Optimizing Translation Information Management in NAND Flash Memory Storage Systems

Qi Zhang[1], Xuandong Li[1], Linzhang Wang[1], Tian Zhang[1]

Yi Wang[2] and Zili Shao[2]

[1] State Key Laboratory for Novel Software Technology

Nanjing University

[2] The Hong Kong Polytechnic University

Outlines

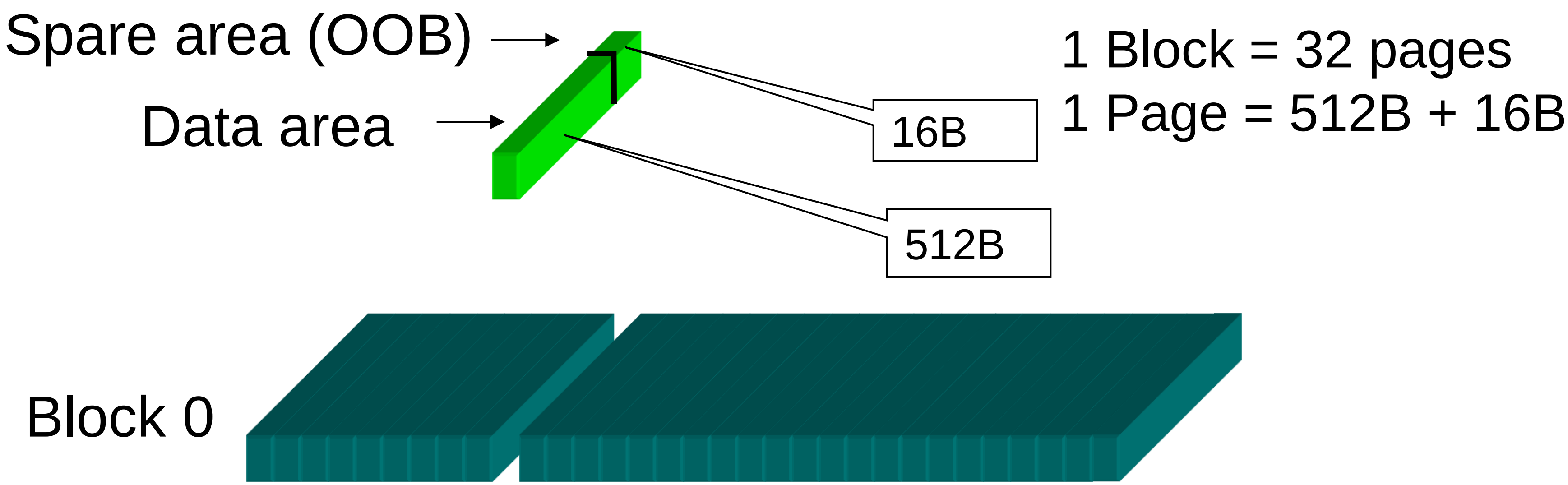
- Introduction
- Motivation
- Our scheme: Translation Information Management
  - Caching Mechanism
  - Multiple Write Pointers Strategy
- Evaluation
- Conclusion

Flash Memory Properties

- Non-volatile memory
- Faster access performance
- Lower power consumption
- Smaller size
- Lighter weight
- Shock resistance

Flash Memory Organization

- **Chip → Block → Page**



- **Block:** basic unit for erase
- **Page:** basic unit for read/write

SAMSUNG 128MB SLC NAND flash memory chip (large block):

Operations	Time
Block erase	**2000 us**
Page read	25 us
Page write	200 us

Constraints

- **Out-of-place update**
  - **A page cannot be overwritten until the block with this page is erased**

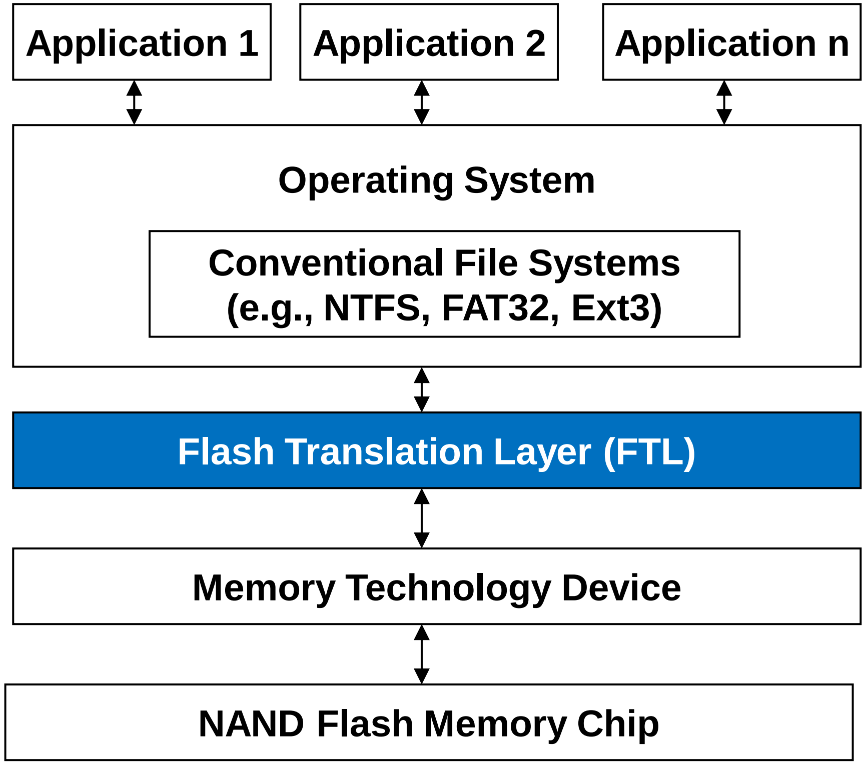
Needs an address mapping table to track the latest data.
Garbage Collection is needed:
Free Pages are used up after a period of time

- **Endurance**
  - **SLC NAND Flash: 100,000 erase counts**
  - **MLC NAND Flash: 10,000 erase counts**

Needs to erase blocks evenly

Flash Translation Layer (FTL)

- A block-device-emulation software
- Built between the MTD Layer and the file system
- Embedded in the flash storage systems

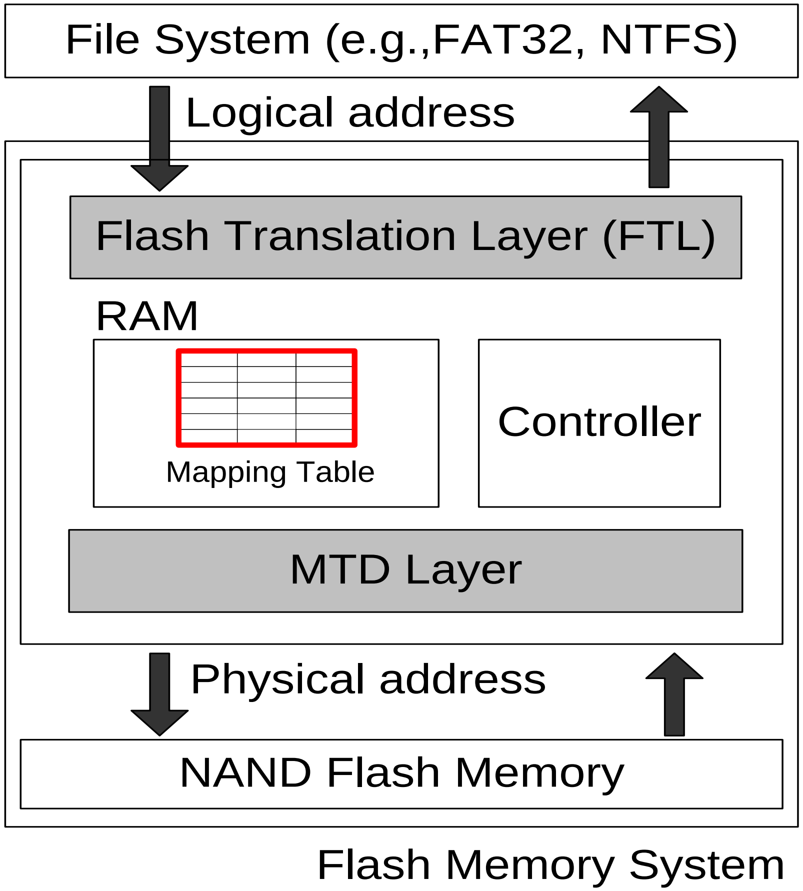


Flash Translation Layer (FTL)

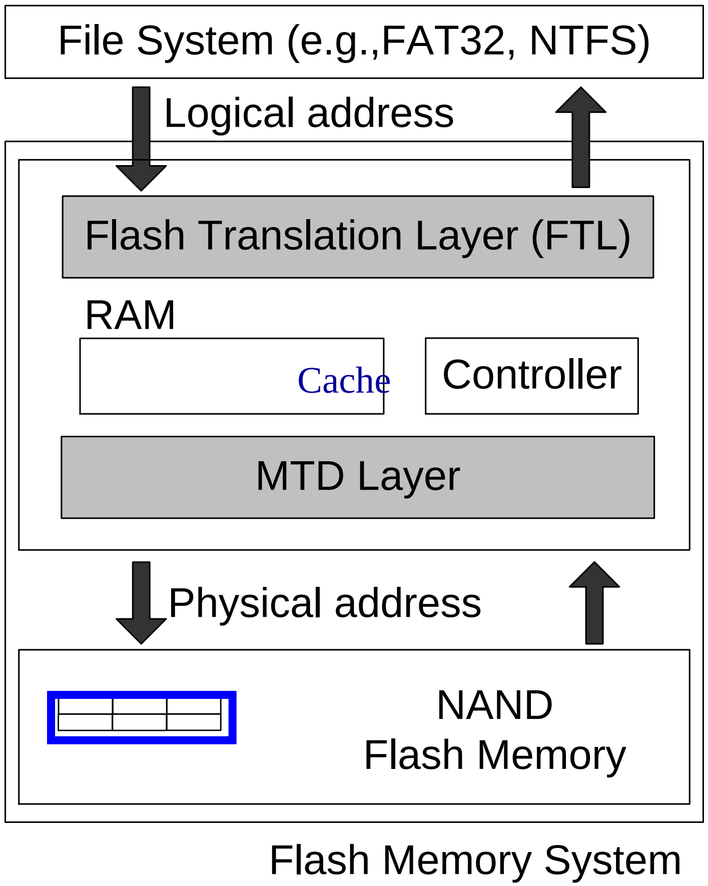
- **Components**
  - **Address Translator**
  - **Garbage Collector**
  - **Wear-leveler**
- **Schemes**
  - **Page-level mapping**
    - **Fine grained mapping -> High performance**
    - **Require large RAM but on-demand approach can solve it.**
  - **Others: Block-level mapping & Hybrid-level mapping**

Can reduce RAM cost but the performance is not as good as page-level mapping

On-demand Approach

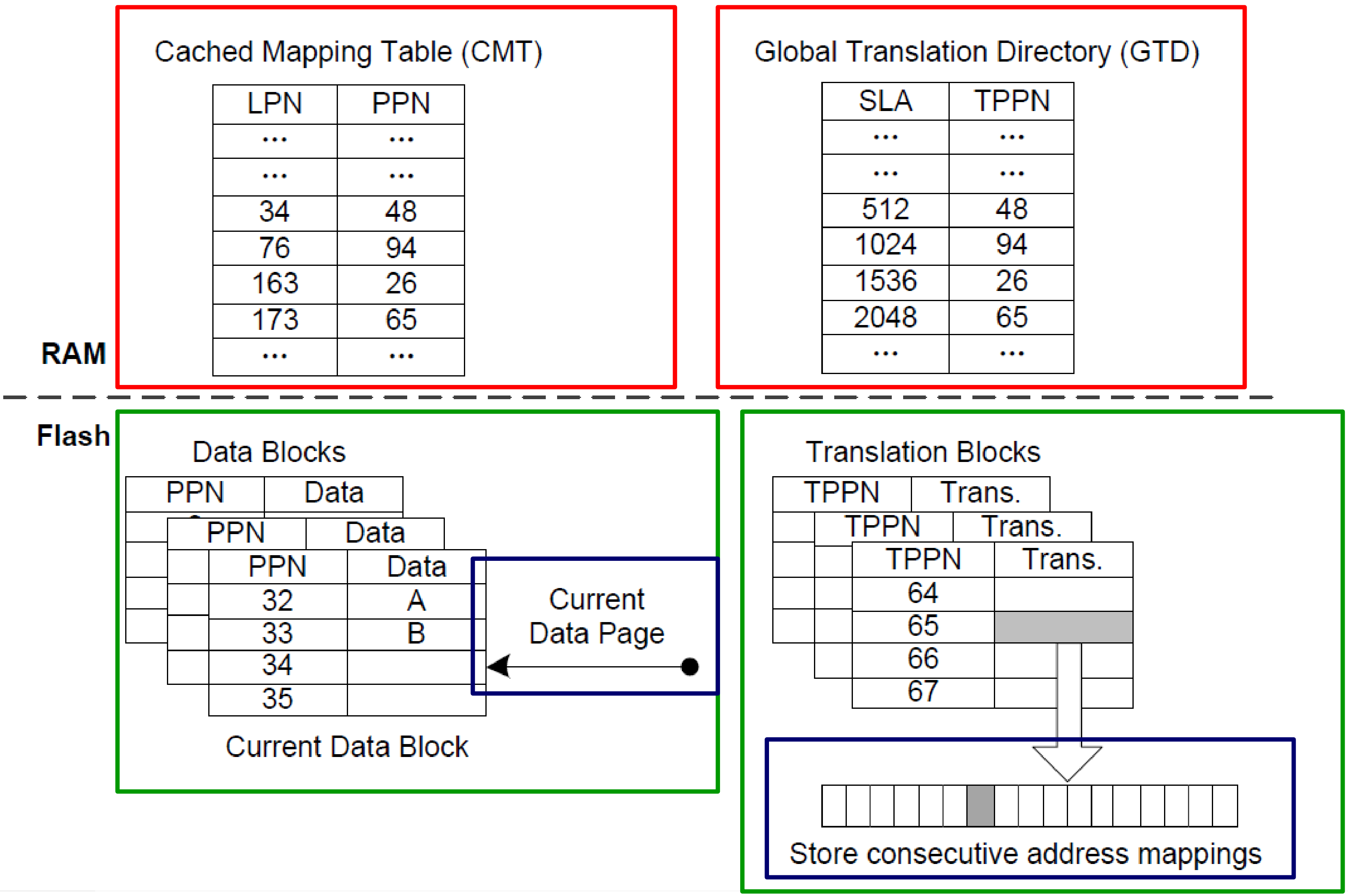


NFTL based system architecture



Demand based system architecture

Demand-based Page-level FTL



Motivation

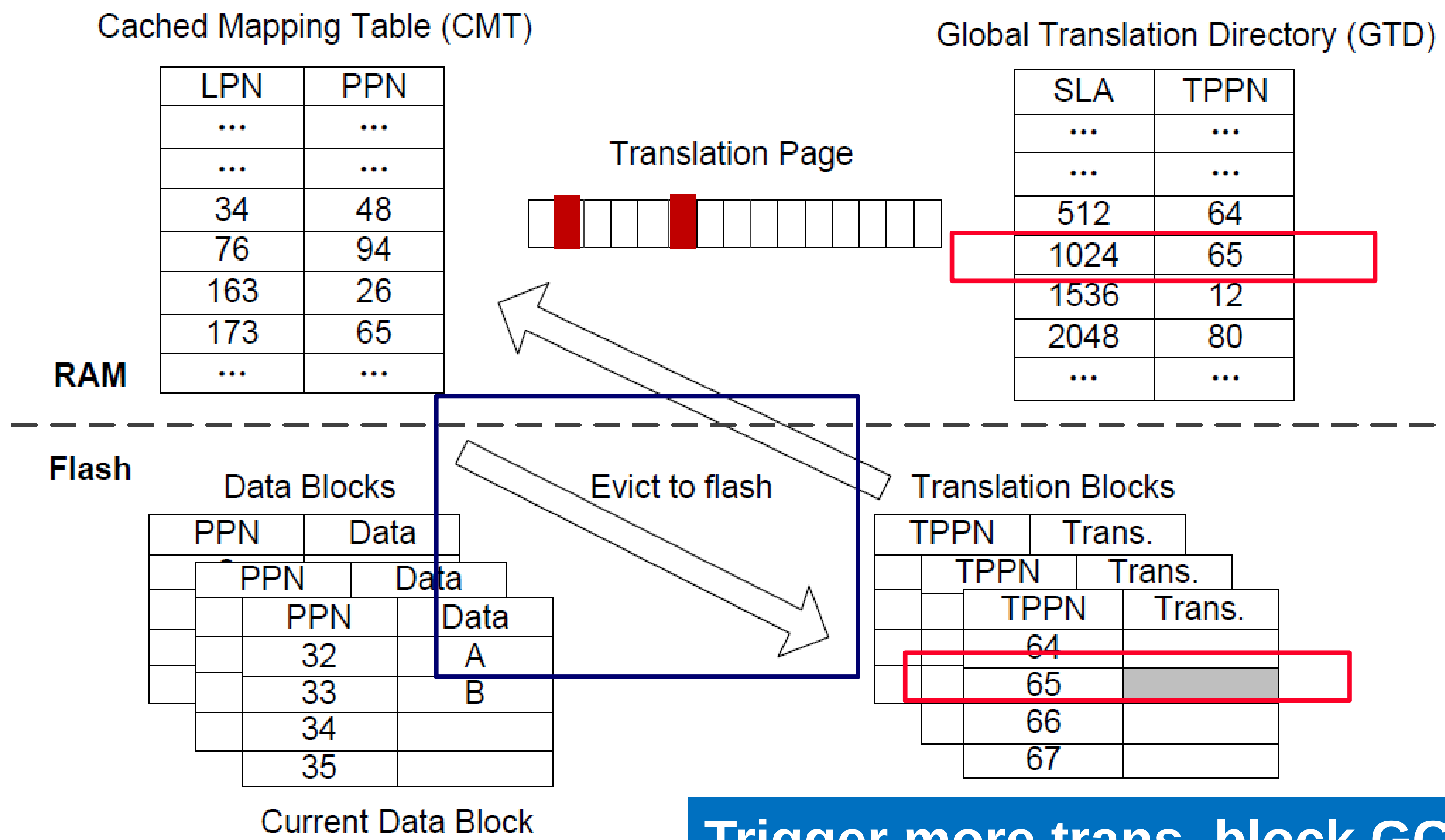
- **Translation Pages in flash memory**

➡ **Extra read/write operations**
Degrade system performance

- **Translation Information Overhead**
  - **From Garbage Collection**
    - **Corresponding translation pages need to be updated**
  - **From Cache Replacement**
    - **Cache miss triggers eviction of translation pages**

Motivation Example on Cache

Data Req. **1025** **1030**



Trigger more trans. block GC

Motivation Example on Data Allocation

Data Req. 511 1011 1025 1576

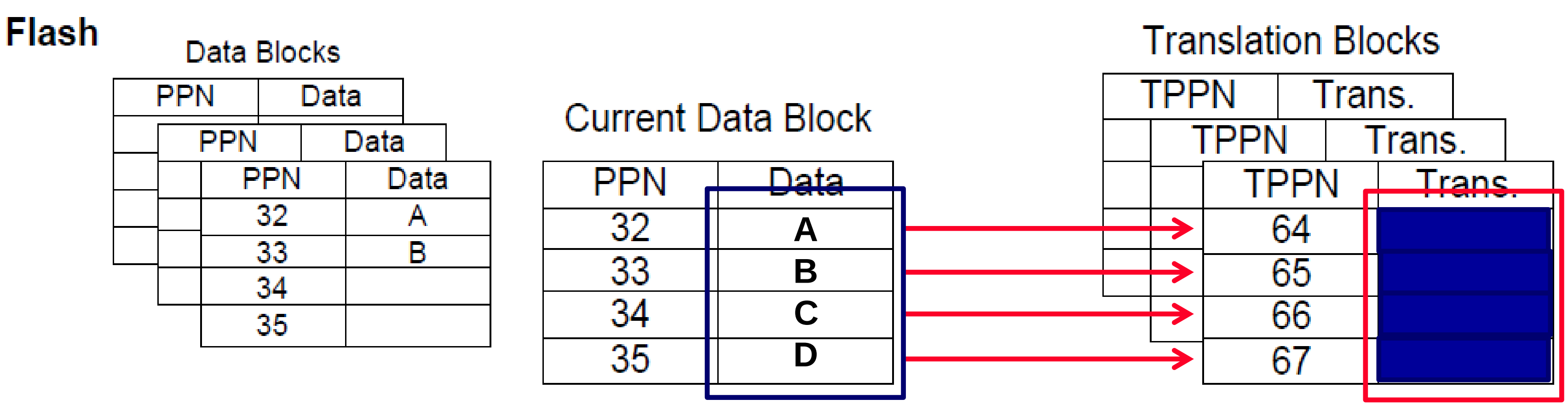
Cached Mapping Table (CMT)

LPN	PPN
…	…
…	…
34	48
76	94
163	26
173	65
…	…

Global Translation Directory (GTD)

SLA	TPPN
…	…
…	…
512	64
1024	65
1536	12
2048	80
…	…

RAM



All related trans. pages need update in GC!

Translation Information Management

- **Optimizing translation information management**
  - **Cache Mechanism**
    - **Directly cache translation pages instead of mapping items**
    - **Directly access the cached translation page by adding a pointer in GTD**
  - **Multiple Write Pointers Strategy**
    - **Data with logical address in the same translation page are written into the same data block**

Caching Mechanism

- **Scheme**
  - **Adopt page-level mapping cache to directly cache translation pages in the RAM**
  - **Add *Cache Index* in GTD to directly access the caching translation pages**
  - **Each access only needs one index computation in GTD, both Cache Index and TPPN are available.**

- **Benefits**
  - **Making use of spatial locality**
  - **Improve mapping utilization and cache hit ratio**
  - **Improve the cache search time**

Caching Mechanism Example

Data Req. **513** **1030**

LPN: 513

LPN: 1030

Global Translation Directory (GTD)

SLA	TPPN	CI
...	...	...
...	...	...
512	64	12
1024	65	-1
1536	12	14
2048	80	8
...	...	...

①

②

Page-level Cached Mapping Table (PCMT)

Index	SLA	Tras. Page
...	...	...
...	...	...
12	512	
13	2560	
14	1536	
15	0	
...	...	...

PPN: 12 PPN: 64

Address Translation Process:

① Compute the GTD Index to locate the GTD item

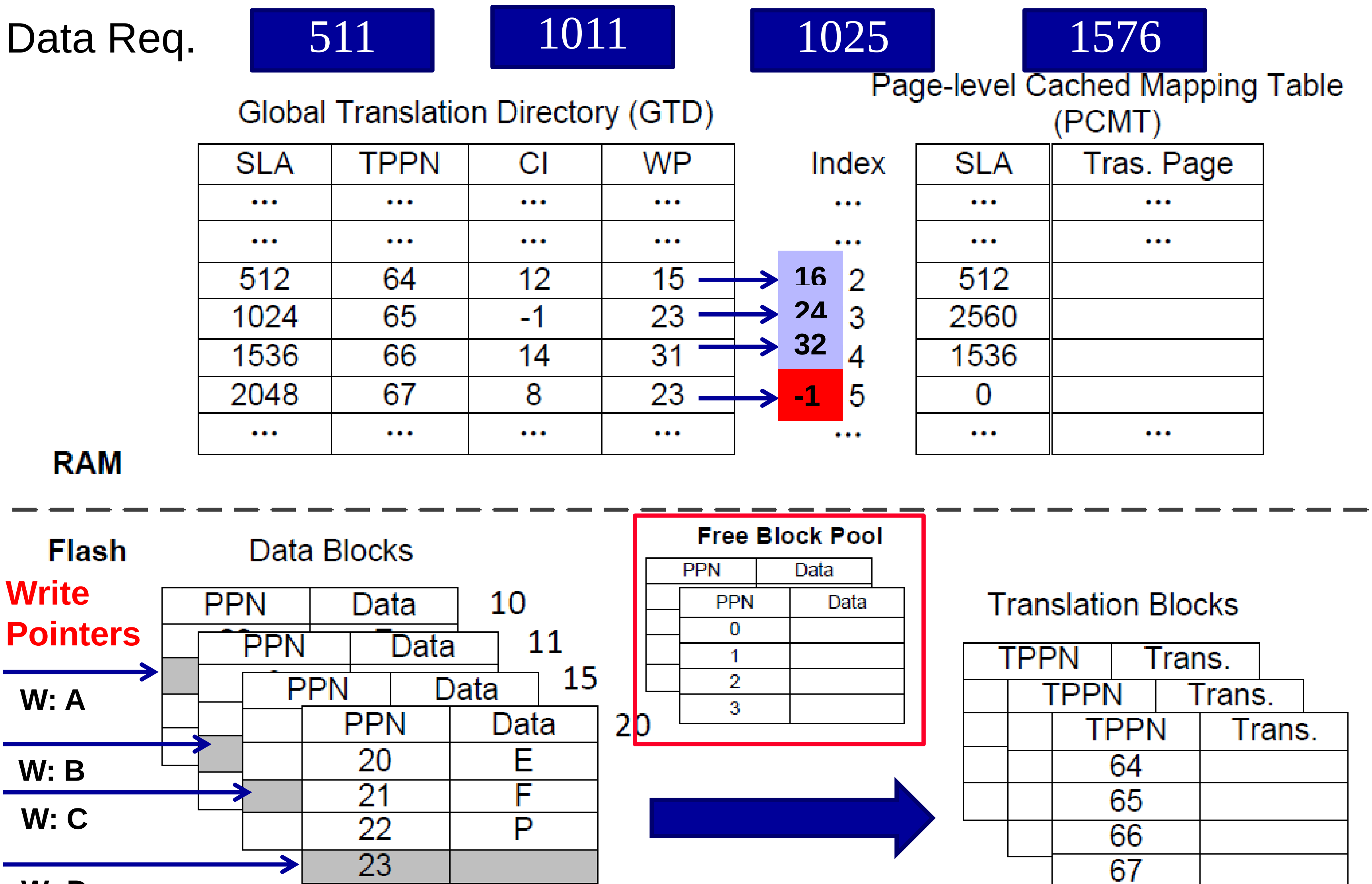
② According to Cache Index (CI) to locate mapping in the Cache

Multiple Write Pointers Strategy

- Objective: No matter the kinds of access pattern, the number of corresponding translation page is at most one.

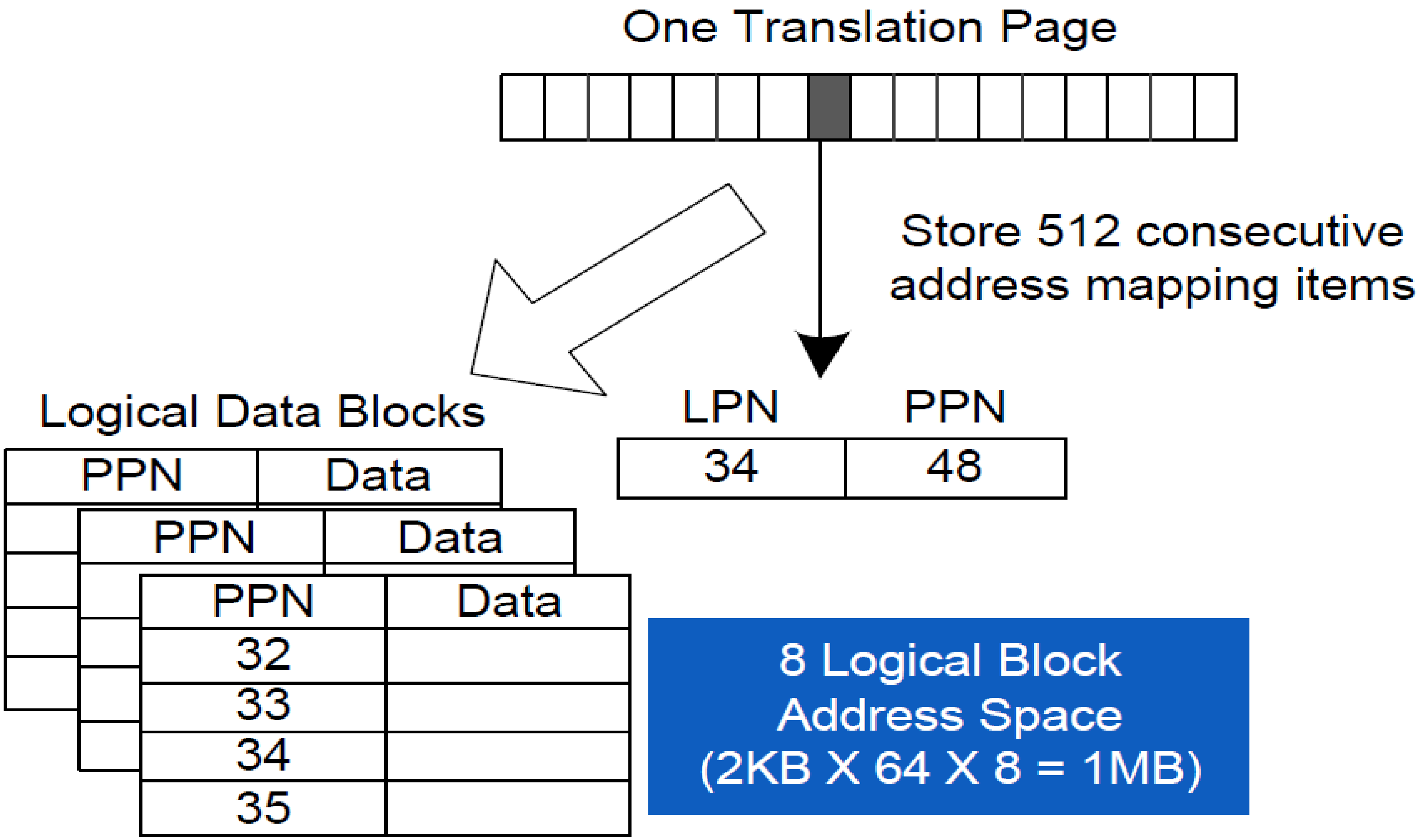
- Multiple Write Pointers Strategy
  - Using Global Translation Directory (GTD) to maintain multiple write pointers
  - According to each individual write pointer, data with logical address in the same translation page are written into the same data block
  - Garbage collector also apply the strategy

MWP Strategy Example



One data block corresponds to at most one trans. page

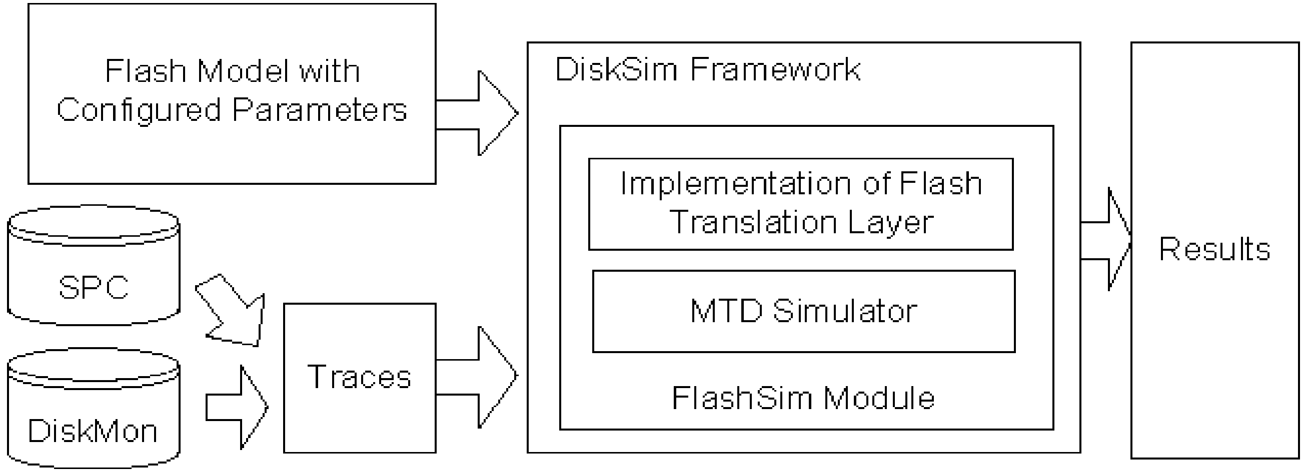
Logical Address Space of One Trans. Page



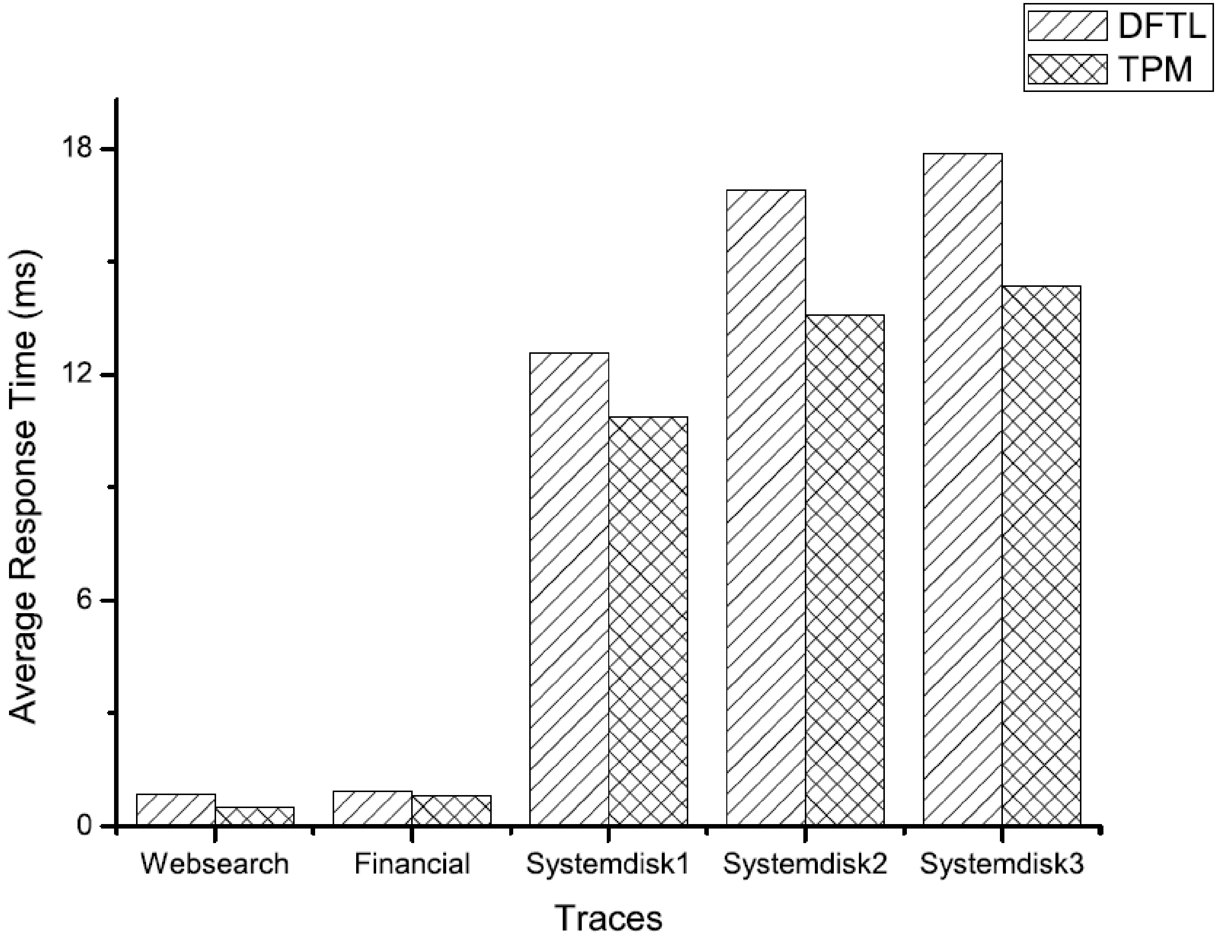
One translation page can map 8 logical blocks (1 MB) to one data block(128KB), so space overhead is small.

Evaluation

- Simulator
  - FlashSim, a simulation framework for flash based storage systems and is built by enhancing Disksim
- Traces
  - Web search, Financial are traces from SPC
  - System disk, collected by running Diskmon on OS
- Compare with the representative Demand-based page-level FTL: DFTL



Average System Response Time

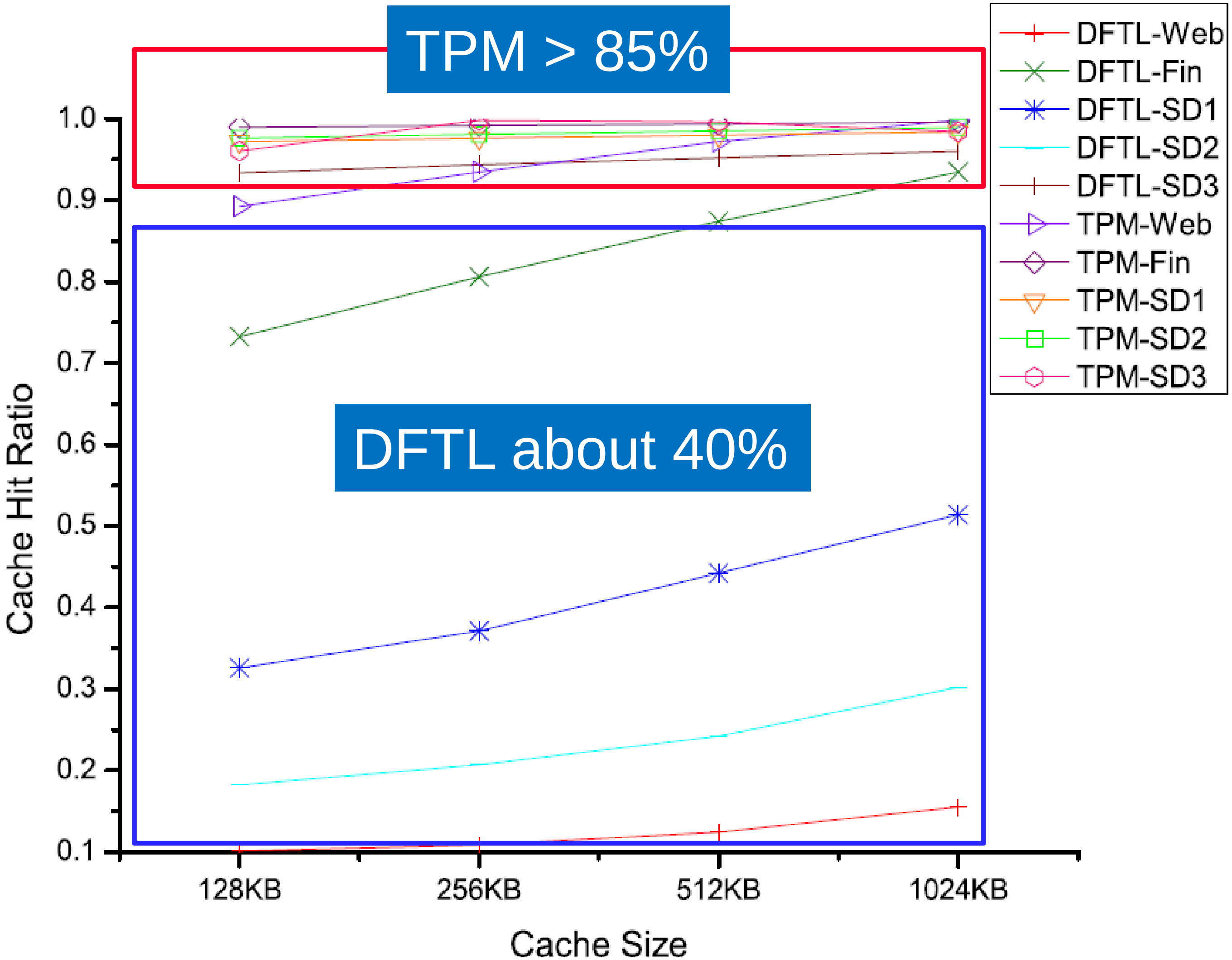


Average improved 22.14%

- **From low trans. page operations**
- **From new data organization**

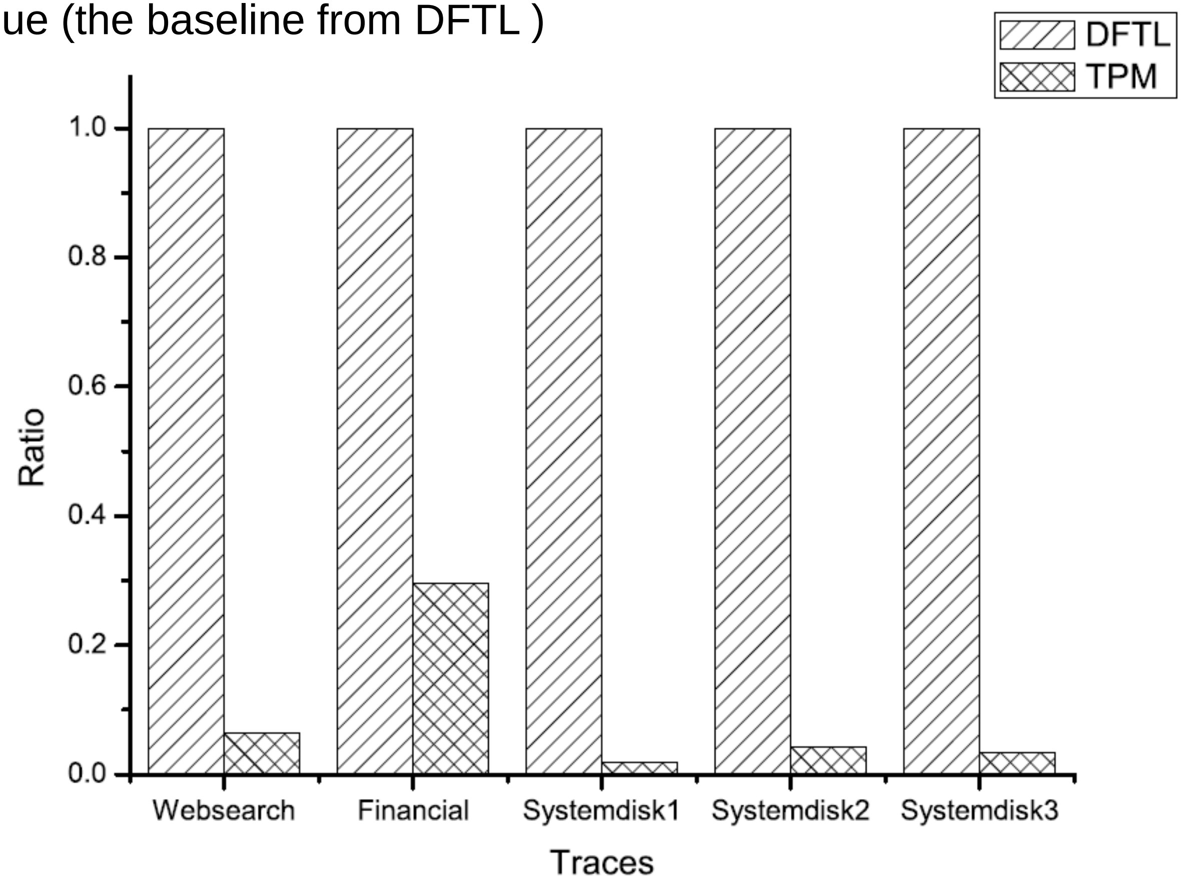
Cache Hit Ratio

- **Compare the cache hit ratio with DFTL in different cache size: 128KB, 256KB, 512KB, 1024KB**



Translation Page Updates

Normalized Value (the baseline from DFTL)

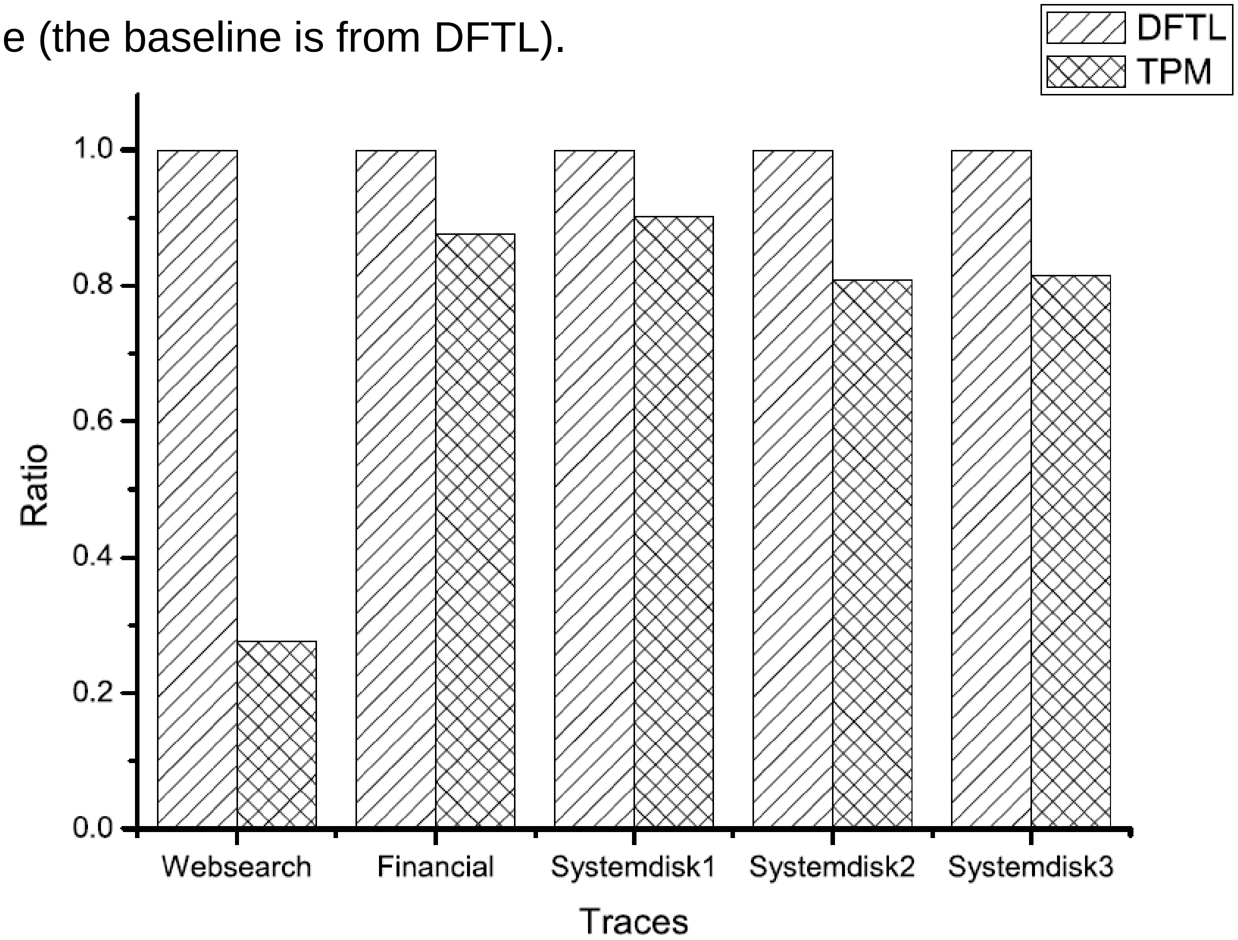


Average reduced 90.93%

- **From high cache hit ratio**
- **From low trans. page update during GC**

Block Erase Counts

Normalized Value (the baseline is from DFTL).



Average reduced 26.51%

- **From high cache hit Ratio**
- **From new data organization**

Overhead

- **Space overhead**
  - **Some blocks may not be fully utilized by allocating to one translation page**
  - **But the overhead is small. In our experiment, only 4.74% extra space overhead in 32GB NAND flash memory**

- **RAM space overhead**
  - **Maintain *Cache Index* and *Write Pointers* in GTD**
  - **Only needs 2KB RAM space for 1GB flash**

Conclusion

- We proposed TPM to optimize translation information management for demand-based page-level mapping scheme in NAND flash storage systems.

- TPM can reduce the extra translation page updates through our caching mechanism and novel data allocation strategy.

- Experimental results show that our scheme is very effective compared with the previous work.

Thanks