WAFTL:
A Workload Adaptive Flash Translation Layer with Data Partition

Qingsong Wei †

Bozhao Gong€, Suraj Pathak€, Bharadwaj Veeravalli€,

Lingfang Zeng€ and Kanzo Okada†

† Data Storage Institute, A-STAR, Singapore

€ National University of Singapore

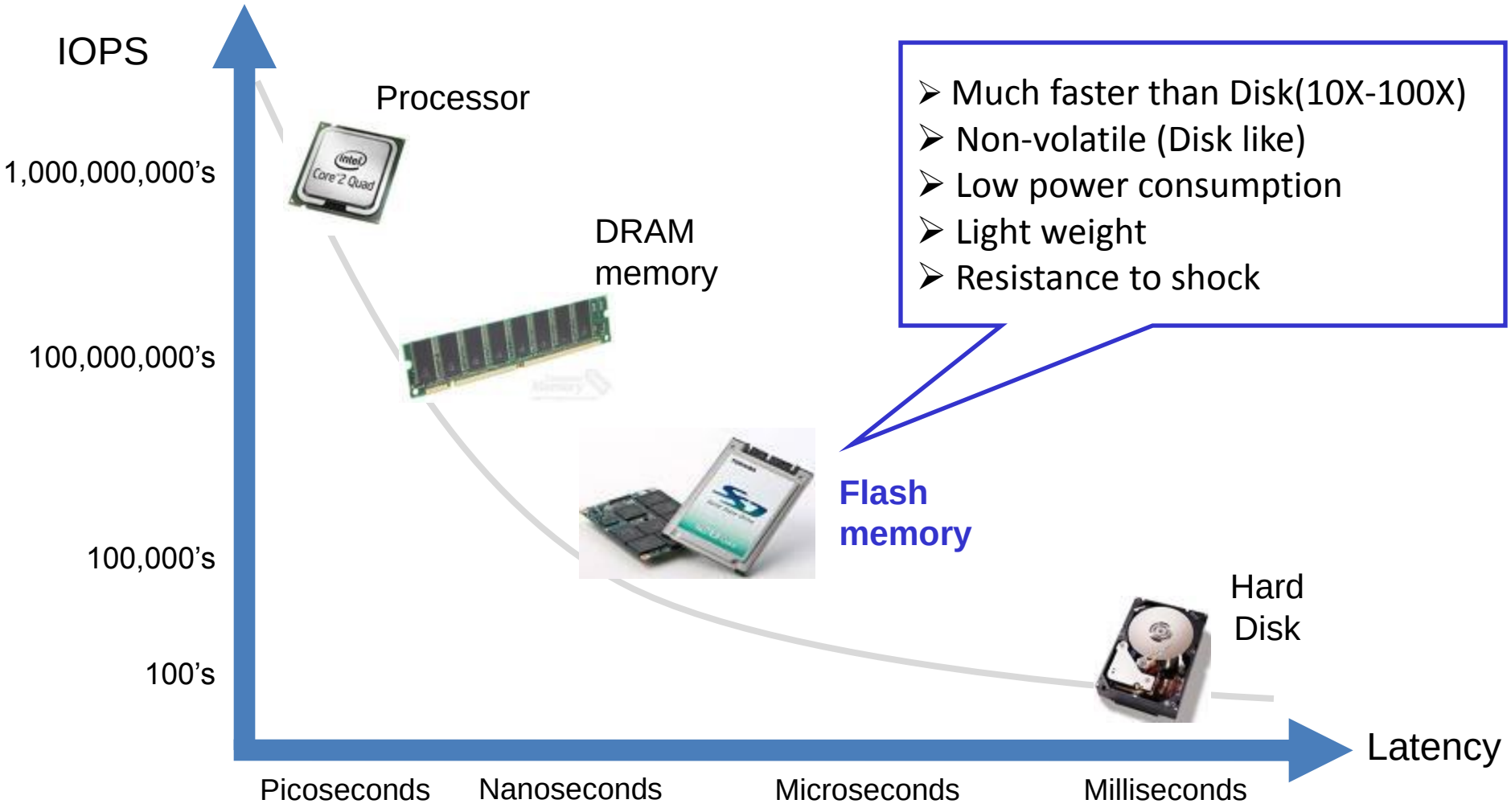
MSST 2011

May 26, 2011

Data Storage Institute

Introduction

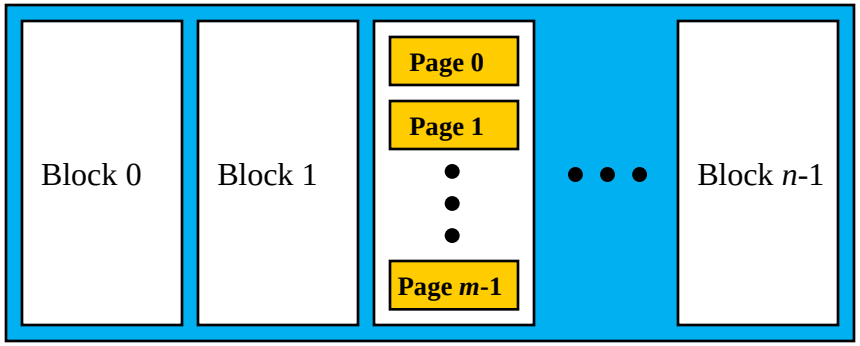
Flash Fills the Performance Gap



Introduction

NAND Flash Memory

- Different access unit
  - ✓ Read/write in pages (*us*)
  - ✓ Erase in blocks (very slow, *ms*)
- Out-place-Update: Does not allow overwrite.
- Limited number of erase per cell. 100-300K for SLC and 10-30K for MLC.
- Poor random write performance.

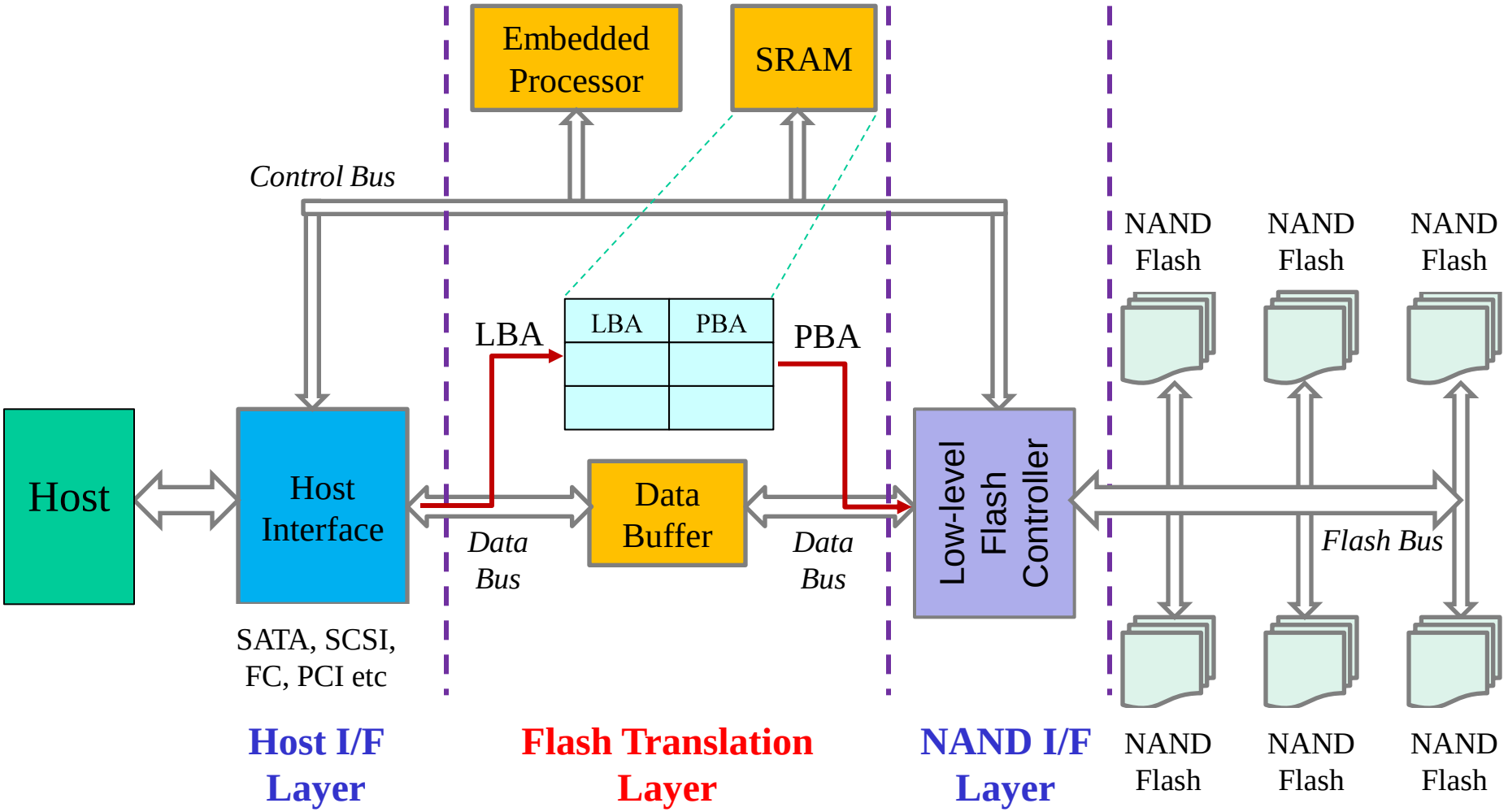


m=64/128/256

Flash Memory

Introduction

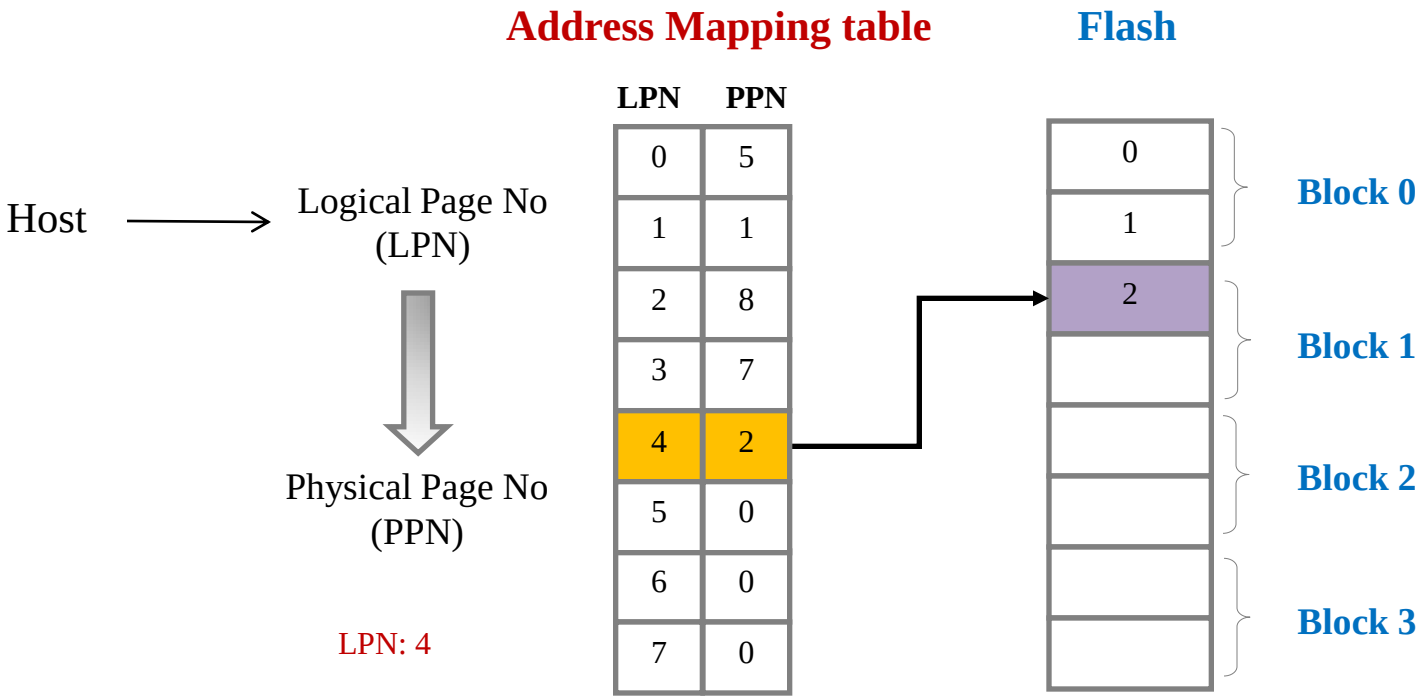
Flash Translation Layer (FTL) in SSD



State-of-the-Art

Page-based FTL

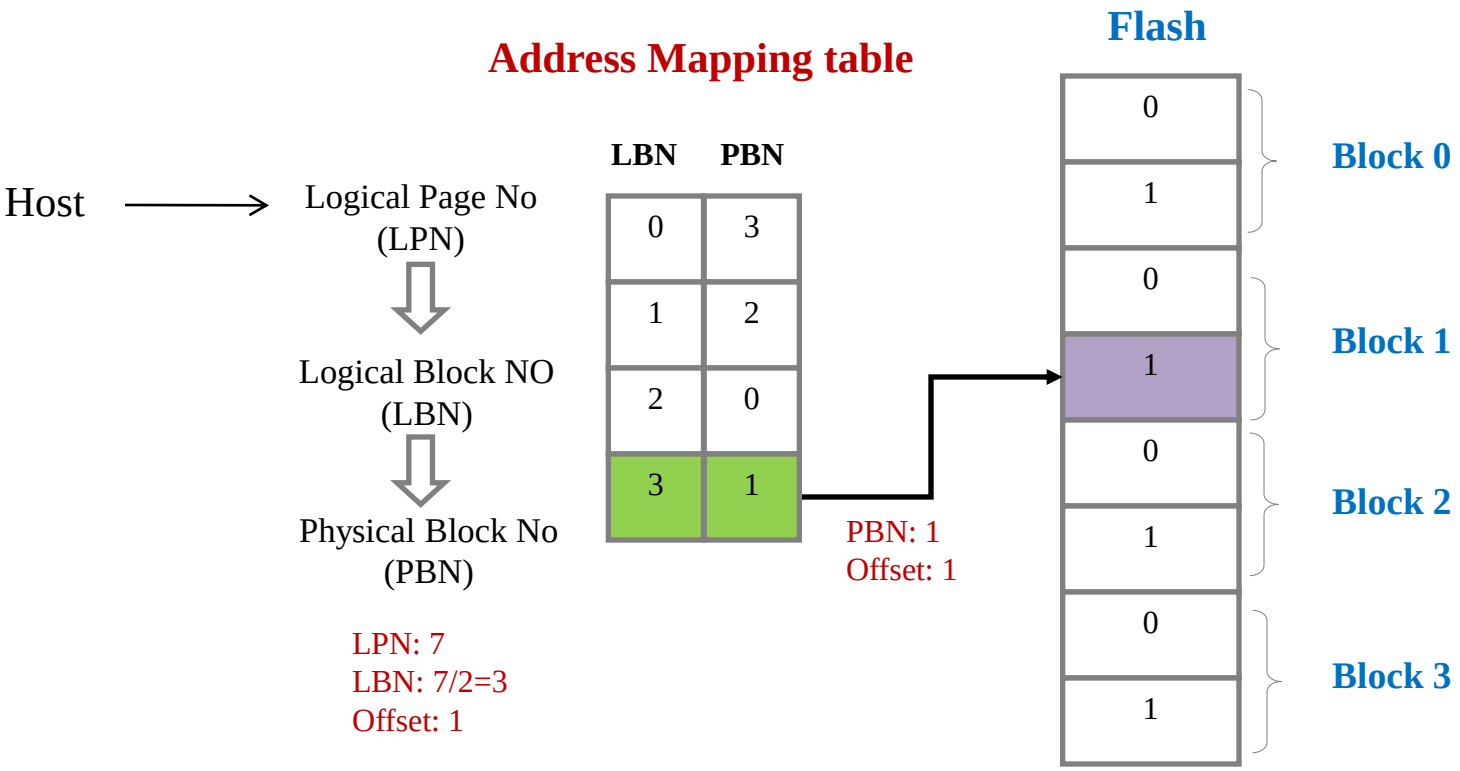
+ High performance
+ High space usage
+ Low garbage collection overhead
- Large mapping table
- Not scalable



State-of-the-Art

Block-based FTL

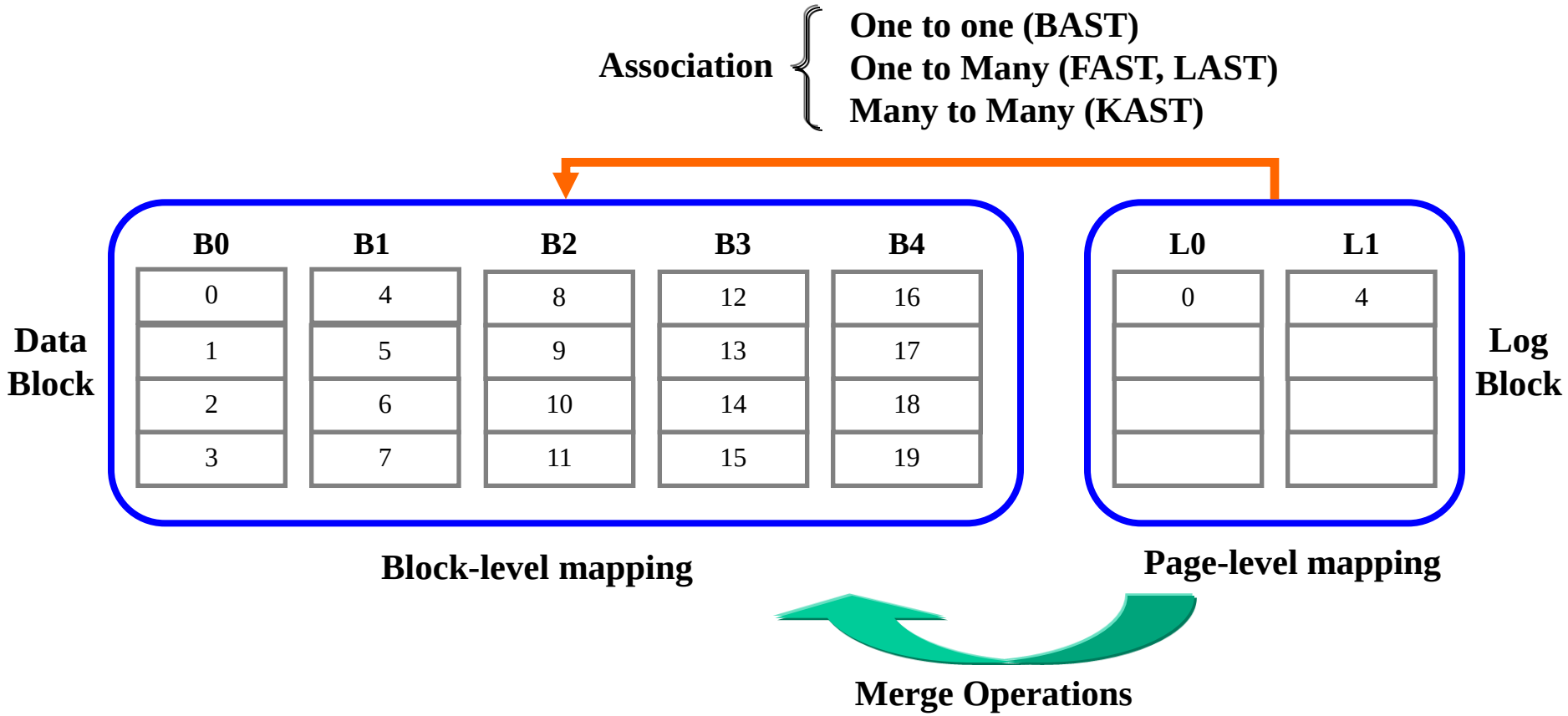
+ Small mapping table
- Low performance
- Low space usage (internal fragmentation)
- High garbage collection overhead



State-of-the-Art

Log-buffer based Hybrid FTL

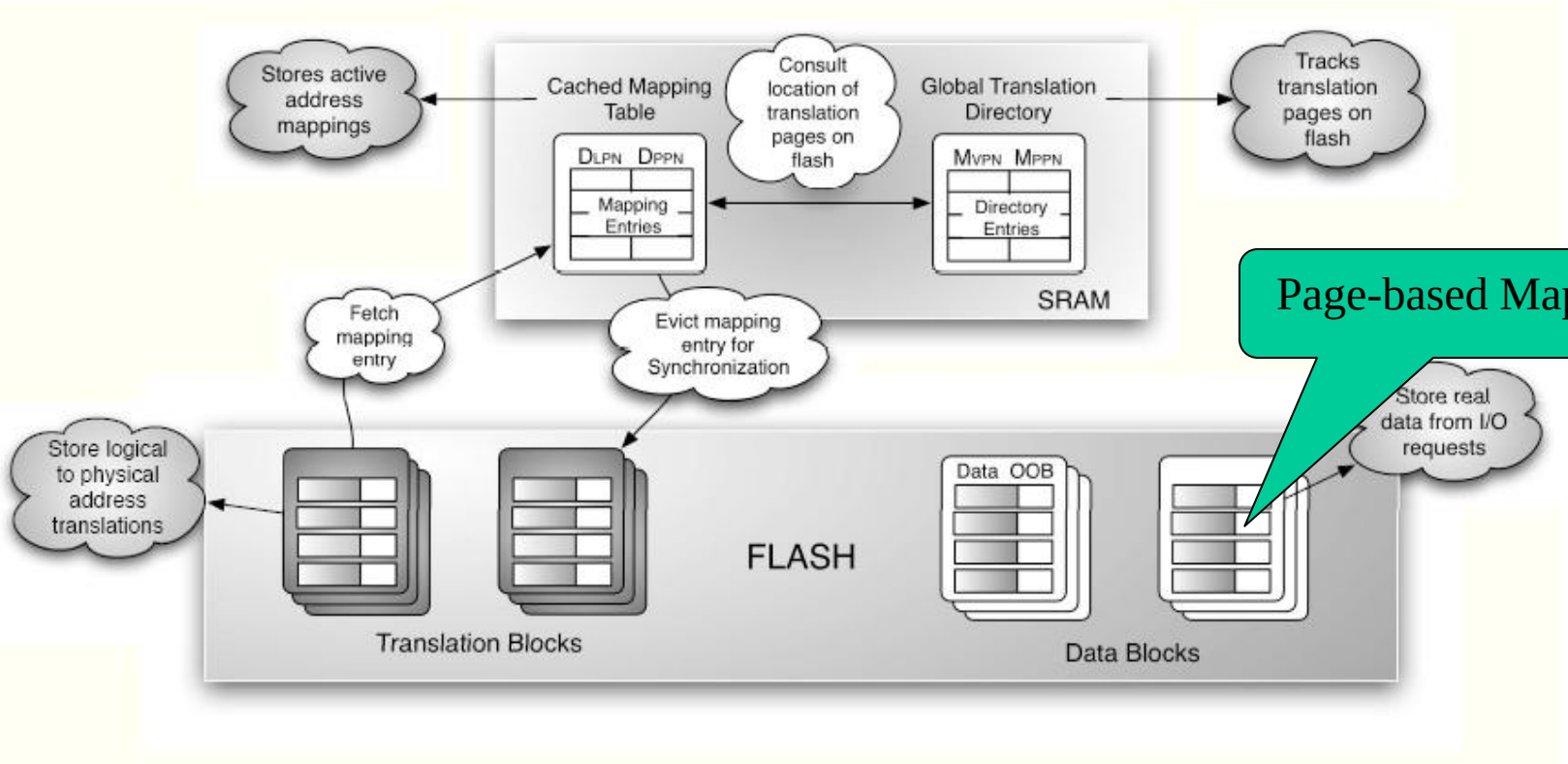
+ Small mapping table
+ Improved performance comparing to Block-based FTL
- Poor performance for random writes because of expensive merge operations



State-of-the-Art

DFTL: Demand-based FTL

+ Partial buffered page-based FTL
+ Good Performance (but lower than pure page-based FTL)
+ Reduced memory requirement
- Large mapping table



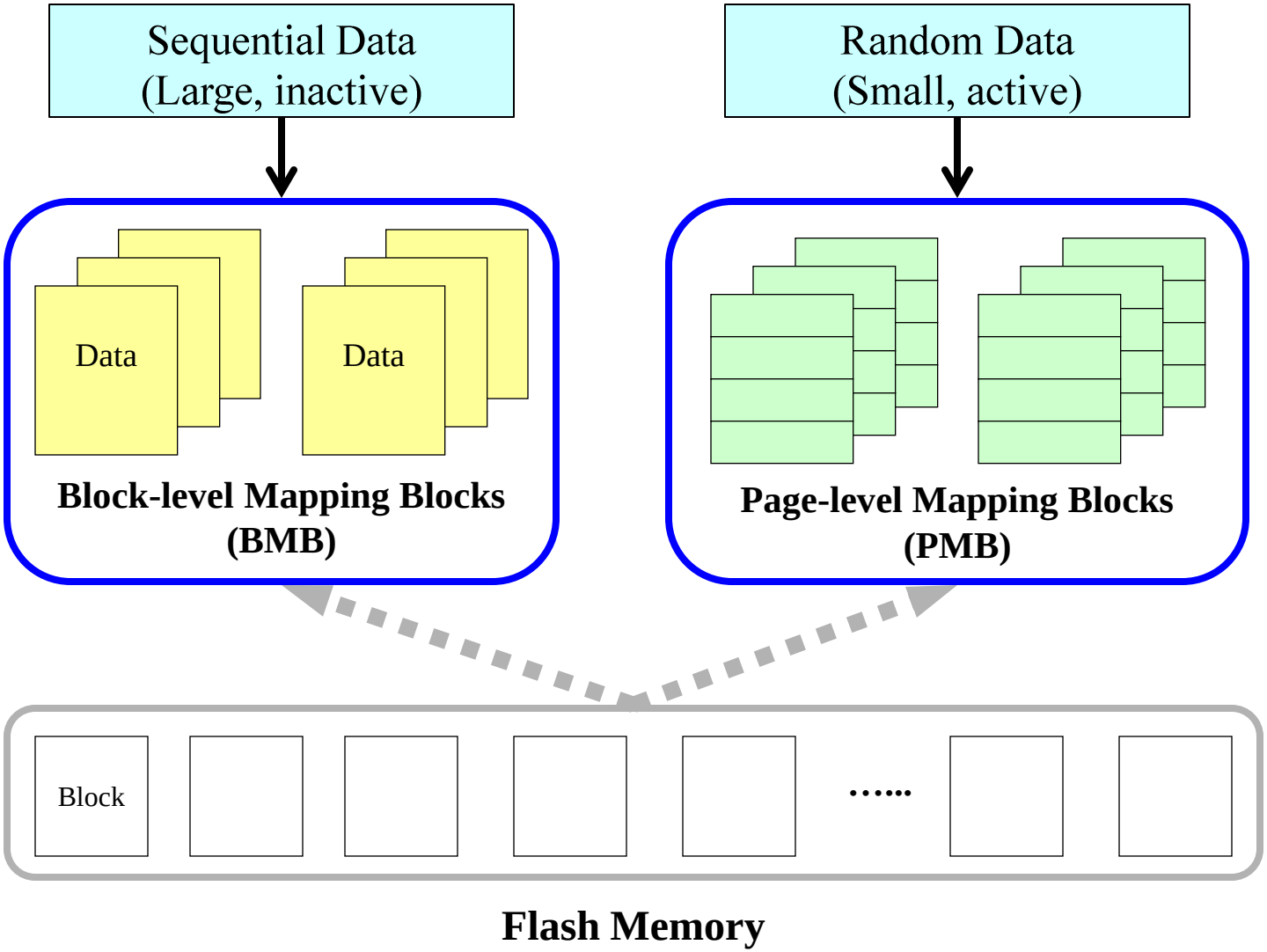
State-of-the-Art

	Page-based FTL	DFTL	Block-based FTL	Hybrid FTL
Performance	High	Good	Low	Middle
Garbage Collection Overhead	Low	Low	High	Middle
Favored Workload	Random	Random	Sequential	Sequential
Mapping table Size	Large	Large	Small	Small
Workload adaptive	No	No	No	No
Scalable with SSD capacity	No	No	Yes	Yes

Generally limited in one way or another, either in memory requirement, performance, garbage collection overhead or scalability.

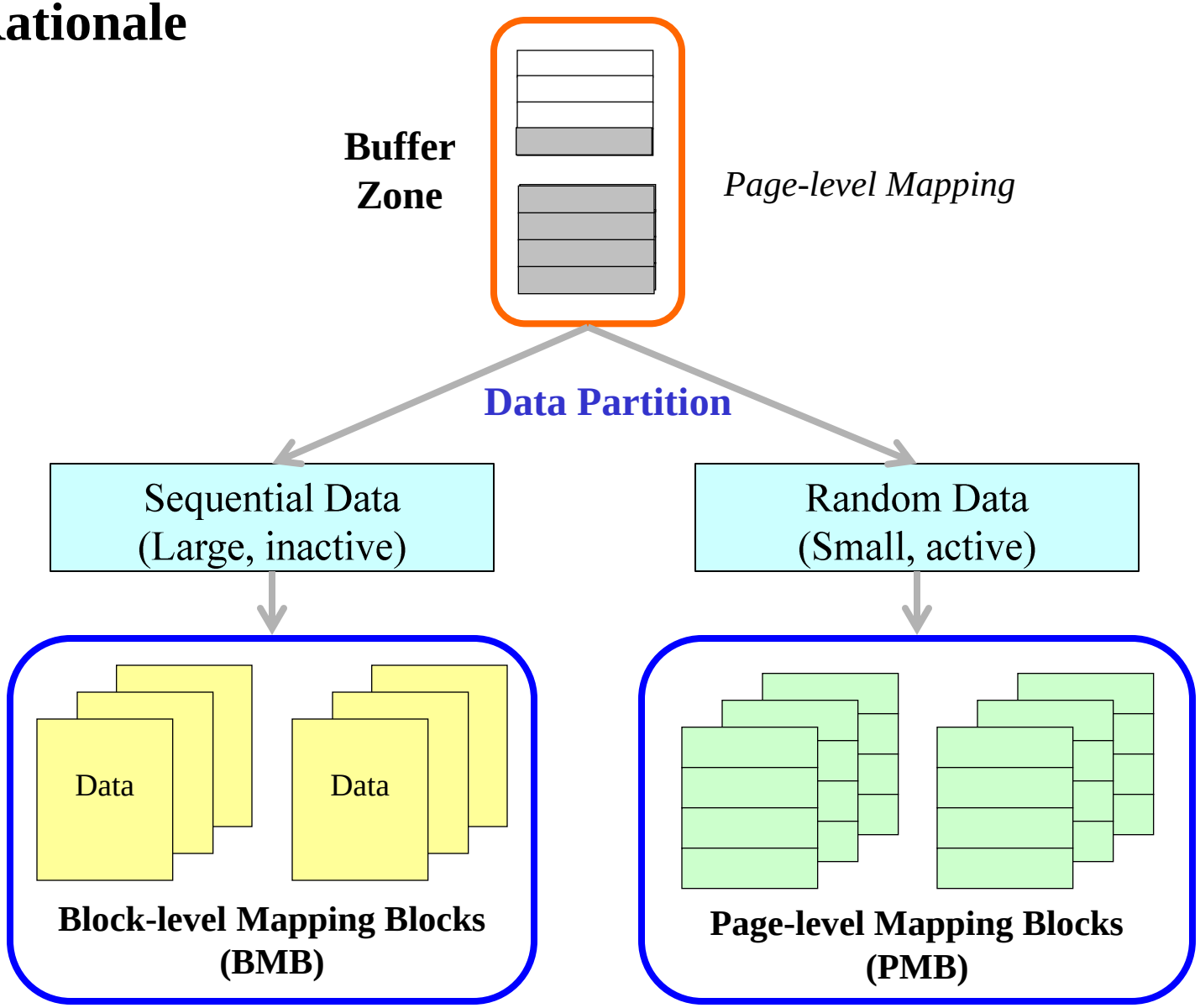
WAFTL: Workload Adaptive FTL

Design Rationale



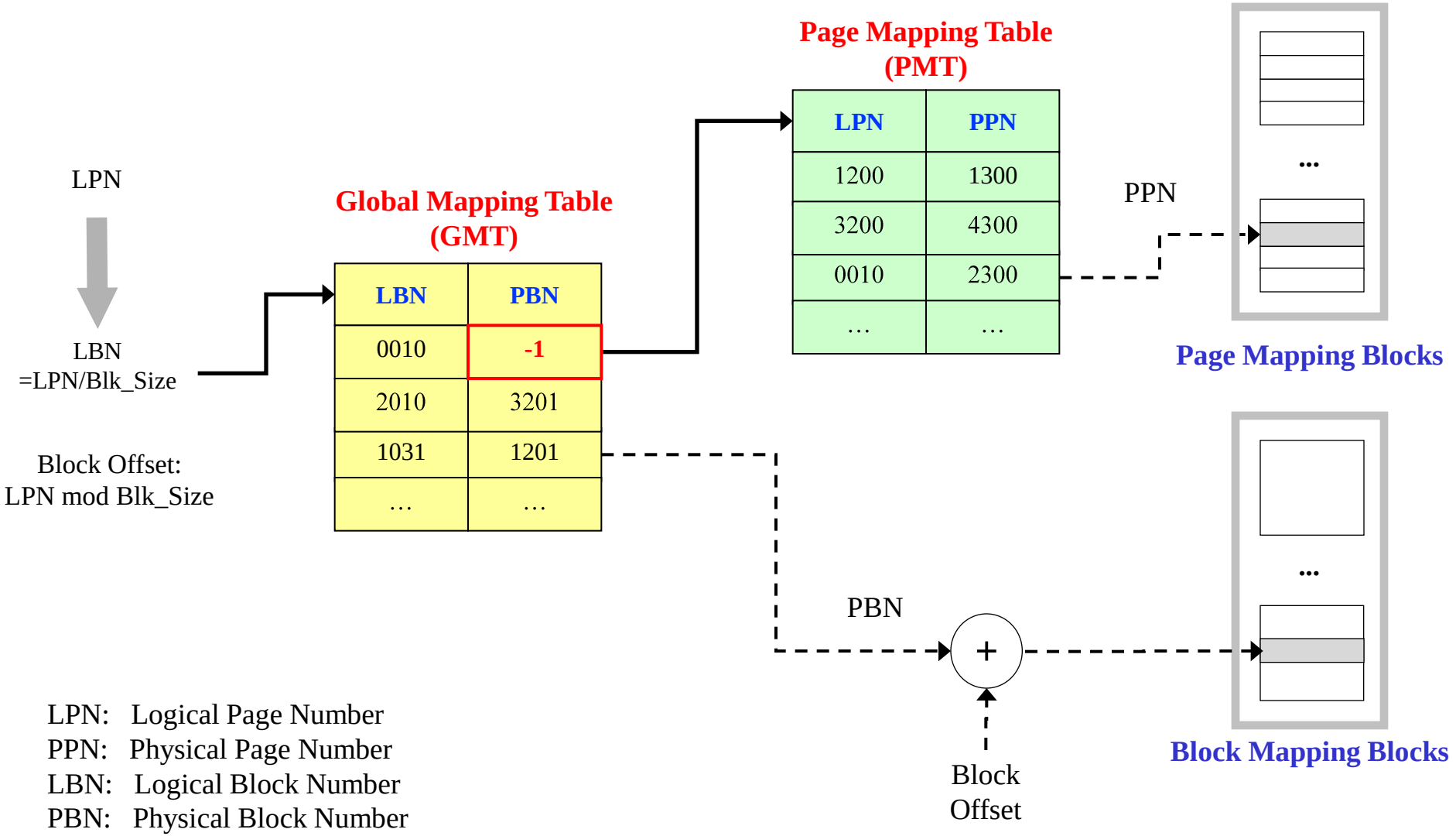
WAFTL: Workload Adaptive FTL

Design Rationale



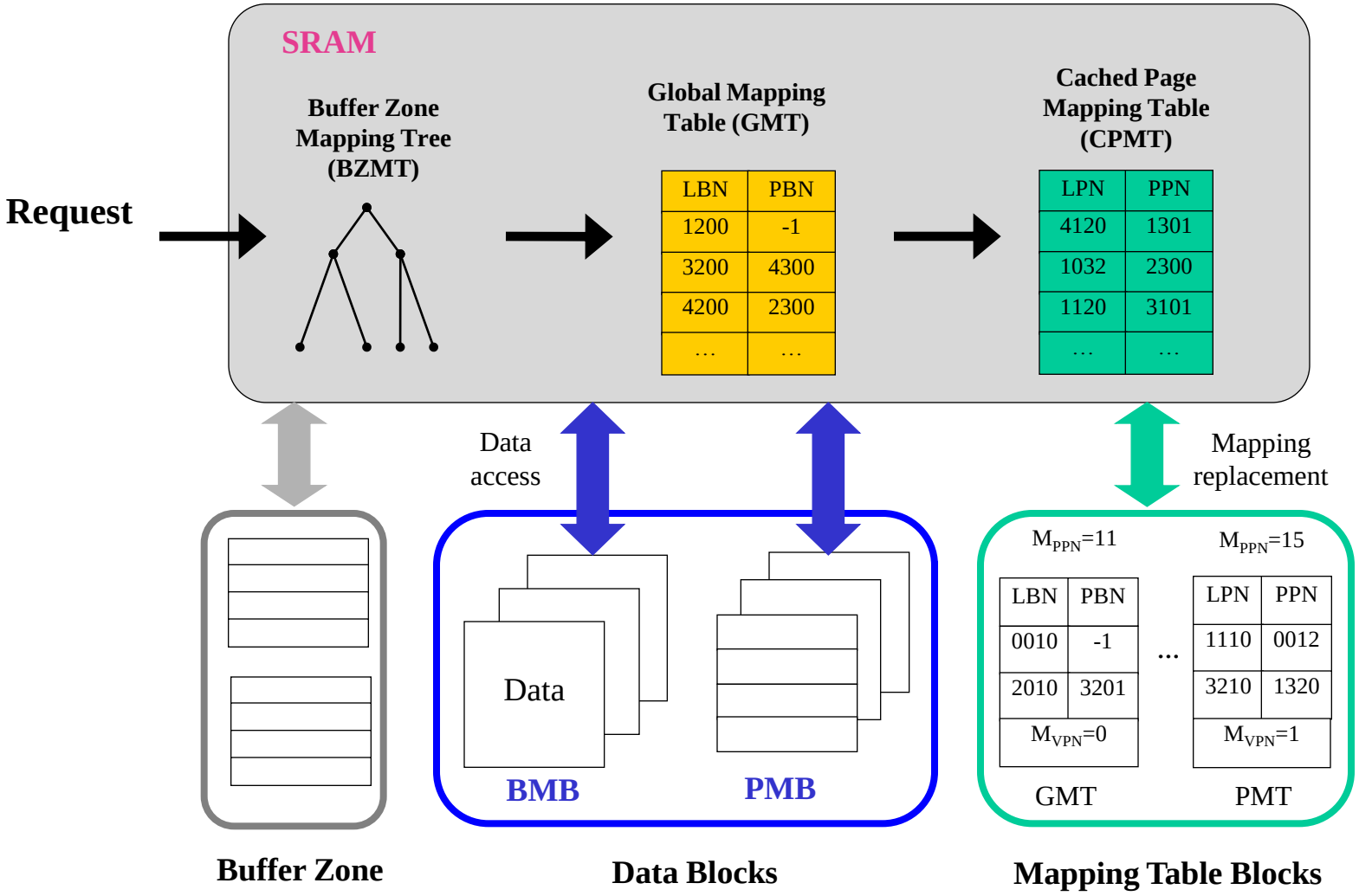
WAFTL: Workload Adaptive FTL

Two-level Address Translation Process



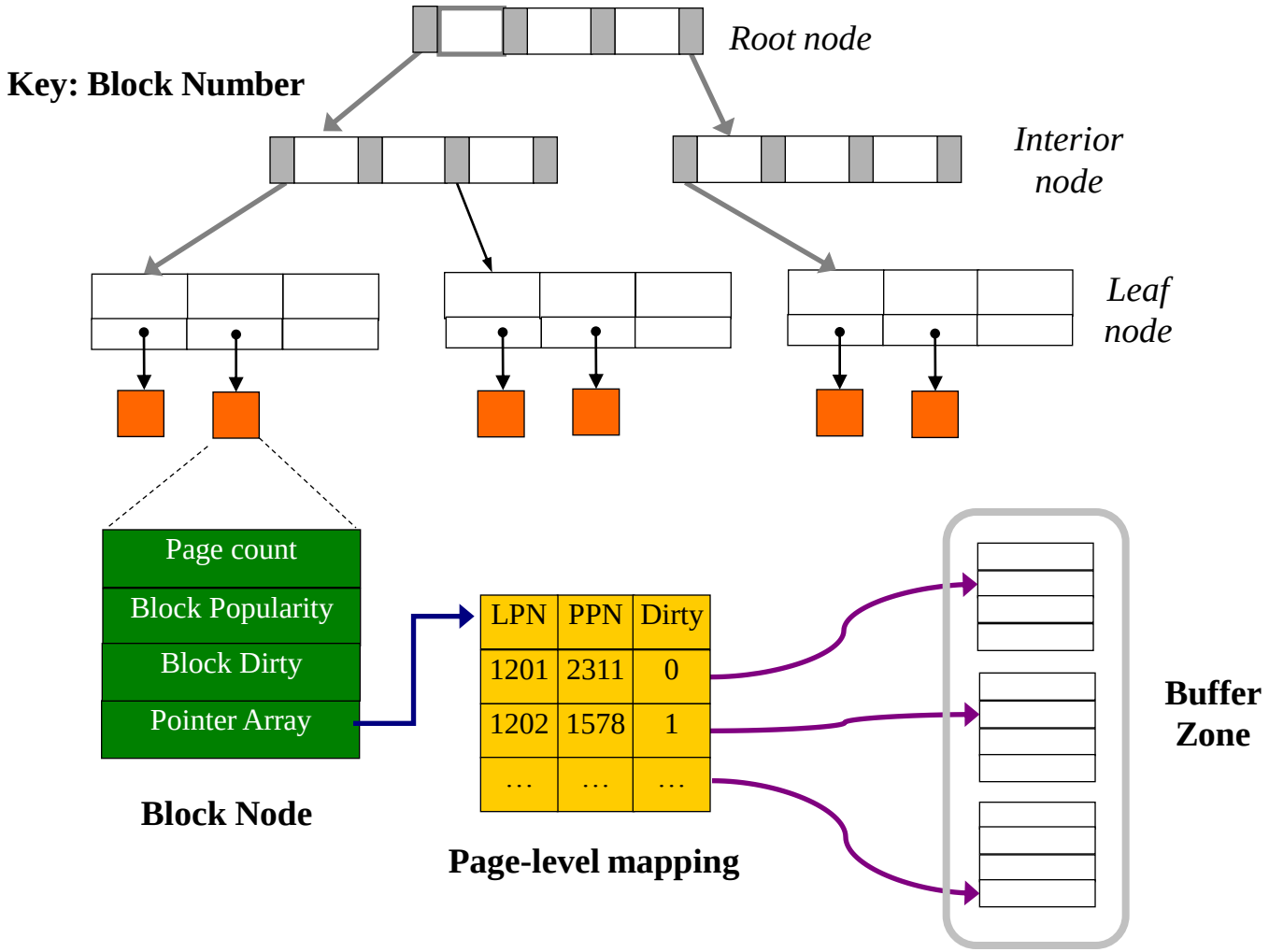
WAFTL: Workload Adaptive FTL

Layout and In-memory Data Structure



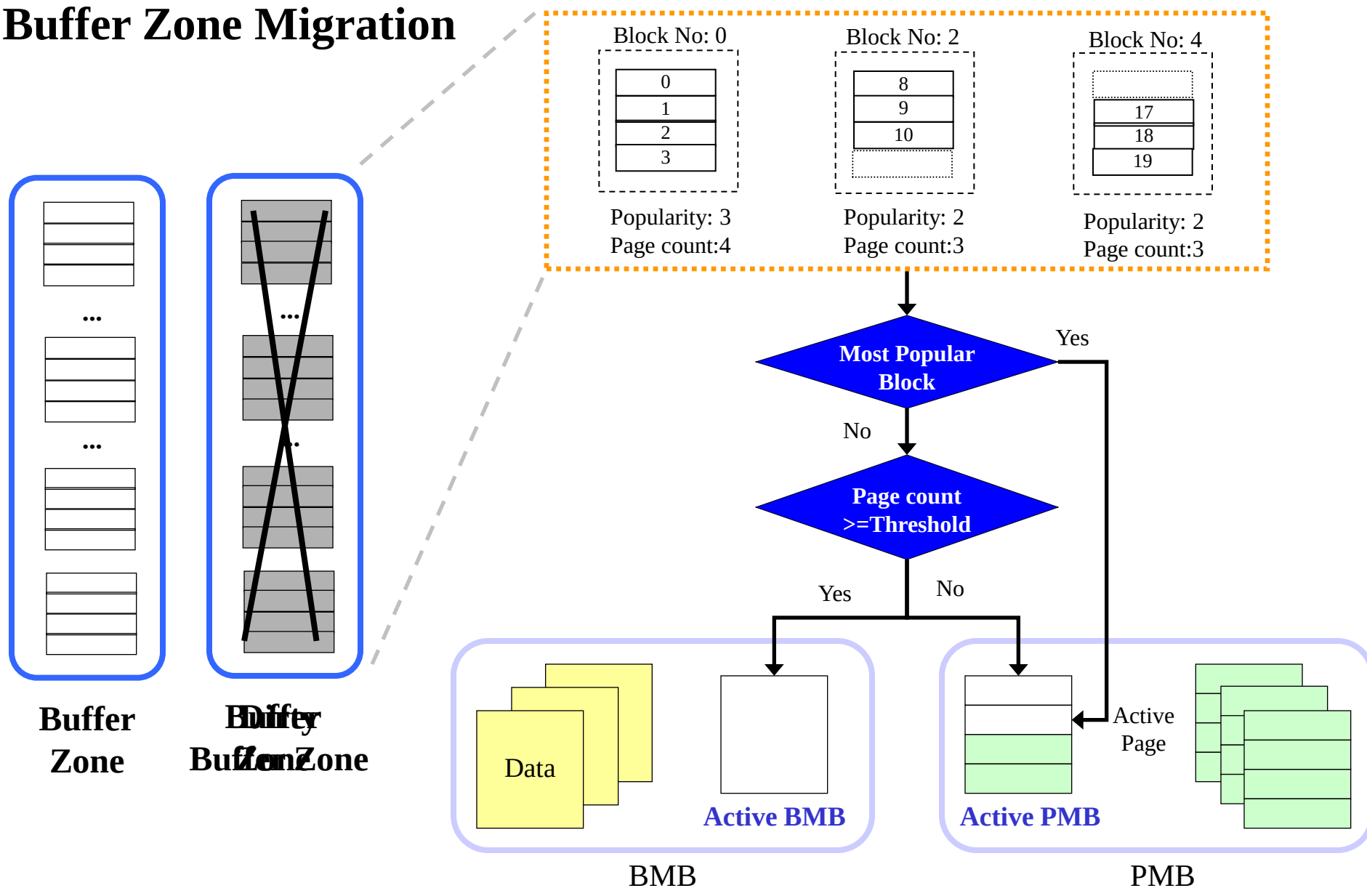
WAFTL: Workload Adaptive FTL

Buffer Zone Mapping Tree



WAFTL: Workload Adaptive FTL

Buffer Zone Migration



WAFTL: Workload Adaptive FTL

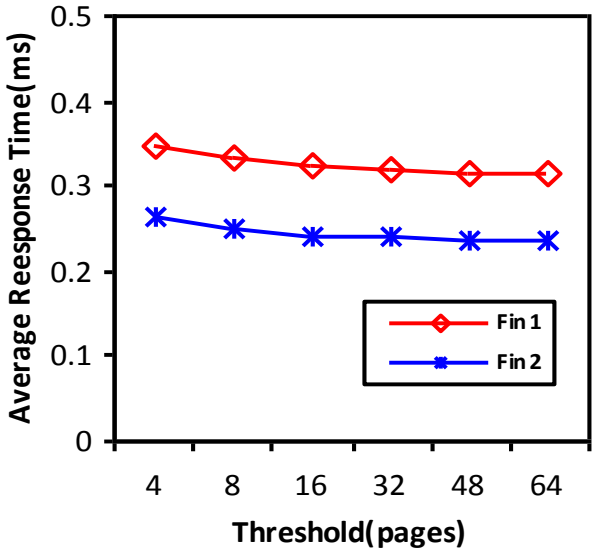
Determining Migration Threshold

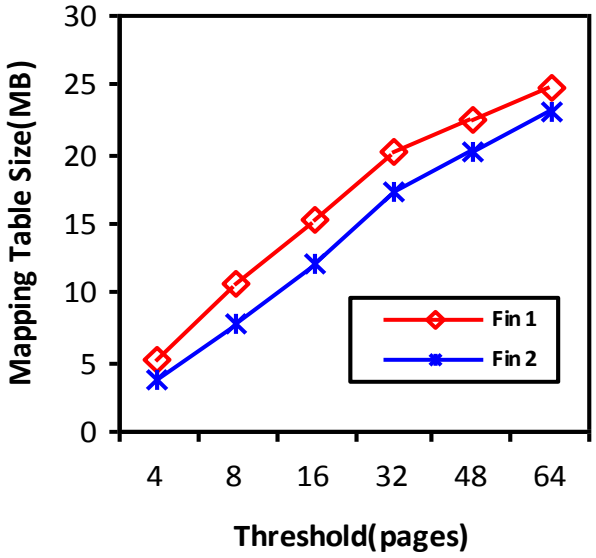
- ***Performance oriented threshold***
  - Threshold is statically set as full block size and only full block data is stored into BMB.
  - *BEST performance, but Big Mapping table*
- ***Dynamic threshold considering memory constraint***
  - Initially, *THR* is set to Block size
  - THR decreases to a smaller value if performance degradation is smaller than D, and mapping table reduction is larger than D. D is a control parameter defined by users.

$$\frac{\Delta P}{P_{Blk\_size}} = \frac{P_{THR} - P_{Blk\_size}}{P_{Blk\_size}} \le D$$

$$\frac{\Delta M}{M_{Blk\_size}} = \frac{M_{Blk\_size} - M_{THR}}{M_{Blk\_size}} \ge D$$

P_{THR} and P_{Blk_size} represent performance when threshold is set as *THR* and full block, respectively. M_{THR} and M_{Blk_size} represent mapping table size when threshold is set as *THR* and full block, respectively

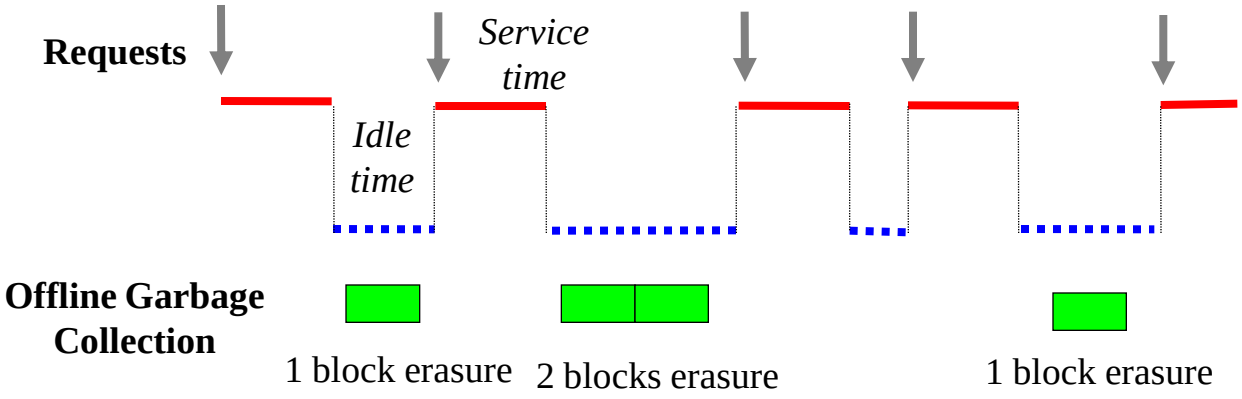




Mapping table size is more sensitive to threshold than performance.

WAFTL: Workload Adaptive FTL

Garbage Collection Policy



- Idle period ($t_{predict}^{i}$) is predicted as $$t_{predict}^{i} = \alpha t_{real}^{i-1} + (1-\alpha) t_{predict}^{i-1}$$
- WAFTL use predicted idle length to calculate and determine how many invalid blocks (N_i) to be erased according to following Equation

$$N_i = \begin{cases} \dfrac{t_{predict}^{i}}{T_{erasure}} & if \quad t_{predict}^{i} \geq T_{erasure} \\ 0 & if \quad t_{predict}^{i} \prec T_{erasure} \end{cases}$$

Evaluation

Setup

- SSD Simulator
- FTL schemes: WAFTL, Pure Page-based, DFTL, FAST
- 7 enterprise workloads

Evaluation Metrics

- Average response time
- Erase count
- Page read/write operations

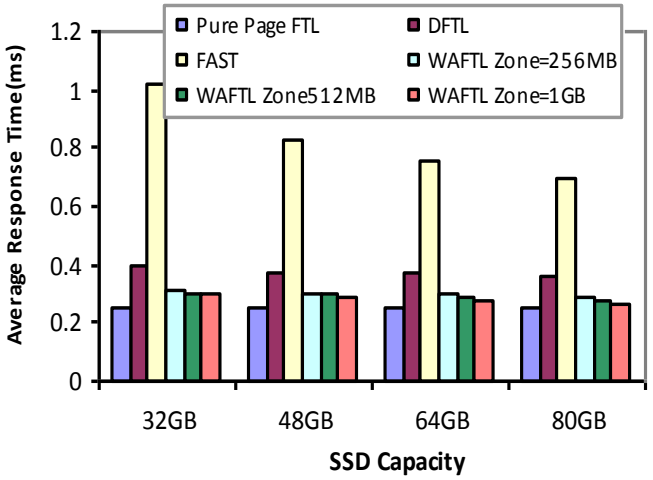
Configuration

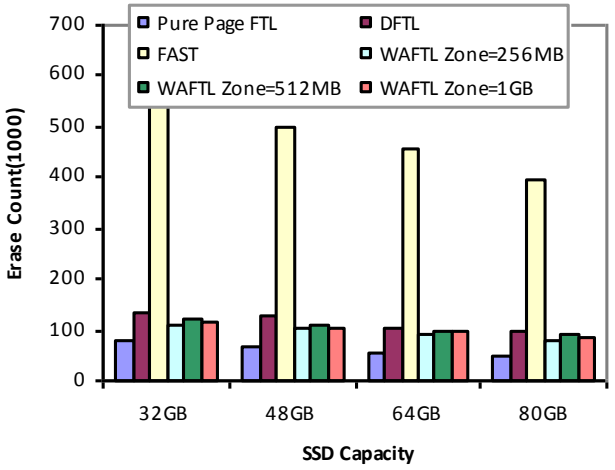
SSD	Page Read to Register	25μs
	Page Program (Write) from Register	200μs
	Block Erase	1.5ms
	Serial Access to Register (Data bus)	100μs
	Die Size	2 GB
	Block Size	256 KB
	Page Size	4 KB
	Data Register	4 KB
	Erase Cycles	100 K
Buffer Zone sizes : 256MB/512MB/1GB		
SRAM for DFTL and WAFTL: 256KB For pure page-level FTL and FAST, we assume SRAM is enough to hold entire mapping table.		

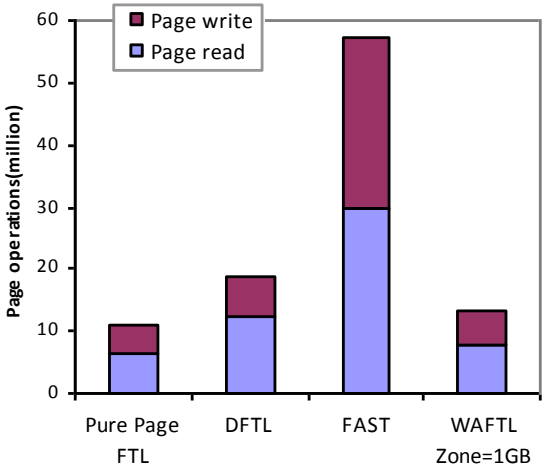
Workload Traces

Workloads	Avg. Req. Size(KB)	Write(%)	Seq.(%)	Avg. Req. Inter-arrive Time(ms)
Fin1	8.291986	35%	0.6%	8.189
Fin2	7.776166	17%	0.7%	11.081
Exchange	54.593399	74%	0.5%	1.3148
DevDiv91	36.168616	91%	0.9%	2.0435
DevDiv	88.983830	72%	1.5%	2.6882
Sequential Read	67.678195	19%	60%	49.857298
Sequential Write	67.552513	80%	70%	49.741739

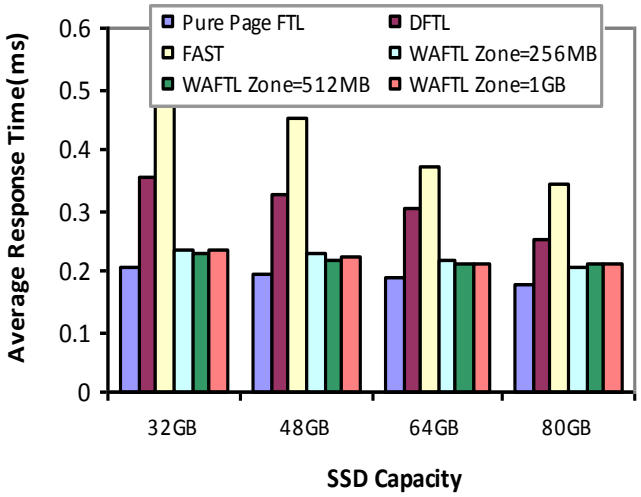
Evaluation

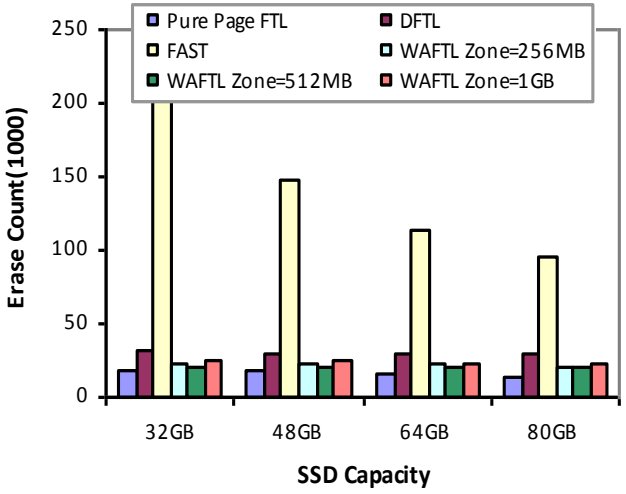


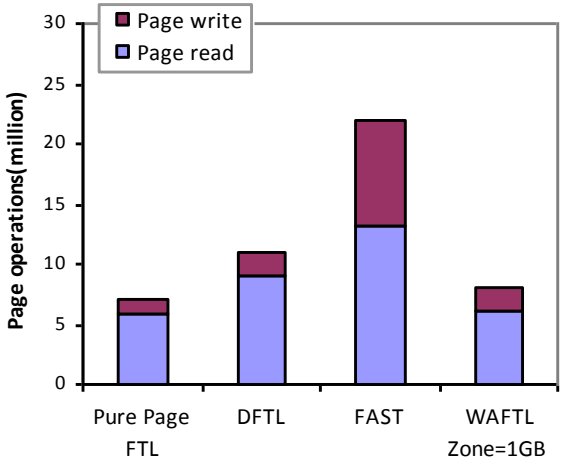




Fin1 Trace (Static Migration threshold=64pages,256KB)

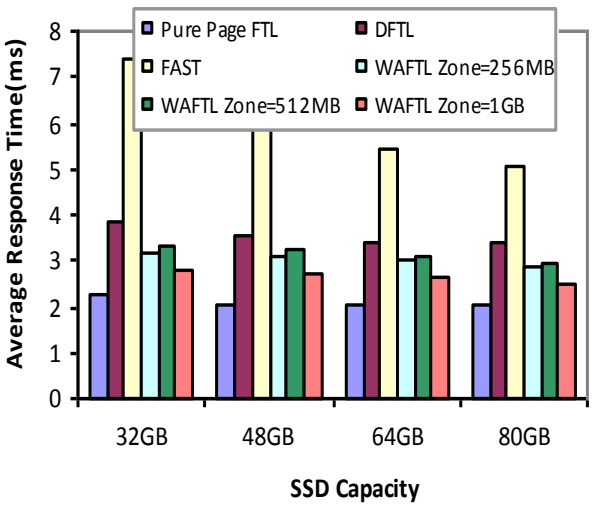


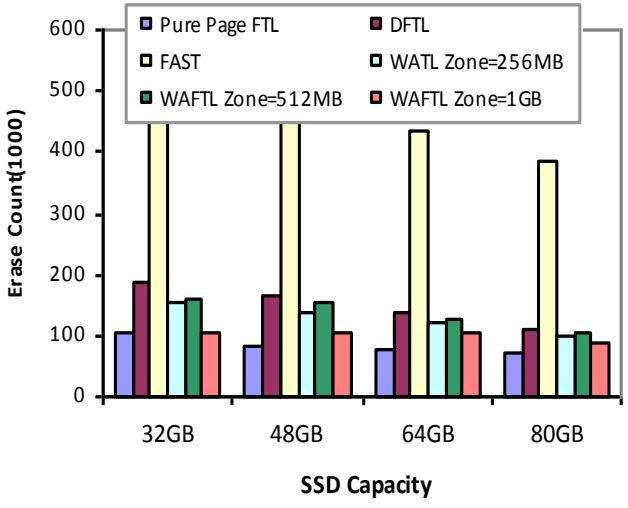


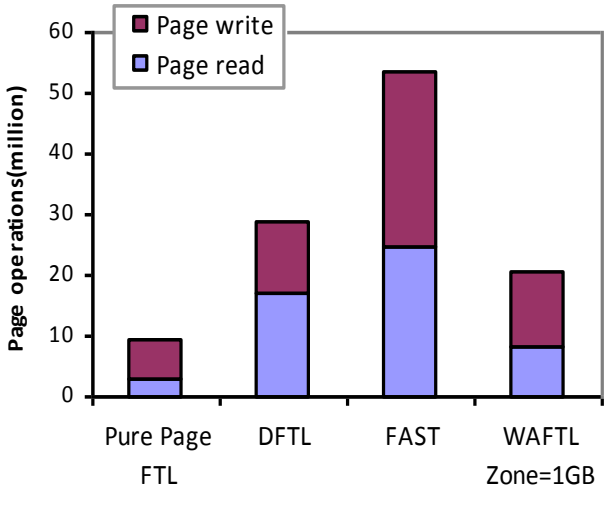


Fin2 Trace (Static Migration threshold=64pages,256KB)

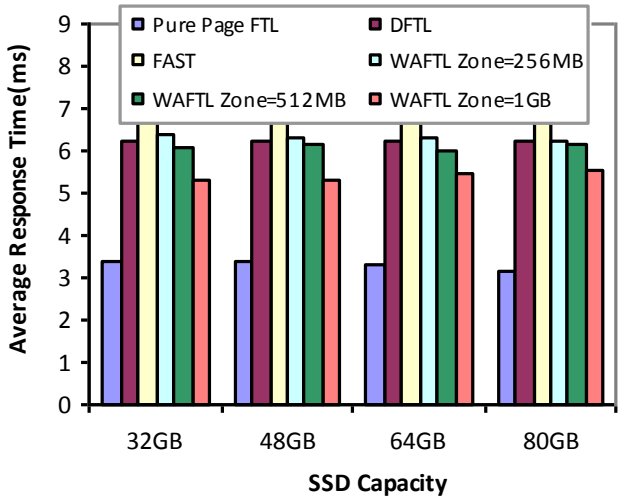
Evaluation

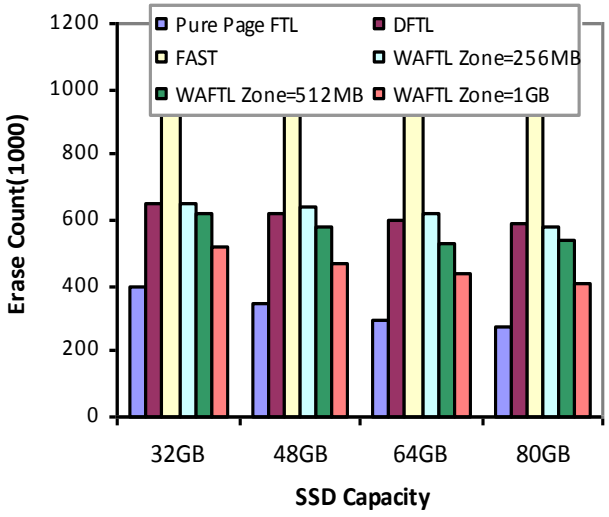


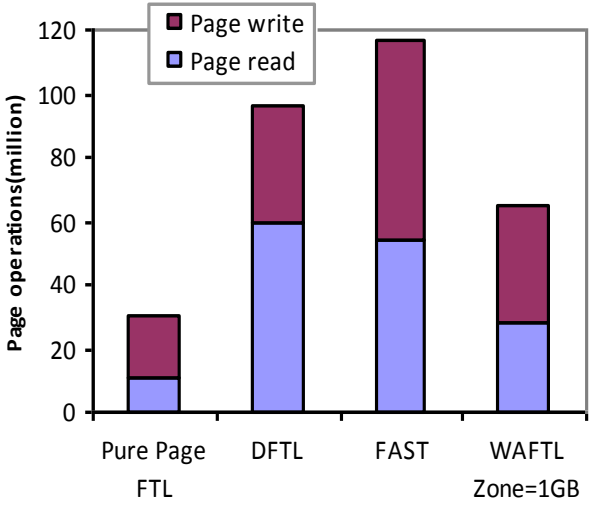




Exchange Trace (Static Migration threshold=64pages,256KB)

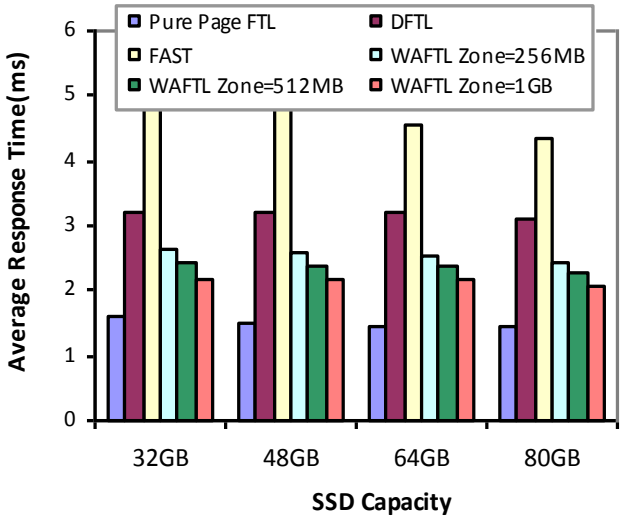


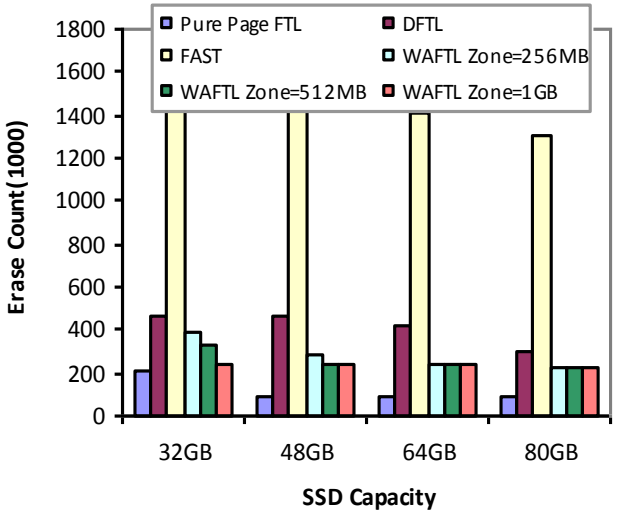


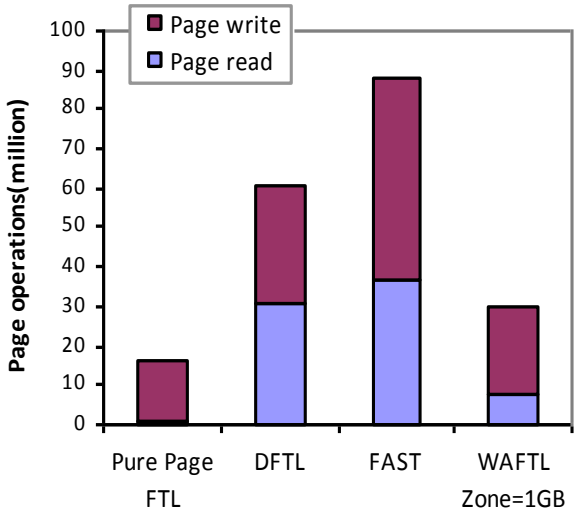


DevDiv Trace (Static Migration threshold=64pages,256KB)

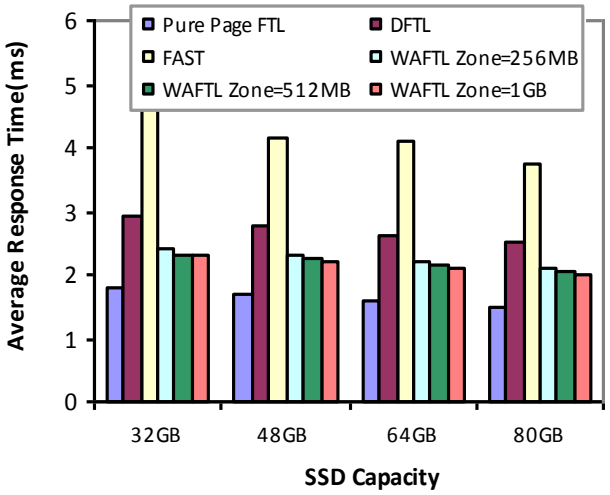
Evaluation

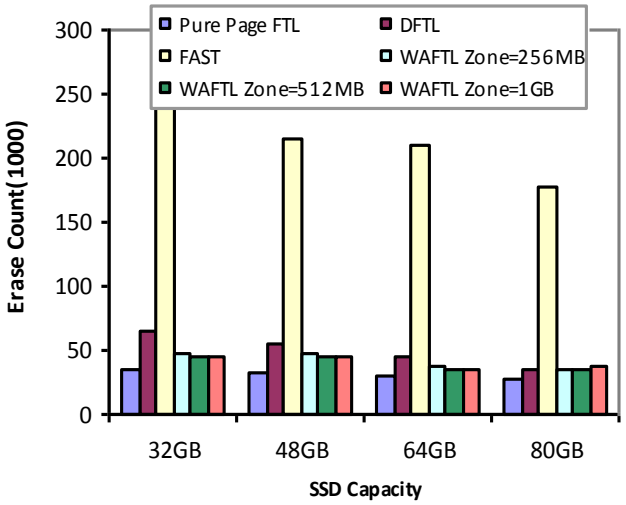


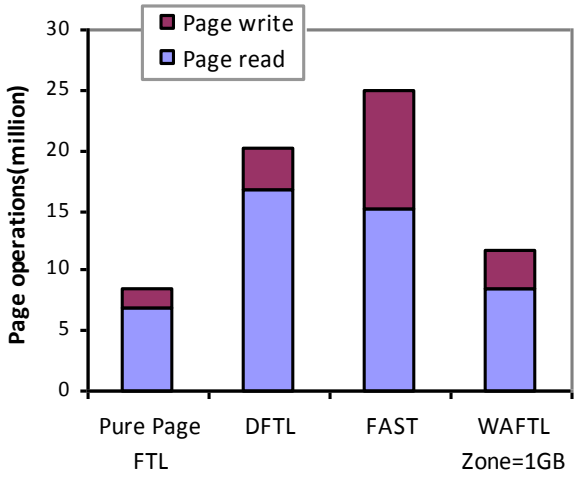




DevDiv91 Trace (Static Migration threshold=64pages,256KB)





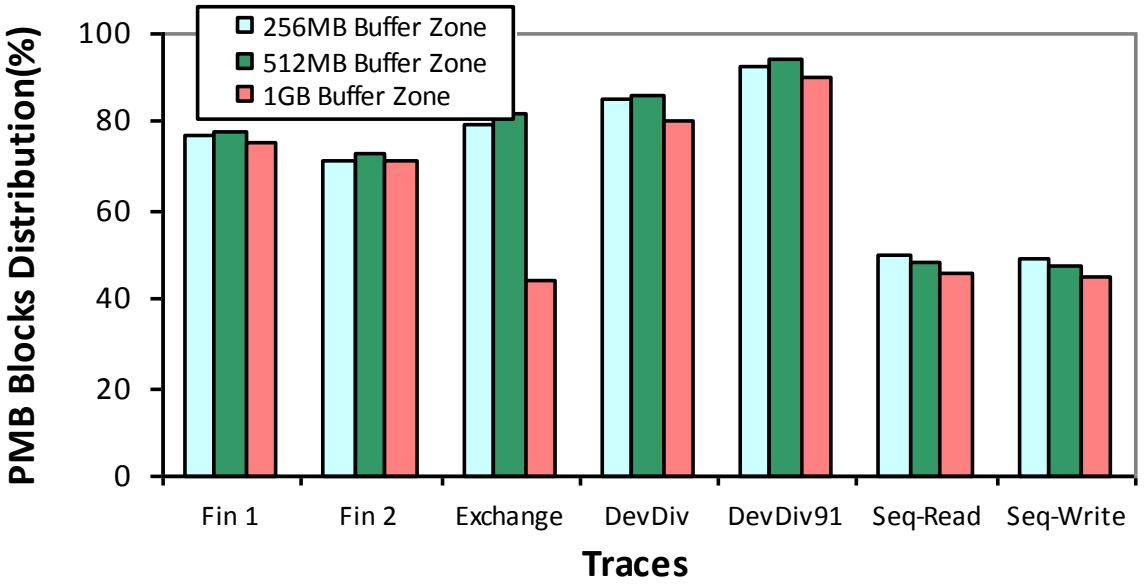


Seq_Read Trace (Static Migration threshold=64pages,256KB)

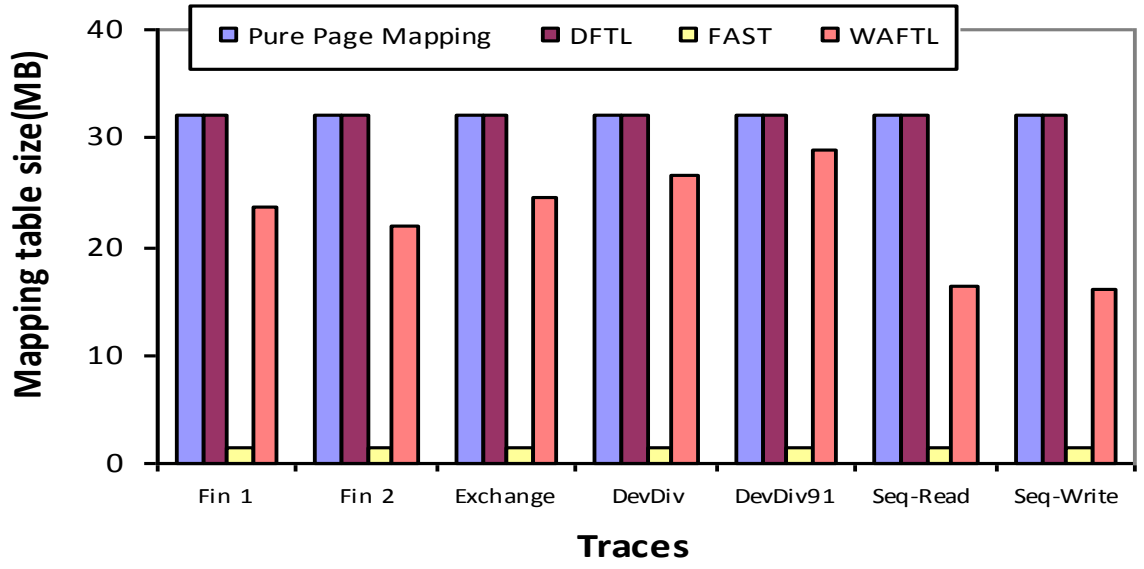
Evaluation

Workload adaptive

WAFTL is adaptive to workloads (32GB SSD)

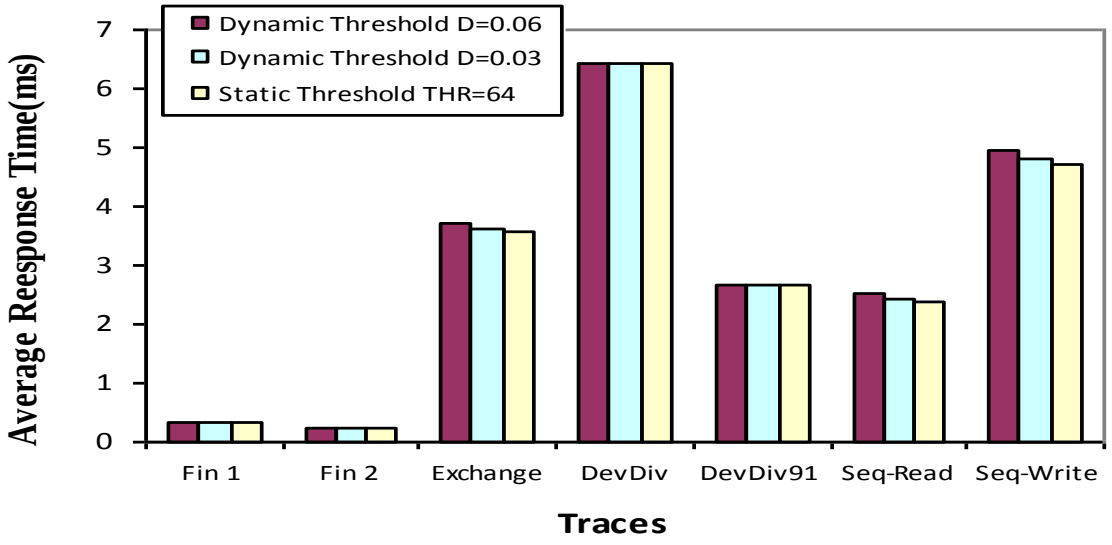


Mapping table sizes under different workloads (256MB buffer zone, 32GB SSD)

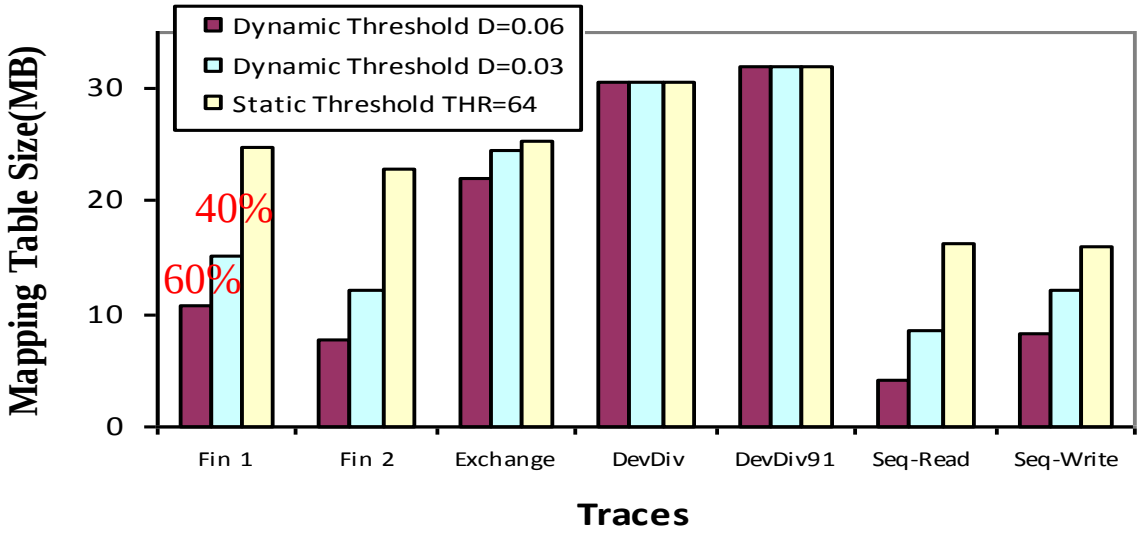


Evaluation

Effect of Threshold



Buffer Zone: 256MB
SSD: 32GB,
64Pages/Block



Buffer Zone:256MB
SSD: 32GB,
64Pages/Block

Conclusion

WAFTL explores

- either page-level or block-level address mapping for data blocks based on access patterns in order to balance the mapping efficiency of a block-based approach and the GC efficiency of a page-based approach.

- a small part of NAND flash space for the buffer area to log data sequentially until it is full and then migrate them into either a page mapping block or block mapping block, depending on the migration threshold.

- different scheme to determine the value of migration threshold

- thereby, improving performance while reducing the mapping table size at the same time.


Thanks & Question