Advanced Flash Translation Layer

Jihong Kim
Dept. of CSE, SNU

Outline

- Problems of Hybrid-Mapping-Based FTL
- FTLs for Memory-Constrained Storage Systems
  - DFTL
  - μ-FTL

Hybrid FTL Schemes

- The main difficulties the FTL faces in giving high performance is the severely constrained size of SRAM
  - Coarse-grained mapping (block-level mapping)
    - Small SRAM size / Poor garbage collection efficiency
  - Fine-grained mapping (page-level mapping)
    - Efficient garbage collection / Large SRAM size

Problems of Hybrid FTL Schemes

- Fail to offer good performance for enterprise-scale workloads
- Require workload-specific tunable parameters
- Not properly exploit the temporal locality in accesses

Basic Approaches to Memory-Constrained Storage Systems

- Cached mapping information

- On-demand loading of mapping information
  - DFTL
- Better data structures for mapping information
  - μ-FTL

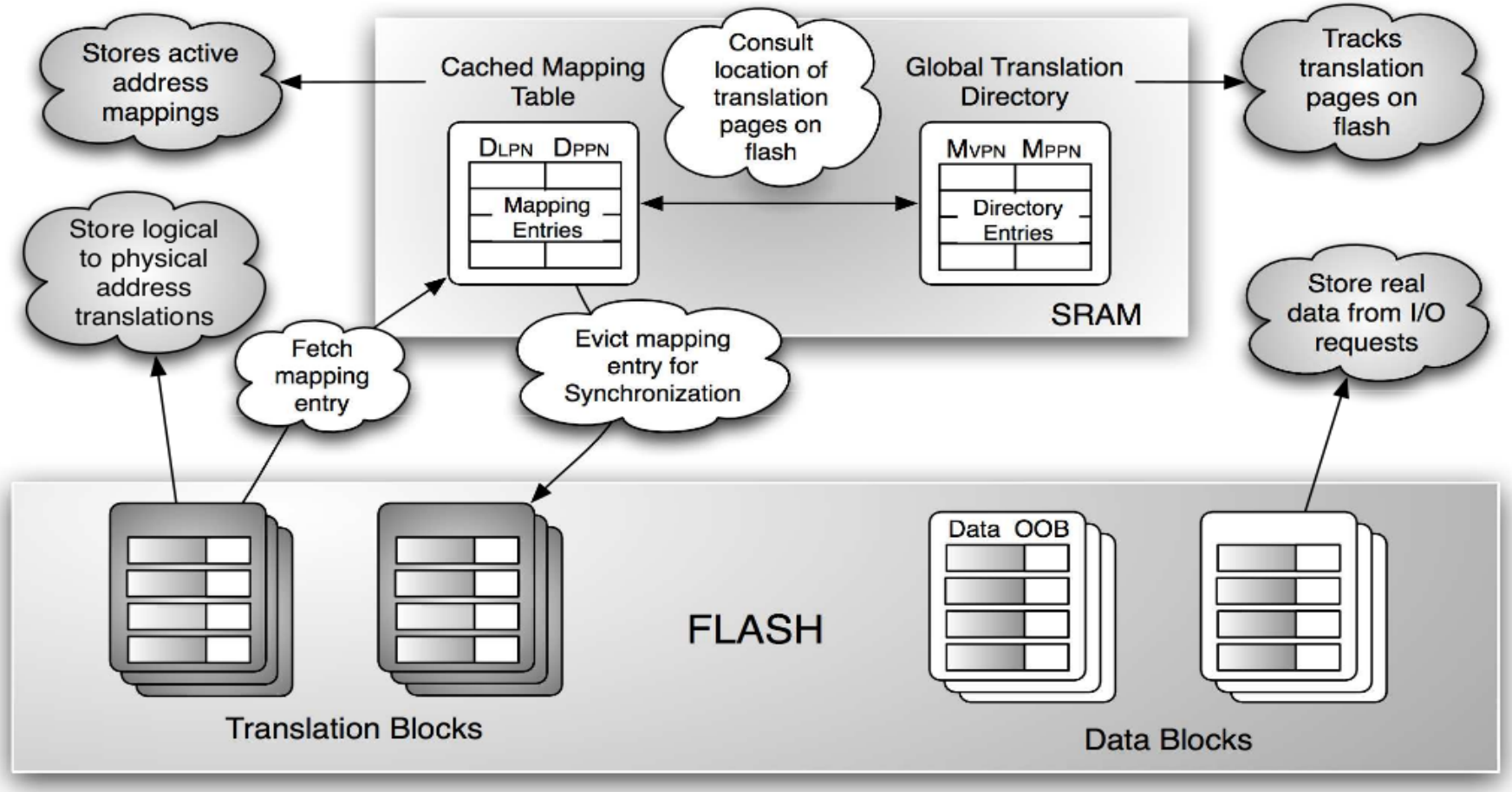
DFTL

- Hybrid FTLs suffer performance degradation due to full merges
  - Caused by the difference in mapping granularity of data and log blocks
  - A high performance FTL must be re-designed without log-blocks
- DFTL is an enhanced form of the page-level FTL scheme
  - Allow requests to be serviced from any physical page on flash
  - All blocks can be used for servicing update requests
- How to make the fine-grained mapping scheme feasible with the constrained SRAM size
  - Use an on-demand address translation mechanism

Demand-based Selective Caching of Page-level Address Mapping

- Propose a novel FTL scheme (DFTL) : Purely page-mapped FTL
  - Exploit temporal locality of accesses
  - Uses the limited SRAM to store the most popular mappings while the rest are maintained on flash
  - Provide an easier-to-implement solution
  - Devoid of tunable parameters

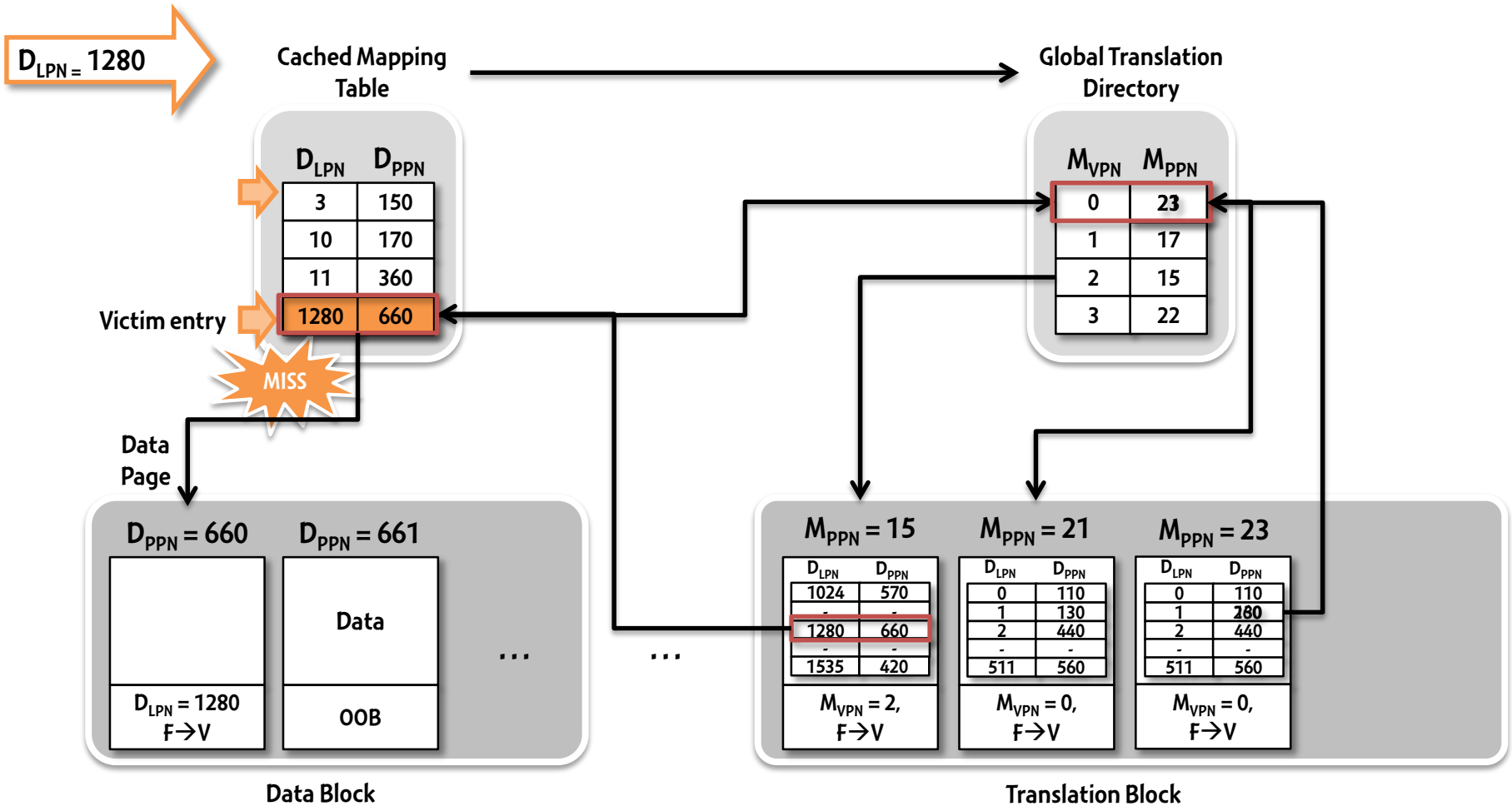
DFTL Architecture



Data Blocks and Translation Blocks

- DFTL partitions all blocks into two groups
  - Data blocks: composed of data pages
    - Each data page contains the real data
  - Translation blocks: consists of translation pages
    - Each translation page stores information about logical-to-physical mappings
    - Logically consecutive mappings information stored on a single page
    - 512 logically consecutive mappings in a single page (page size: 2 KB, addr: 4 Byte)

Example:
When a Request Incurs a CMT miss



Overhead in DFTL Address Translation

- The worst-case overhead in DFTL address translation
  - Two translation page reads
    - One for the victim by the replacement policy
    - The other for the original requests
  - One translation page write
    - For the translation page write for the victim
- The address translation overhead can be mitigated
  - The existence of temporal locality helps in reducing the # of evictions
  - Batch updates for the pages co-located in the victim could also reduce the # of evictions

Read/Write Operation

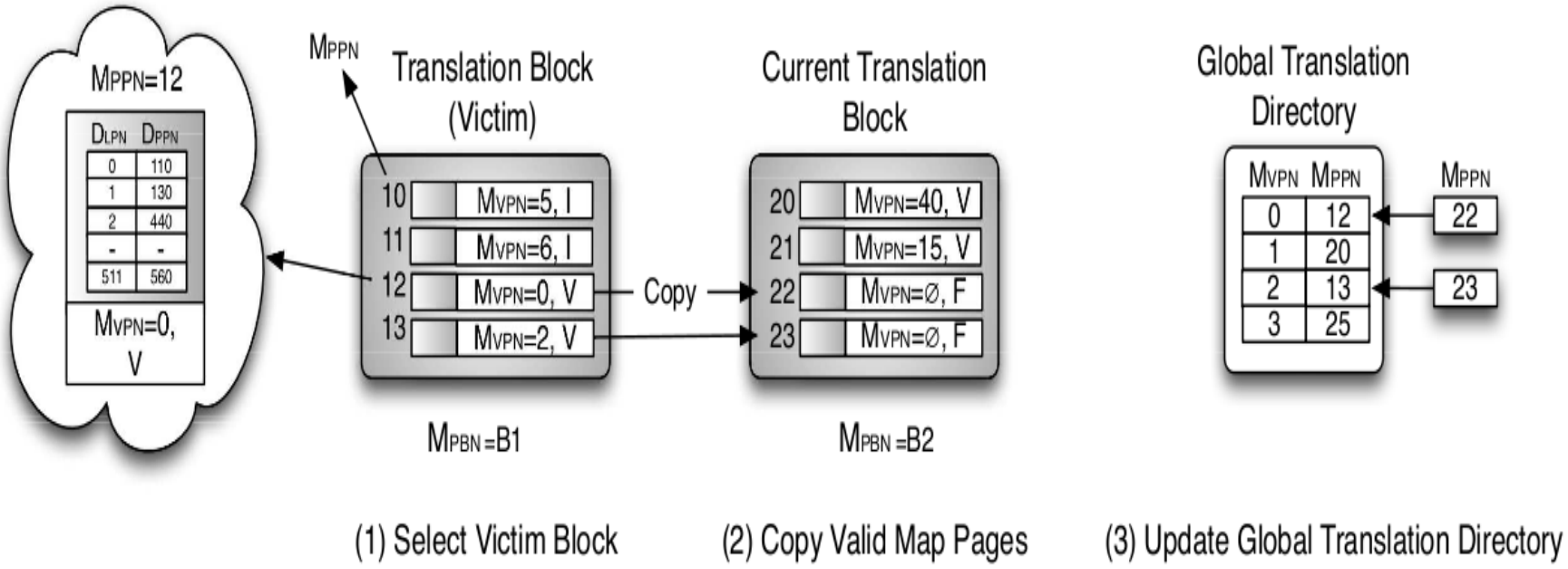
- For a read operation
  - Directly serviced through flash page read operation once the address translation is completed
- For a write operation
  - Maintain two types of blocks for data block and translation blocks
    - *Current data block* and *current translation block*
  - Sequentially writes the given data into these blocks

Garbage Collection

- Different steps are followed depending on the type of a victim block
  - Translation block:
    - Copy the valid pages to the current translation block
    - Update the corresponding GTD
  - Data block:
    - Copy the valid pages to the current data block
    - Update all translation pages and CMT entries associated with these pages

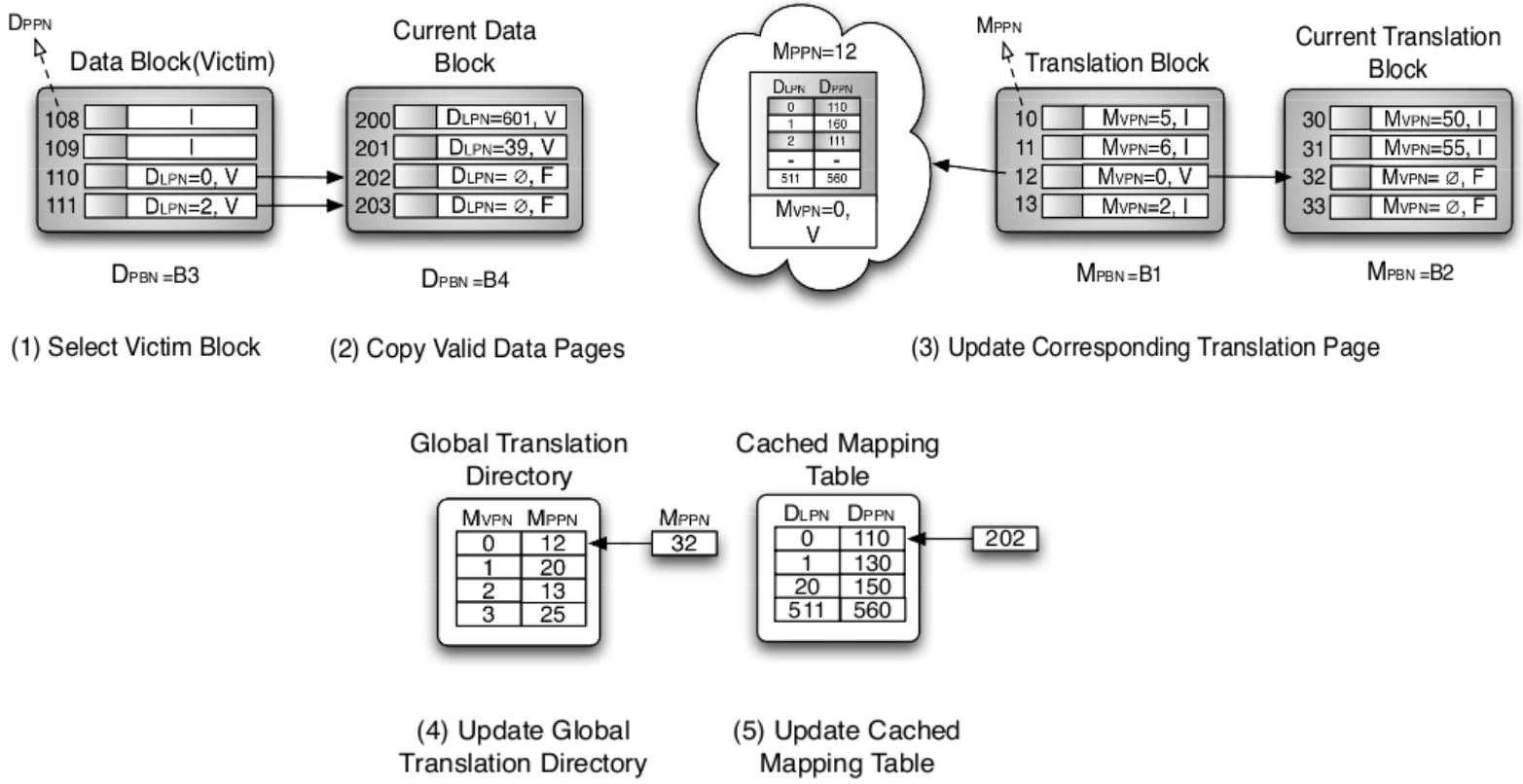
Example: Translation Block

- Translation block as victim for garbage collection



Example: Data Block

- Data block as victim for garbage collection



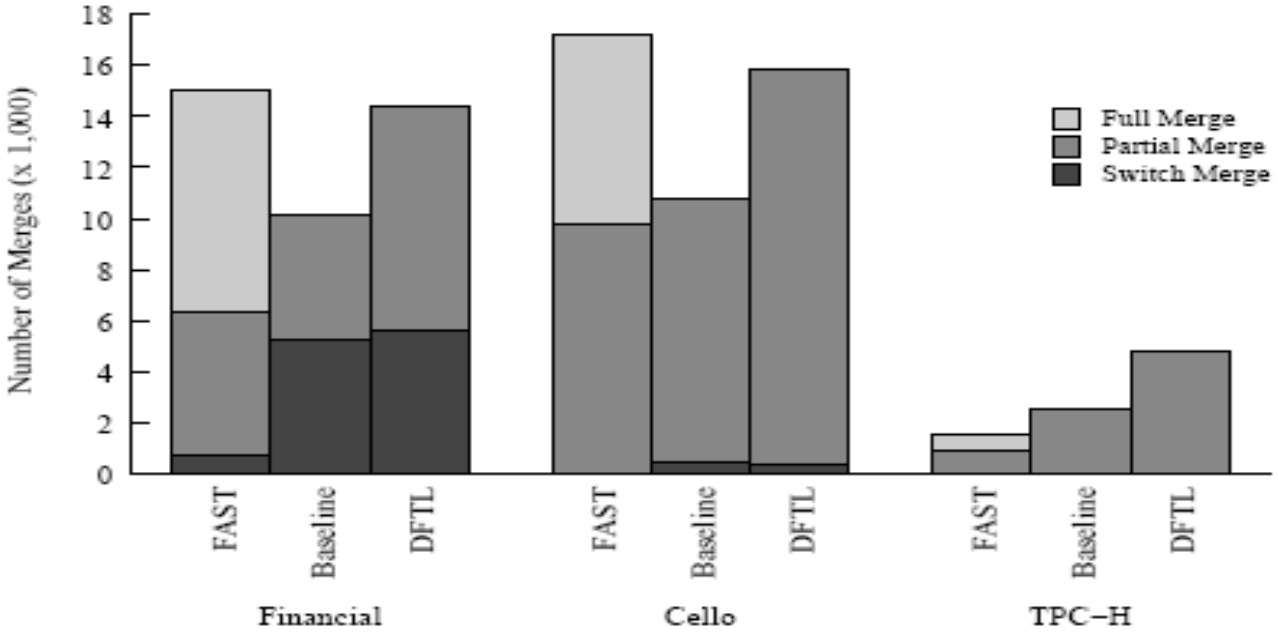
Evaluation Setup

- Parameters
  - Flash memory size: 32 GB / SRAM size: 2 MB
  - Log buffer size: 512 MB (about 3% of the total flash capacity)
  - Evaluated schemes: FAST, baseline, DFTL
- Workloads

Workloads	Avg. Req. Size (KB)	Read (%)	Seq. (%)	Avg. Req. Inter-arrival Time (ms)
Financial [25]	4.38	9.0	2.0	133.50
Cello99 [10]	5.03	35.0	1.0	41.01
TPC-H [28]	12.82	95.0	18.0	155.56
Web Search [26]	14.86	99.0	14.0	9.97

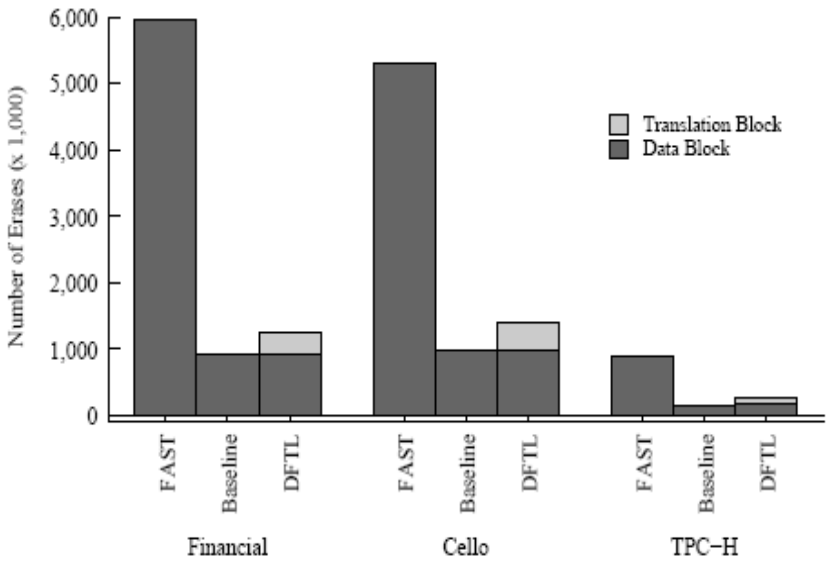
- Performance metrics
  - Garbage collection's efficacy
  - Response time (device service time + queuing delay)

The Number of Block Merges

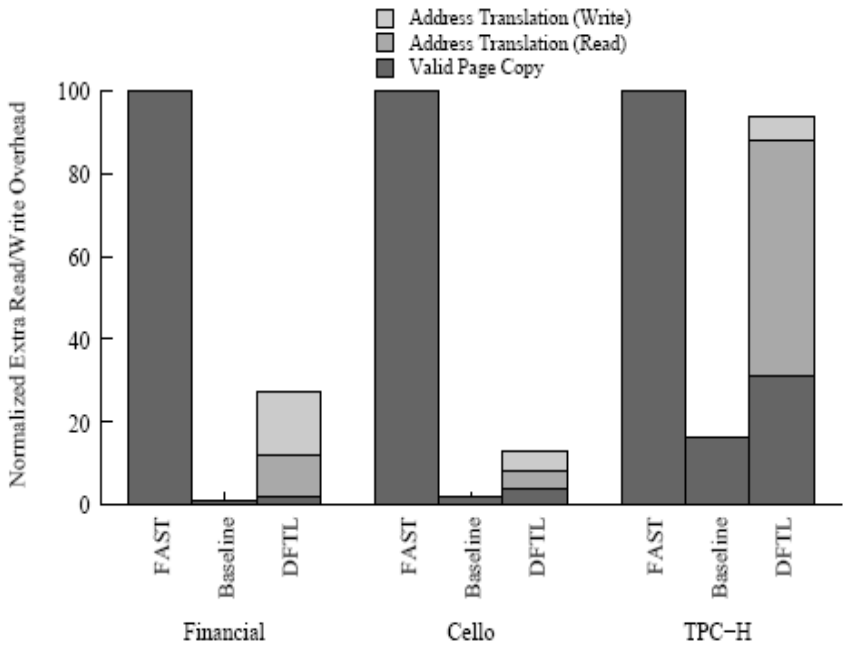


- Baseline and DFTL show a higher number of switch merges
- FAST incurs lots of full merges
  - 20% and 60% of full merges involve more than 20 data blocks in Financial and TPC-H benchmarks, respectively

Address Translation Overhead



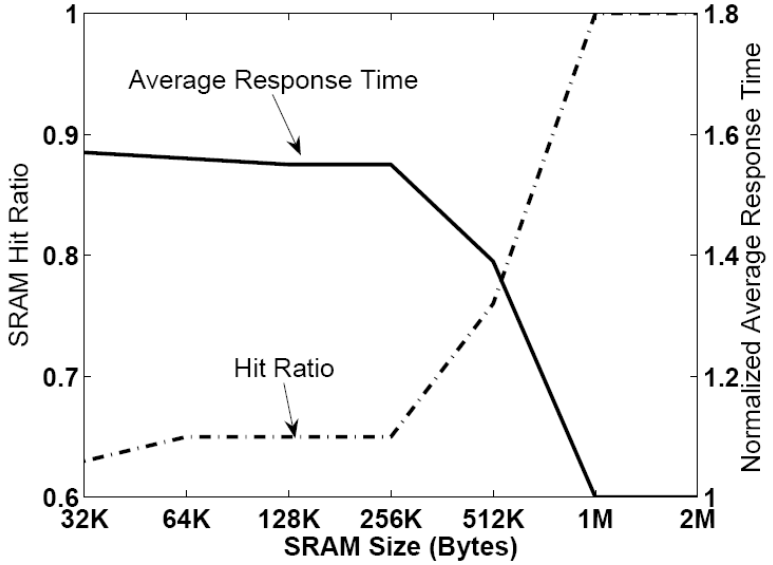
Block Erases



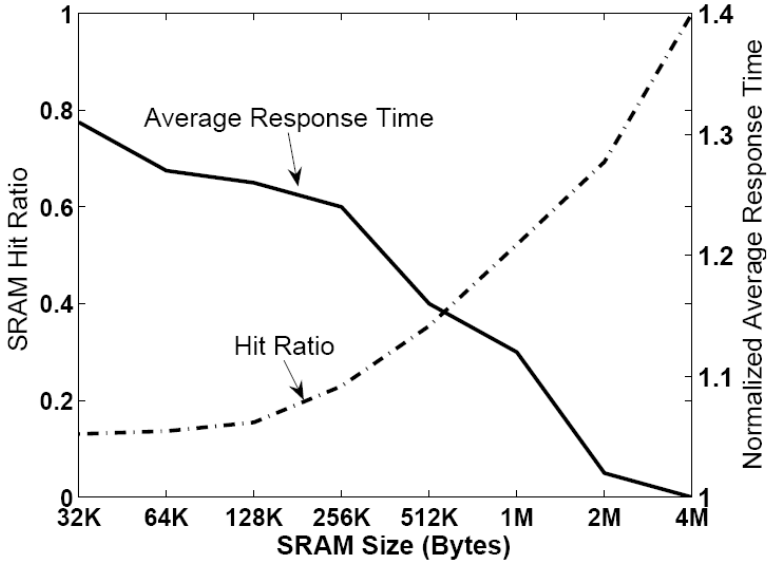
Extra Read/Write Ops.

- DFTL incurs extra overheads due to its translation mechanism
  - The address translation accounts for 90% of the extra overhead
- DFTL yields a 3-fold reduction in extra ops. over FAST
  - 63% hits for address translations in SRAM

Impact of SRAM size



(a) Financial Trace



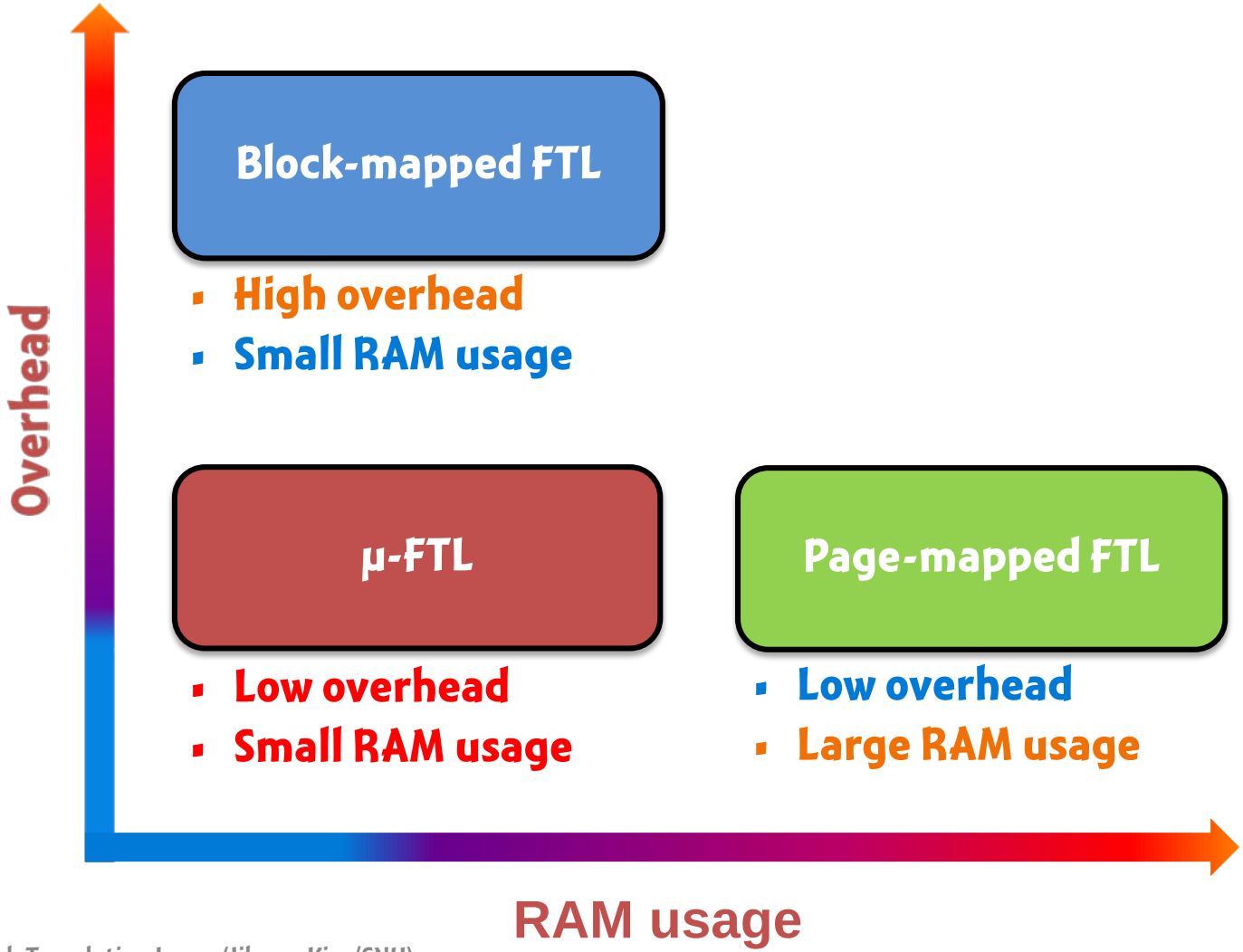
(b) TPC-H Benchmark

- With the SRAM size approaching the working set size
  - DFTL's performance becomes comparable to Baseline (=page level FTL)

μ-FTL

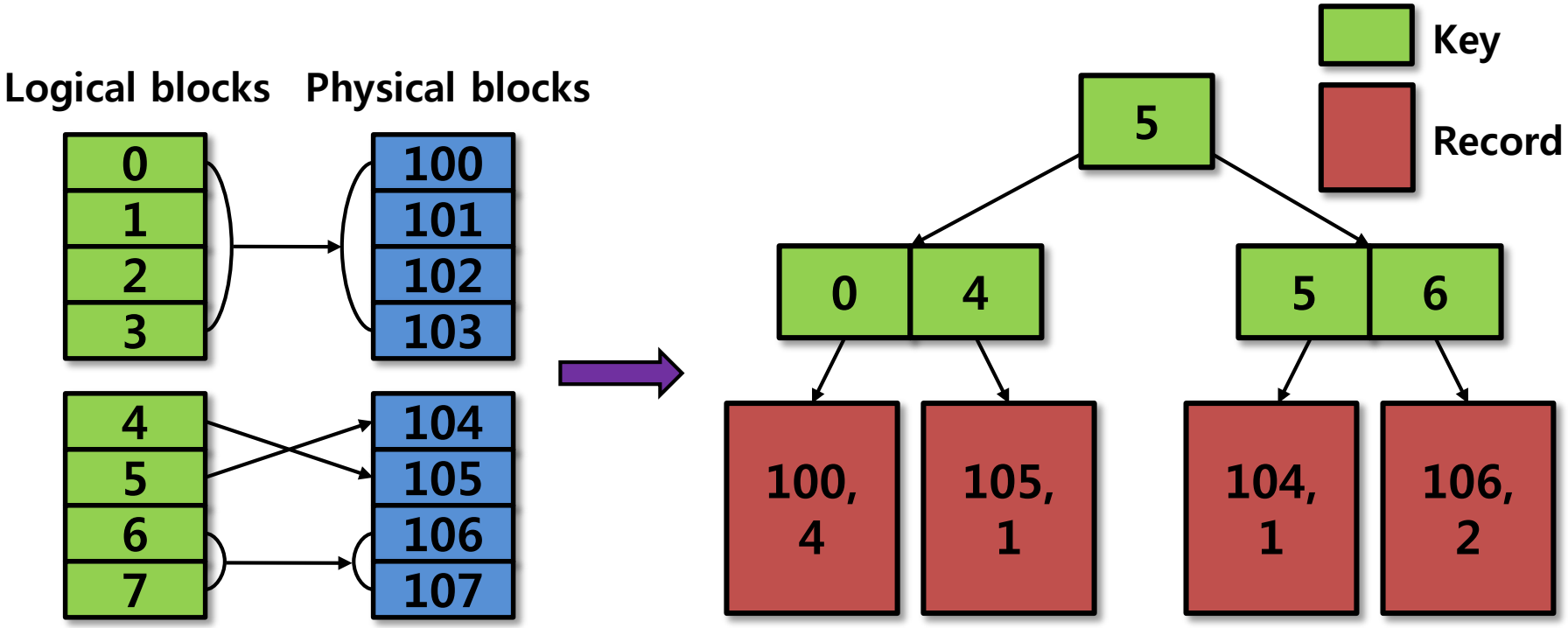
- Design goal
  - Reduce the RAM usage as small as block-mapped FTLs
  - Providing the performance comparable to page-mapped FTLs
- Key observations: Two dominant workloads
  - Coarse-grained mapping for large and sequential writes
  - Fine-grained mapping for small and random writes
  - Adjusts mapping granularities according to the size of incoming write requests

μ-FTL


Overhead
Block-mapped FTL
High overhead
Small RAM usage
μ-FTL
Low overhead
Small RAM usage
Page-mapped FTL
Low overhead
Large RAM usage
RAM usage

Address Mapping

- µ-FTL implements multiple mapping granularities
  - Extent: A variable-sized mapping entry (a pair of key and record)



µ-FTL mapping example

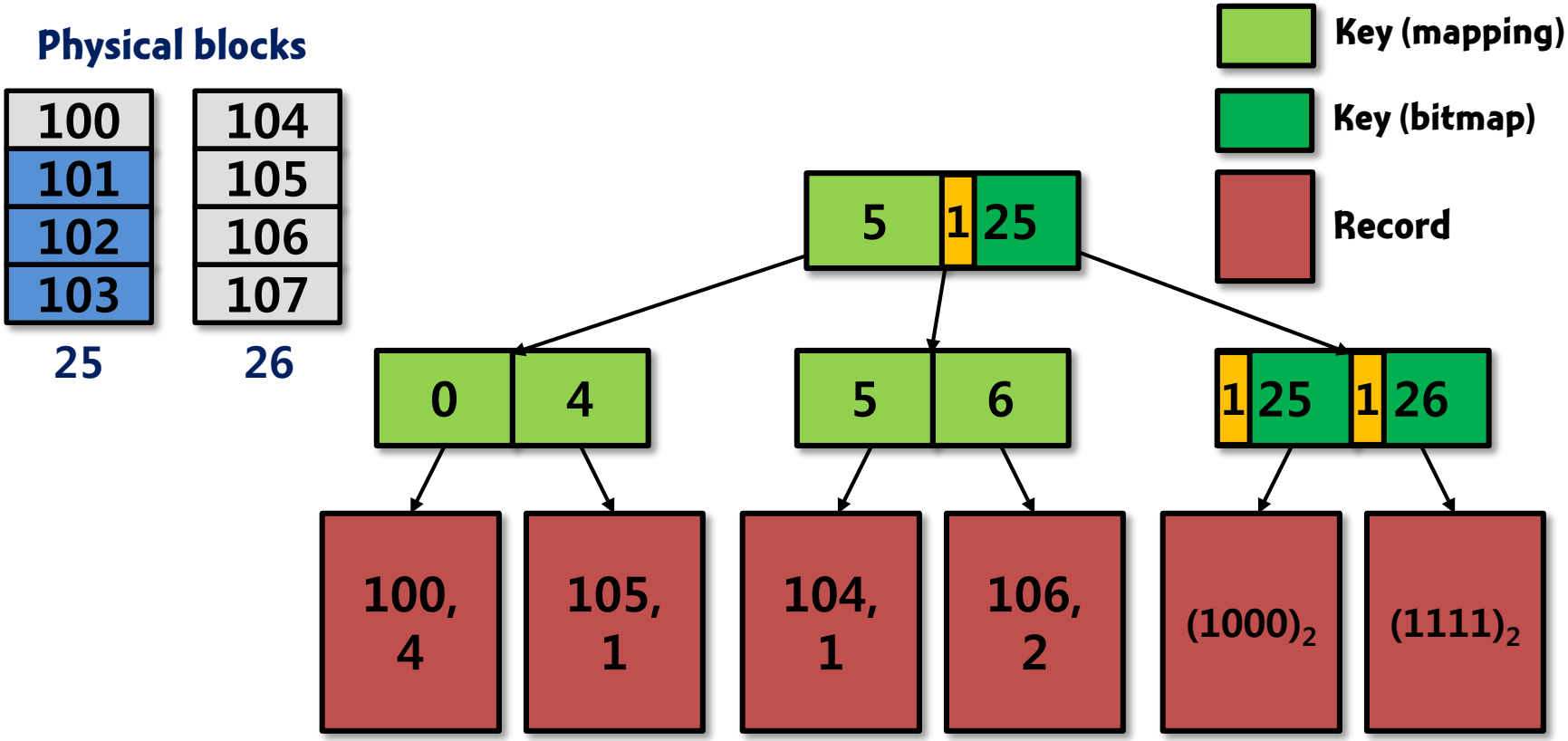
The implementation of mapping example

Garbage Collection

- Victim selection policy
  - Having the largest # of invalid pages
- Garbage collection information
  - Per-block invalid page counter
    - Maintaining the number of invalid pages in each physical block
  - Bitmap information
    - Distinguishing valid pages from invalid pages

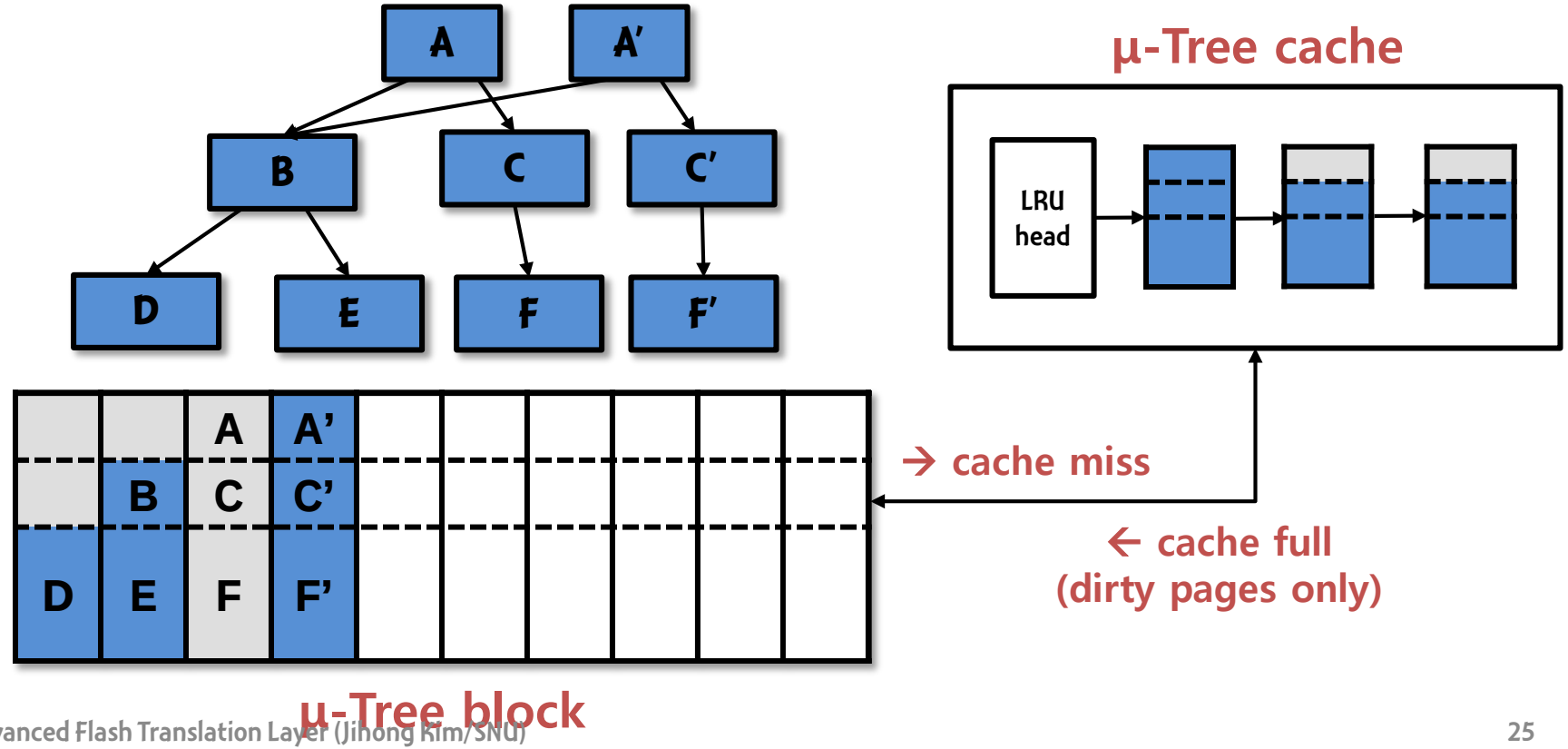
Bitmap Information

- Bitmap information is too large to be stored in RAM
- We store the bitmap information in μ-tree



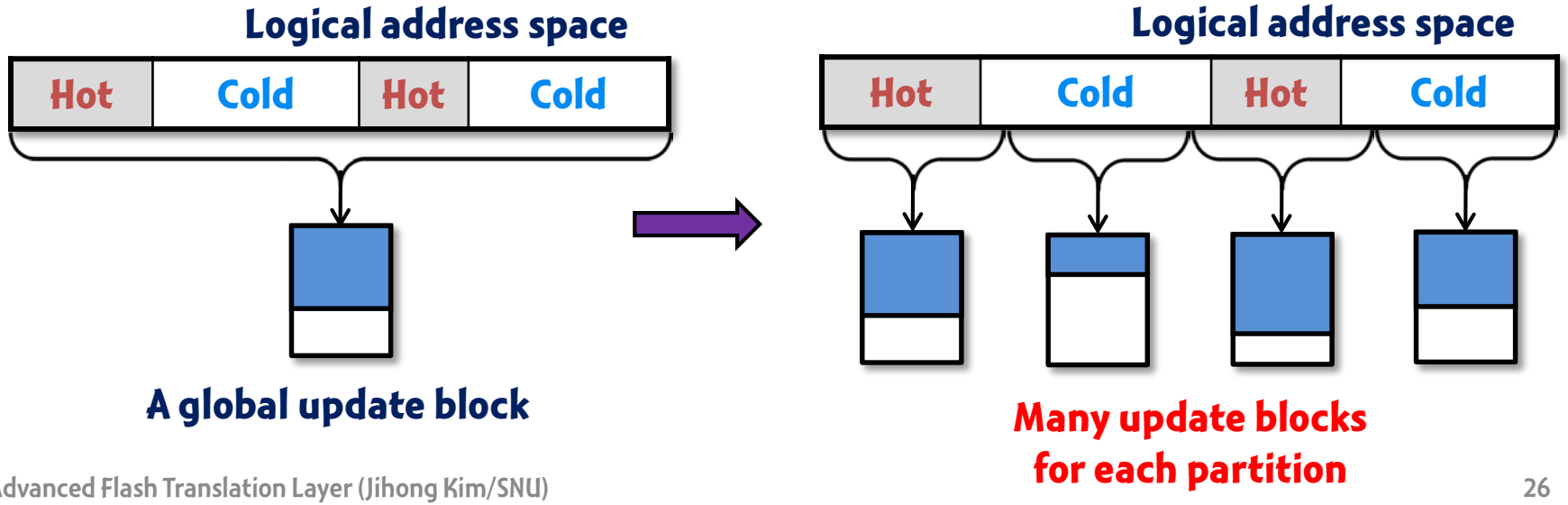
Key Data Structure: μ-Tree

- Handling any insertion/deletion/update operation of the tree with only one flash write operation
  - By storing multiple nodes from the leaf to the root in a single page

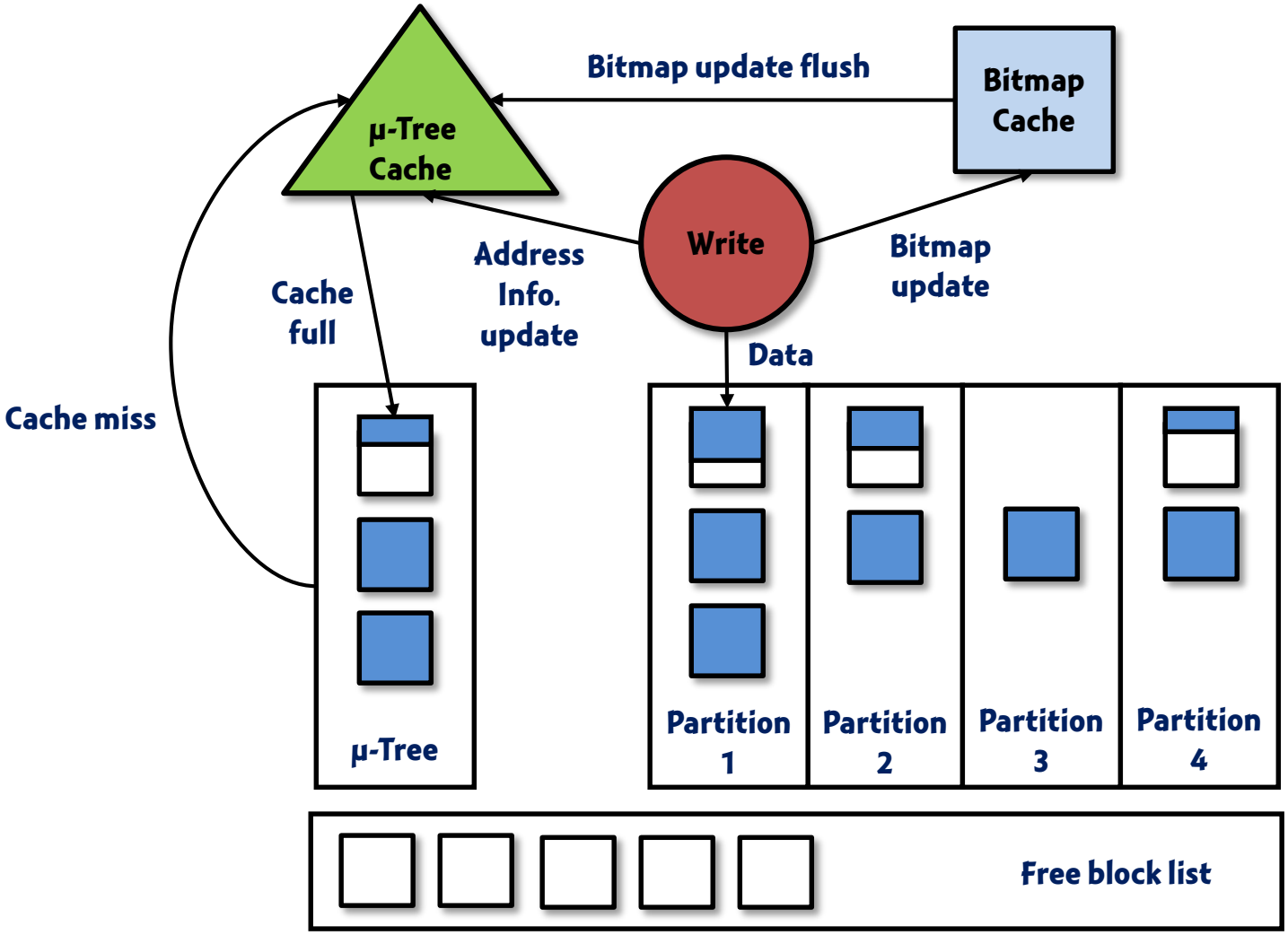


Logical Address Space Partitioning

- Separating hot data from cold data has a beneficial effect on the performance of an FTL
- Each part of the logical address space exhibits different degree of "hotness"
- Update block
  - Being charged for receiving data from incoming write requests



μ-FTL Design Summary



Evaluation Methodology

- Trace-driven simulators for several FTLs
  - Block-mapped: the log block, FAST, the super block
  - Page-mapped: DAC
- 6 traces
  - 3 multimedia traces: PIC(8GB), MP3(8GB), MOV(8GB)
  - 3 PC traces: WEB(32GB), GENERAL(32GB), SYSMARK(40GB)
- The standard configurations

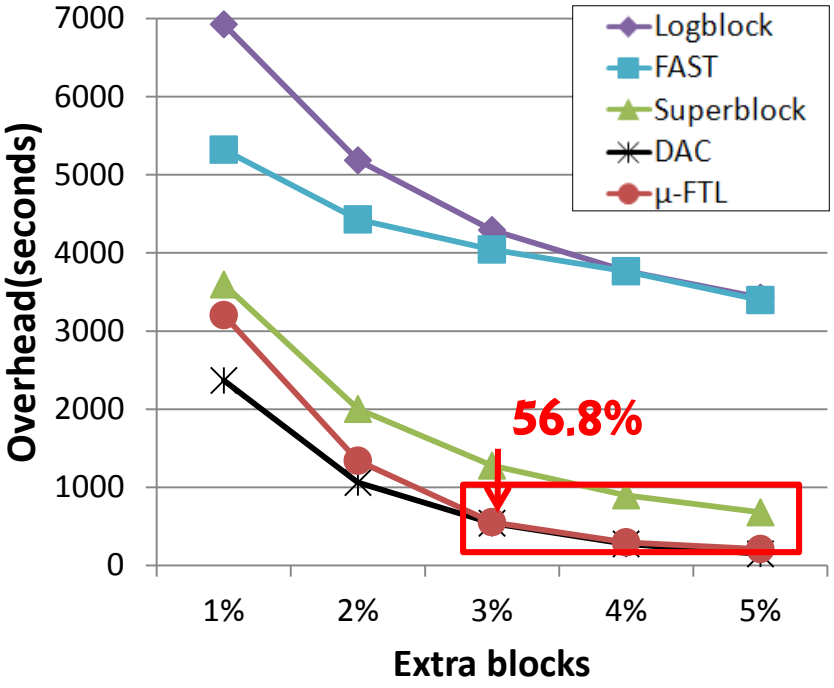
Partition Size	Bitmap cache size	The portion of extra blocks
256MB (512 logical blocks)	4KB for multimedia traces 32KB for PC traces	3%

The RAM Usage

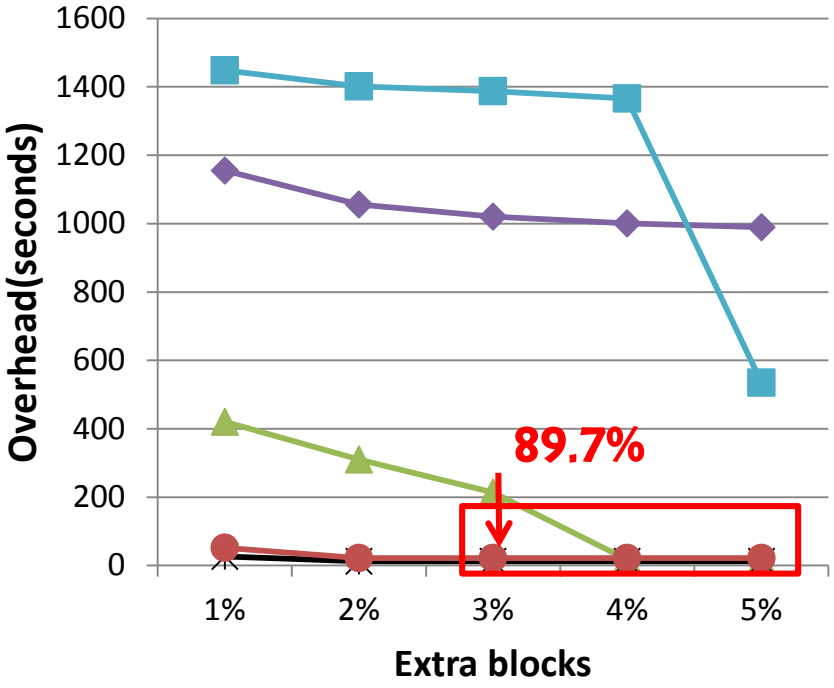
	Block-mapped FTL	μ-FTL (A+B+C)	Page-mapped FTL
8GB	64KB	16KB+44KB+4KB = 64KB	8MB
32GB	256KB	64KB+160KB+32KB = 256KB	32MB
40GB	320KB	80KB+208KB+32KB = 320KB	40MB

(A) Per-block invalid page counter
(B) The μ-Tree cache size
(C) The bitmap cache size

The Comparison of FTL Performance

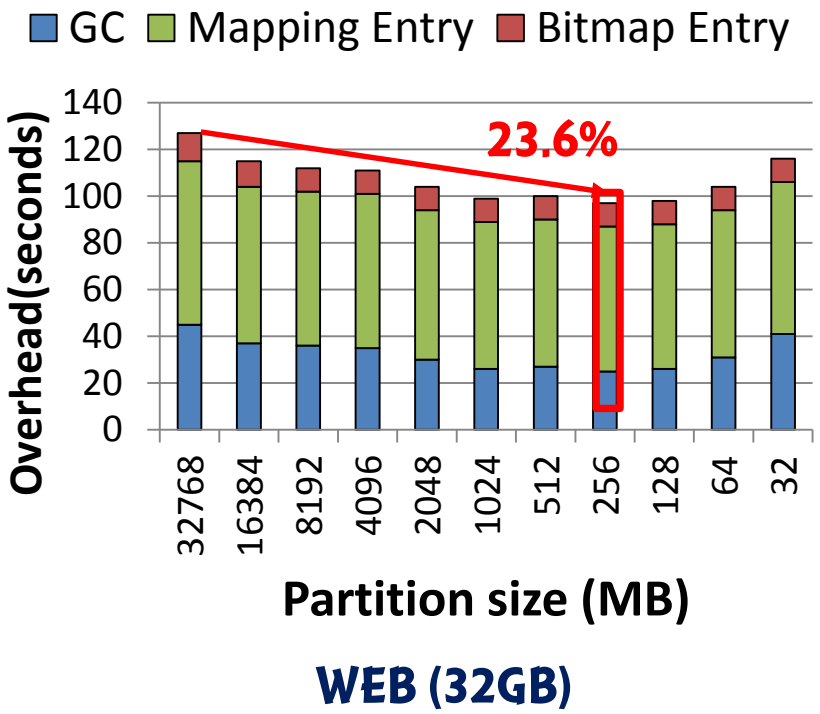


GENERAL (32GB)

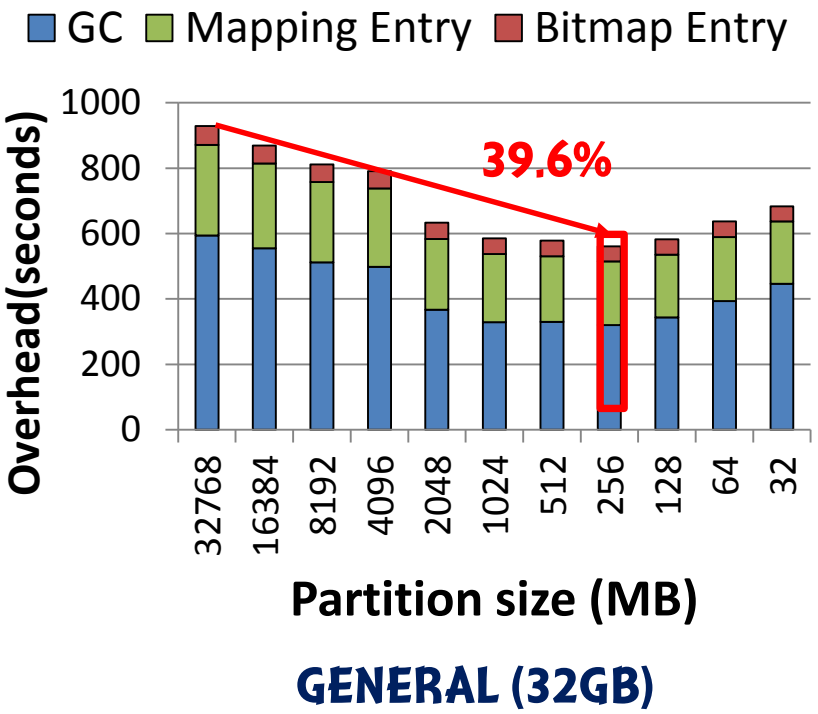


SYSMARK (40GB)

The Effect of Partitioning



WEB (32GB)



GENERAL (32GB)

Reference

- Aayush Gupta et al., "DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings", ASPLOS, 2009
- Yong-Goo Lee et al., "μ-FTL: A Memory-Efficient Flash Translation Layer Supporting Multiple Mapping Granularities," EMSOFT, 2008
- Dongwon Kang et al, "μ-Tree : An Ordered Index Structure for NAND Flash Memory," EMSOFT, 2007