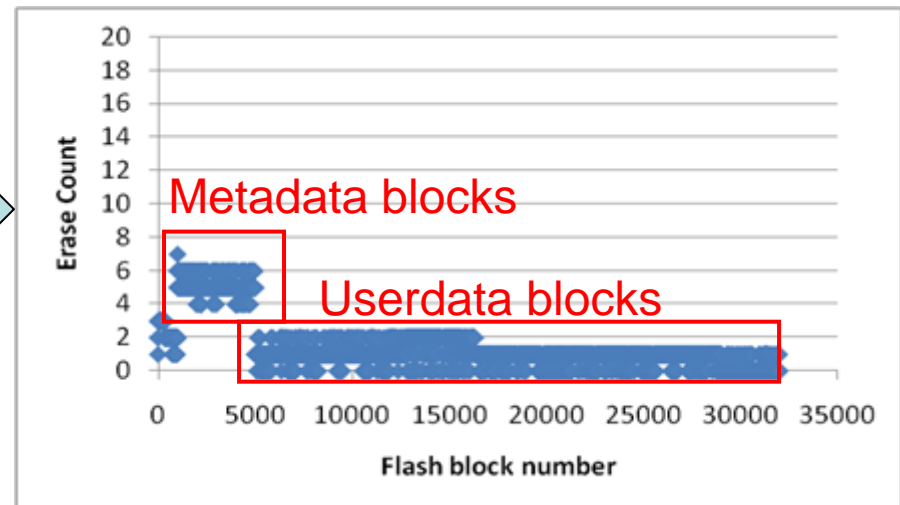
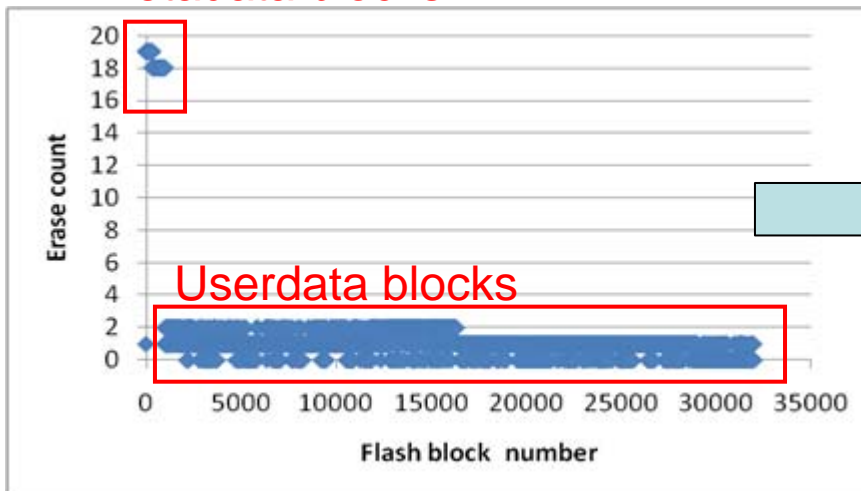
- RAM consumption
- 20 GB storage, 2KB per page, 64 pages per block
 - NTFS: 185KB for NTFS Metadata files
 - Ext2, Ext3: 30KB, for block bitmap and inode bitmap in each group




Wear-leveling

- The blocks for Metadata with higher erase counts.
- If the ratio of the erase counts of Metadata blocks to those of the Userdata blocks is higher than the threshold, new blocks for Metadata are allocated.

Metadata blocks



臺灣大學



Conclusion

- We propose a Metadata filter to separate Metadata and Userdata requests.
- Manage Metadata in a **log-based** fashion with a **fine-grained address translation** mechanism to improve the system performance and reliability.
- The performance of ext2/ext3 and NTFS file systems could be improved by more than 20% and 10%, respectively.

臺灣大學



Future Work

- Explore the trade-off between the main-memory consumption and performance improvement
- Exploit the adoption of multi-channel designs to improve the performance of multi-chip flash-memory storage devices