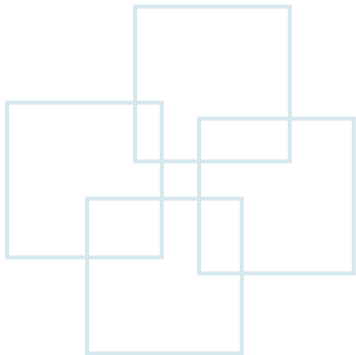


# **A Caching-Oriented FTL Design for Multi-Chipped Solid-State Disks**

Yuan-Hao Chang, Wei-Lun Lu, Po-Chun  
Huang, Lue-Jane Lee, and Tei-Wei Kuo





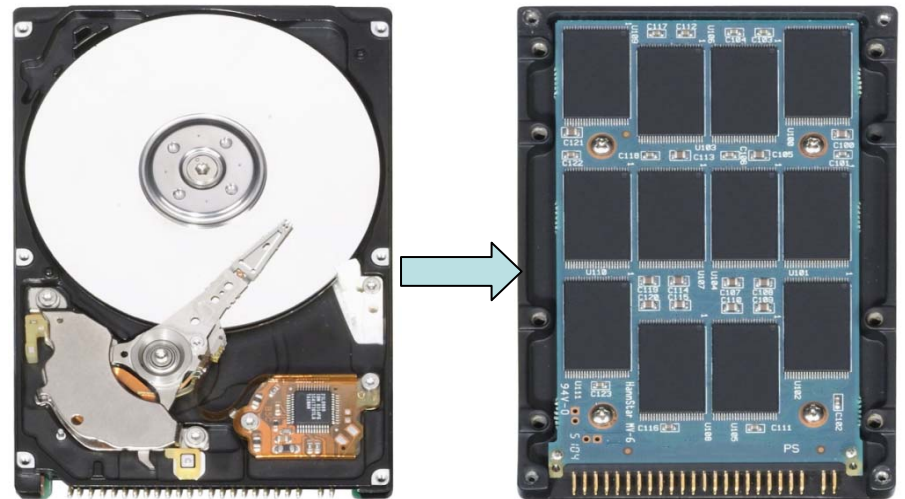
# Outline

- **Introduction**
- System Architecture
- A Multi-Chipped FTL with Caching Support
- Experiments
- Conclusion



# Introduction – Why Solid-State Disks

- Solid-State Disks (SSDs) are composed of multiple flash chips with a multi-channel architecture.
  - Shock resistance
  - Energy efficiency<sup>1</sup>
  - Small size (form factor)
  - High performance
- SSDs are replacing hard drives in some application domains.
  - E.g., Desktop PCs

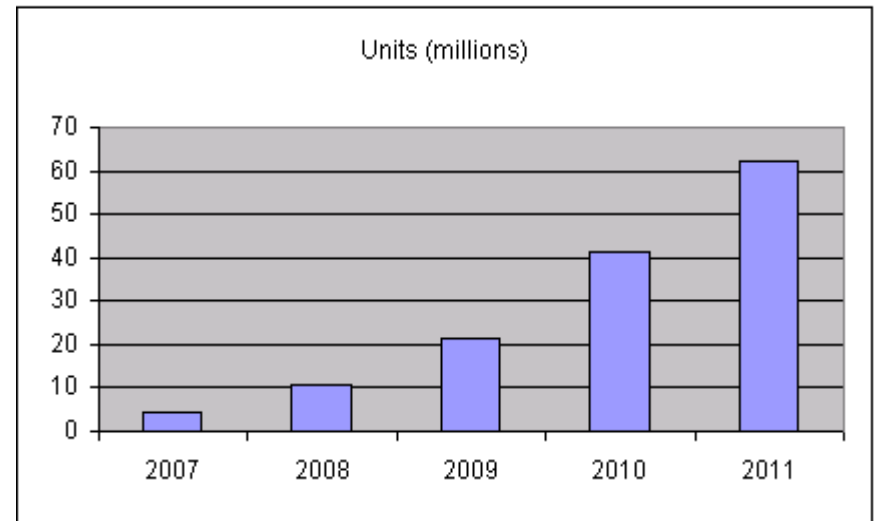
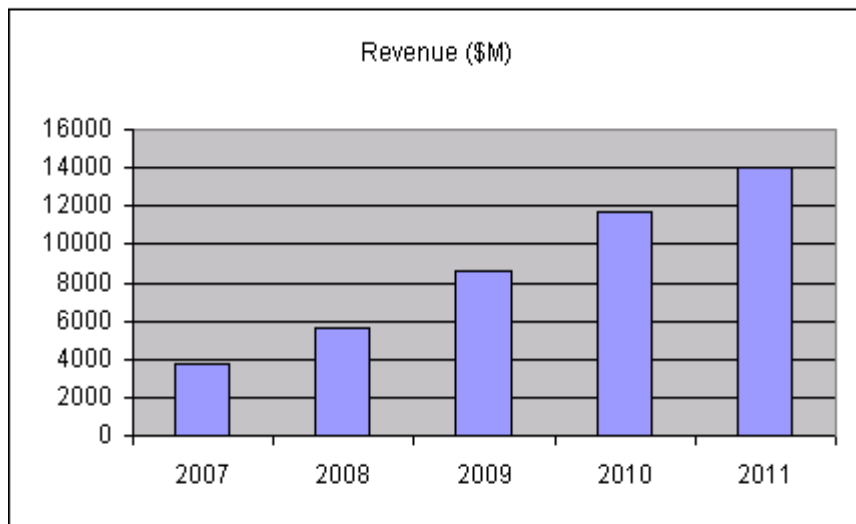


[1] Some SSDs equip a large SDRAM that might be energy efficient.



# SSD Market Forecast

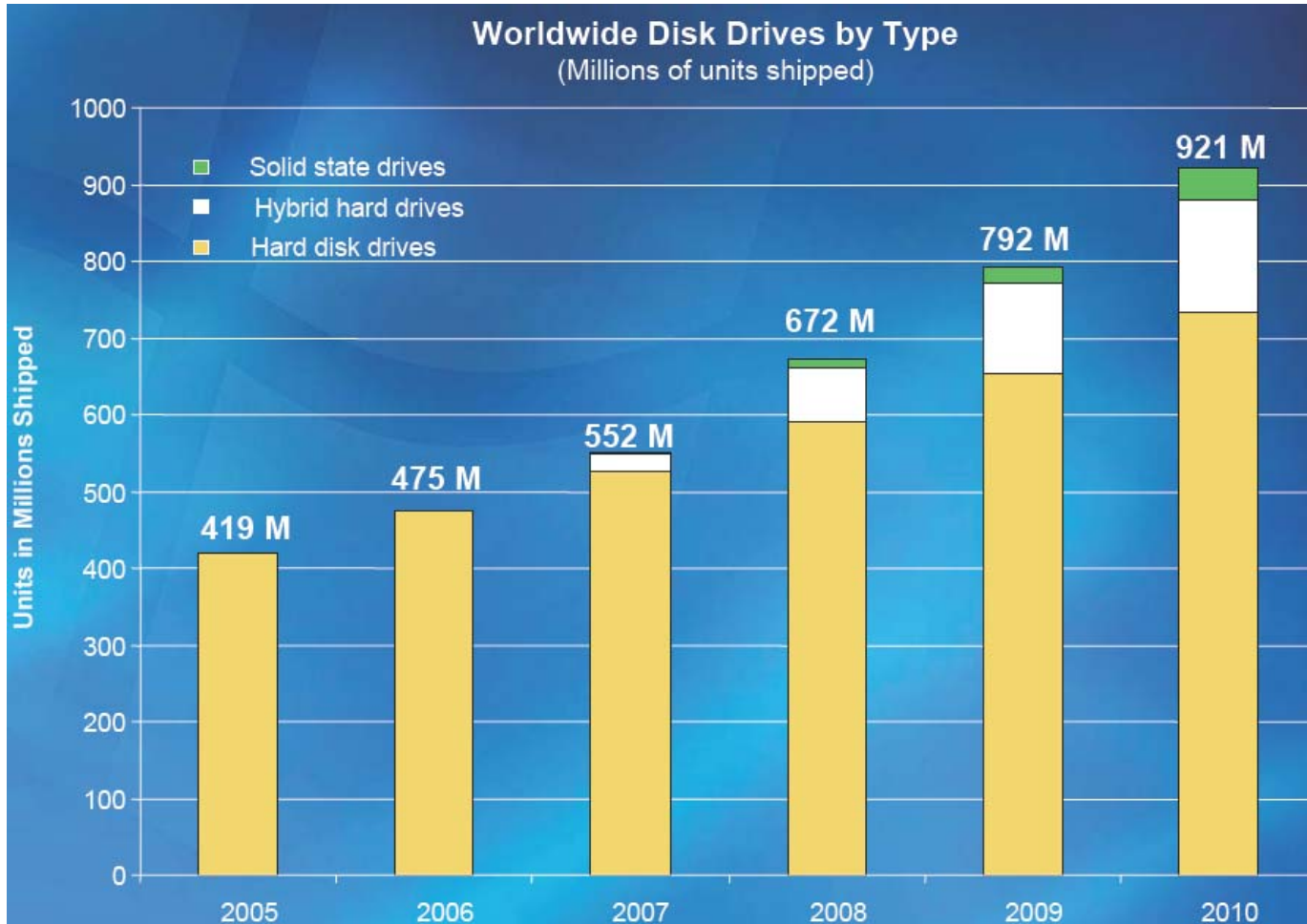
	2007	2008	2009	2010	2011
Revenue (\$M)	\$3,800	\$5,600	\$8,600	\$11,700	\$14,000
Units (millions)	4.3	10.8	21.6	41.3	62.2



Source: Web-Foot Research 2006 (from Microsoft's WinHec 2007 presentation)



# SSD vs. Hard Drives

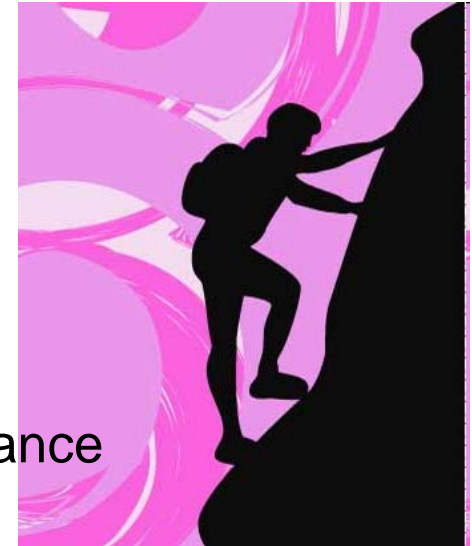


Source: Web-Foot Research 2006 (from Microsoft's WinHec 2007 presentation)



# SSD Challenges

- Performance
  - Bus utilization
  - Chip read/write performance
    - Lower-cost flash memory with lower performance
- Management scalability
  - The number of chips
  - The number of channels
  - The capacity of flash storage devices





# Outline

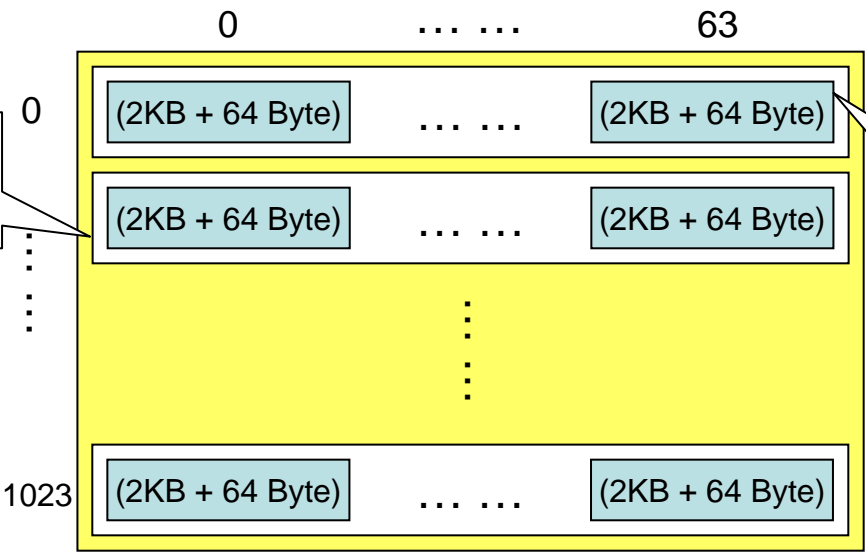
- Introduction
- **System Architecture**
- A Multi-Chipped FTL with Caching Support
- Experiments
- Conclusion



# Characteristics of Flash Memory

- Out-of-place update
- Bulk erasing

**Block:** basic erase-operation unit.



**Page:** basic write-operation unit.

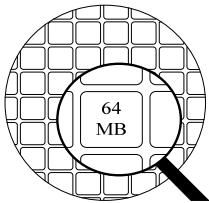
## Garbage Collection





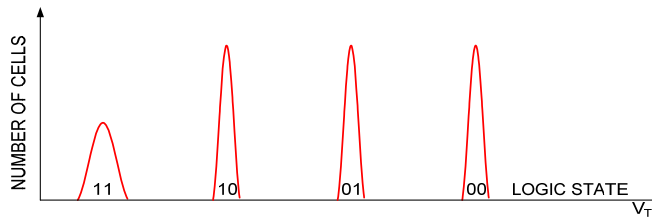
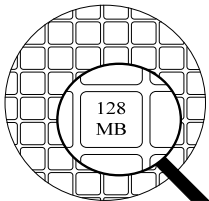
# SLC vs. MLC

SLC Flash



a) BILEVEL (1 BIT/CELL)

MLC Flash



b) MULTILEVEL (2 BIT/CELL)

- MLC write constraints
  - Partial-page programming prohibition
  - Sequential page writes in a block

Type & Part Number	SLC	MLC <sub>x2</sub>
Price (US dollars/GB, 2009 Q1)	5.65	1.57
Serial Access	25ns	25ns
Random Read	20μs	60μs
Write/Program	200μs	800μs
Erase	1.5ms	1.5ms
Partial Page Write/Program	Yes (4)	No (1)
Random Page Write	Yes	No



# Flash Translation Layer

- Purpose
  - Emulate the underlying flash media as a block device.
  - Translate logical block addresses (LBAs) to physical block addresses (PBAs)
- Design issue
  - Main-memory requirement
  - Service time on each read/write request
    - Address translation time
    - Extra read, write, and erase operations
- Popular existing flash-translation-layer designs
  - FTL (page-level address translation)
  - BL (block-level address translation)



# BL (Block-Level Address Translation)

- BL adopts a block-level translation mechanism.
  - A logical address under BL is divided into a virtual block address and a block offset.

**1st write: LBA (20, 2)**

$$\text{VBA} = 20 / 8 = 2,$$

$$\text{Offset} = 20 \% 8 = 4$$

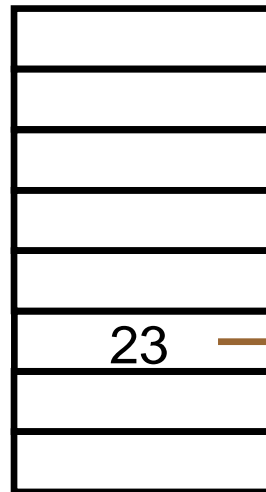
**2nd write: LBA(22, 2)**

$$\text{VBA} = 22 / 8 = 2,$$

$$\text{Offset} = 22 \% 8 = 6$$

VBA

7  
6  
5  
4  
3  
2  
1  
0

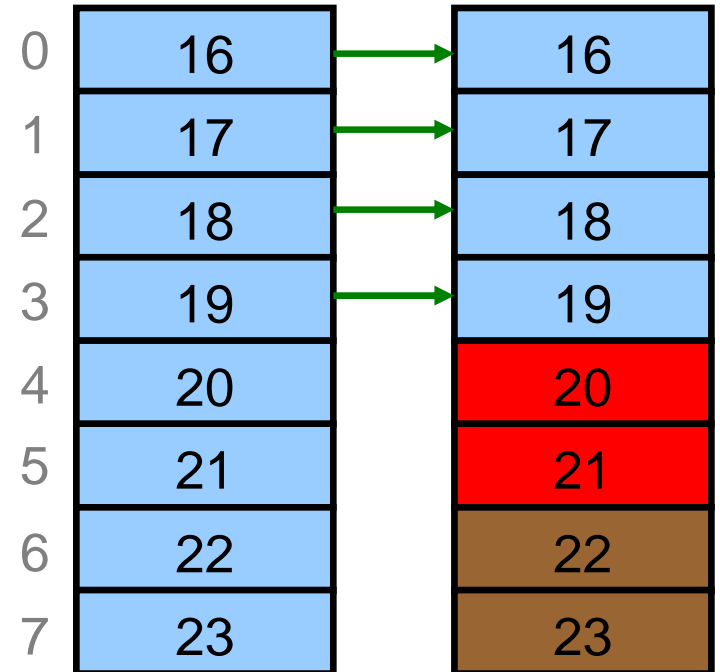


33

**Address Translation Table  
(in main-memory)**

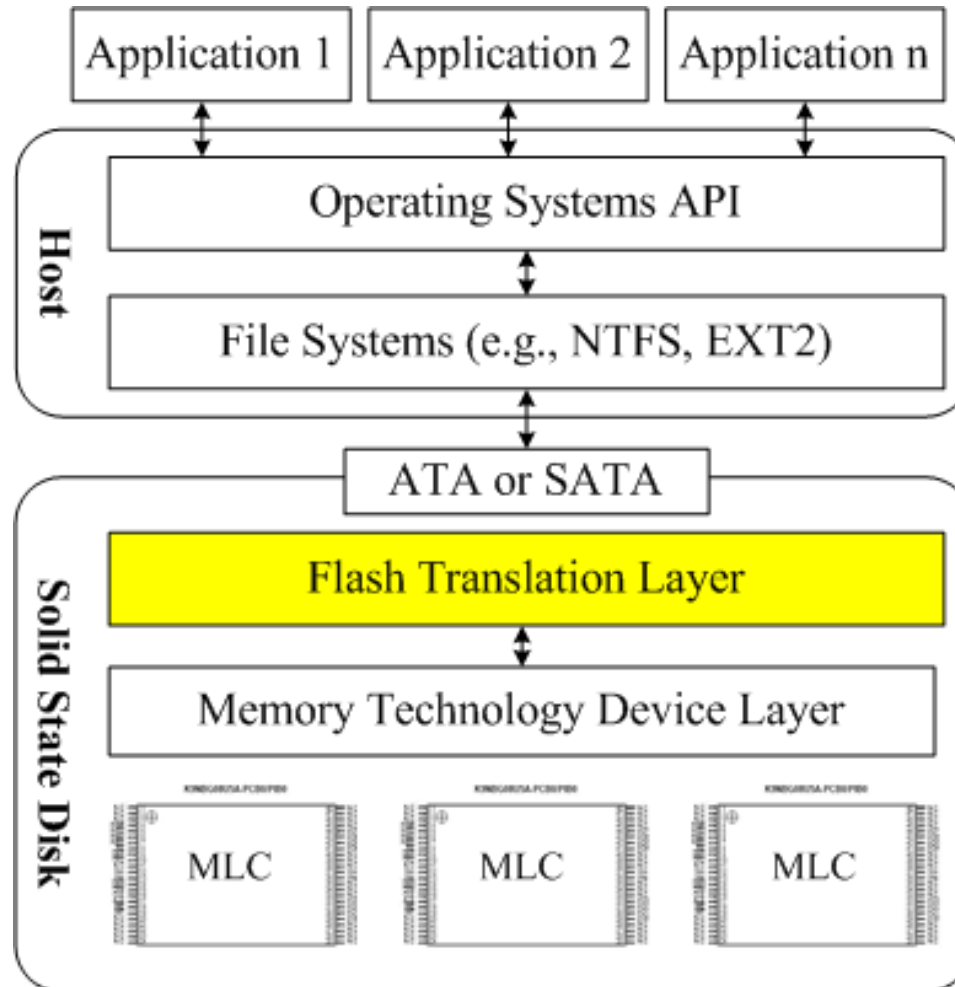
Original block  
Address = 23

Free block  
Address = 33



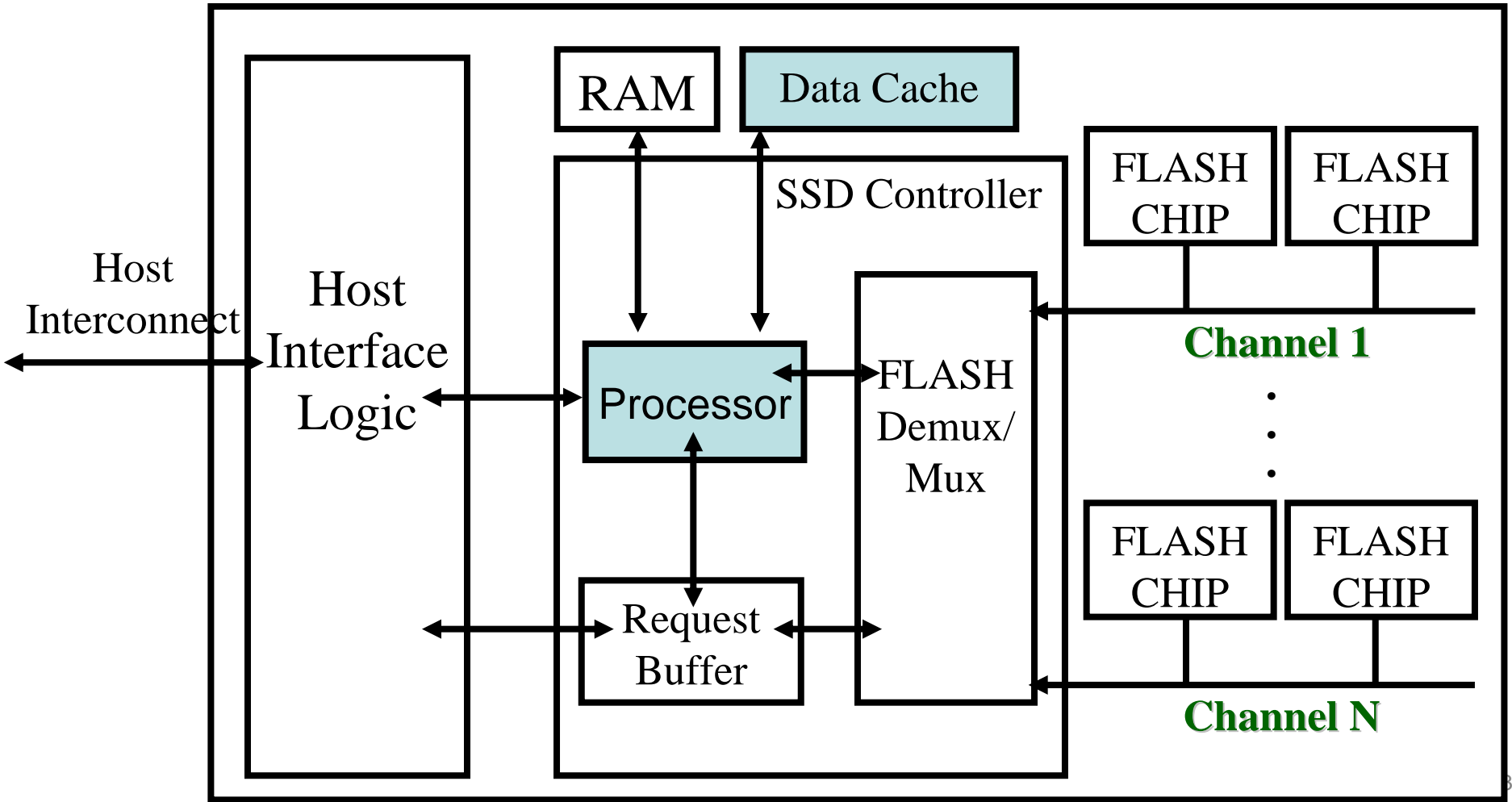


# System Architecture





# System Architecture – SSD Block Diagram



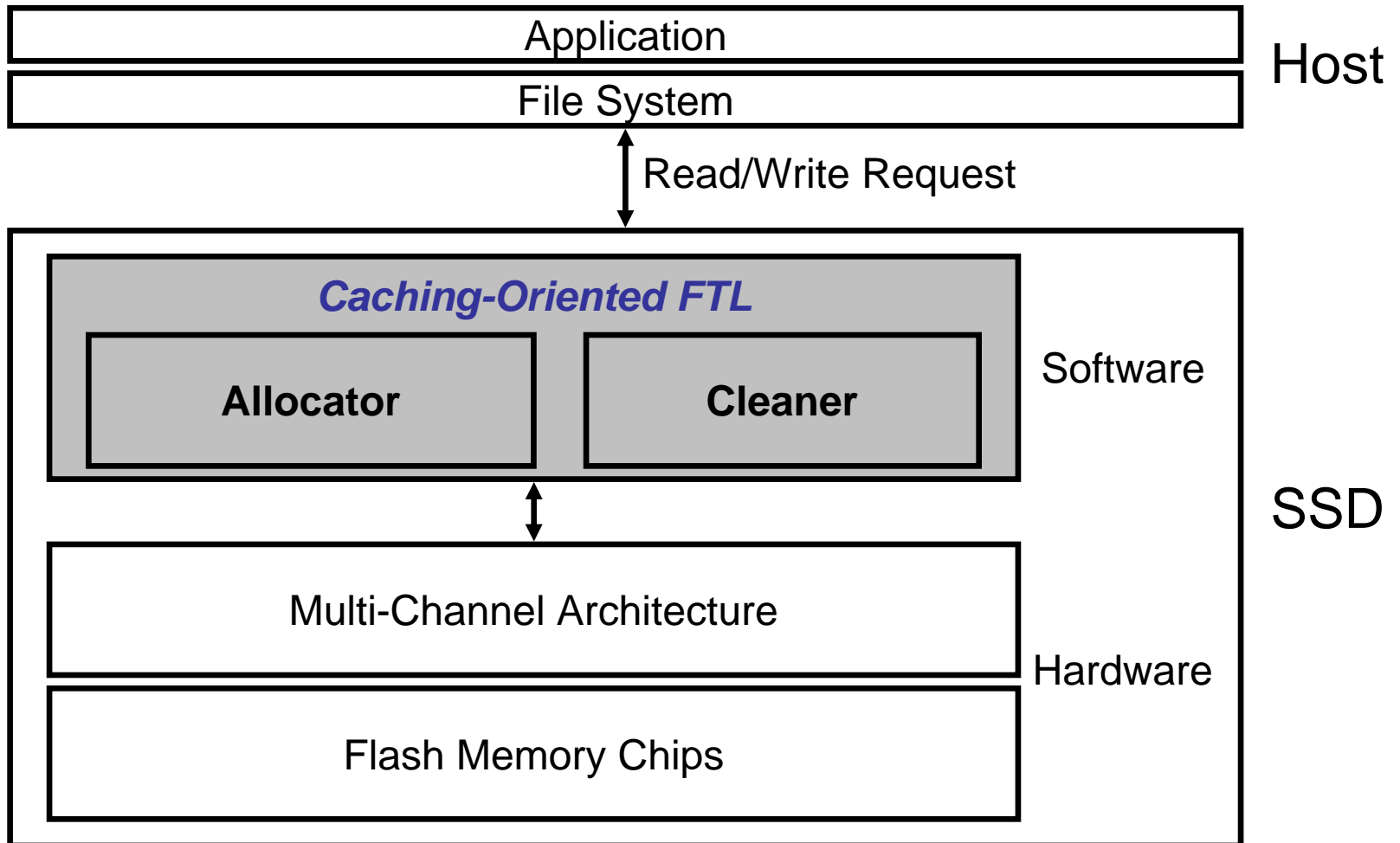


# Outline

- Introduction
- System Architecture
- **A Multi-Chipped FTL with Caching Support**
- Experiments
- Conclusion



# Components of Caching-Oriented FTL



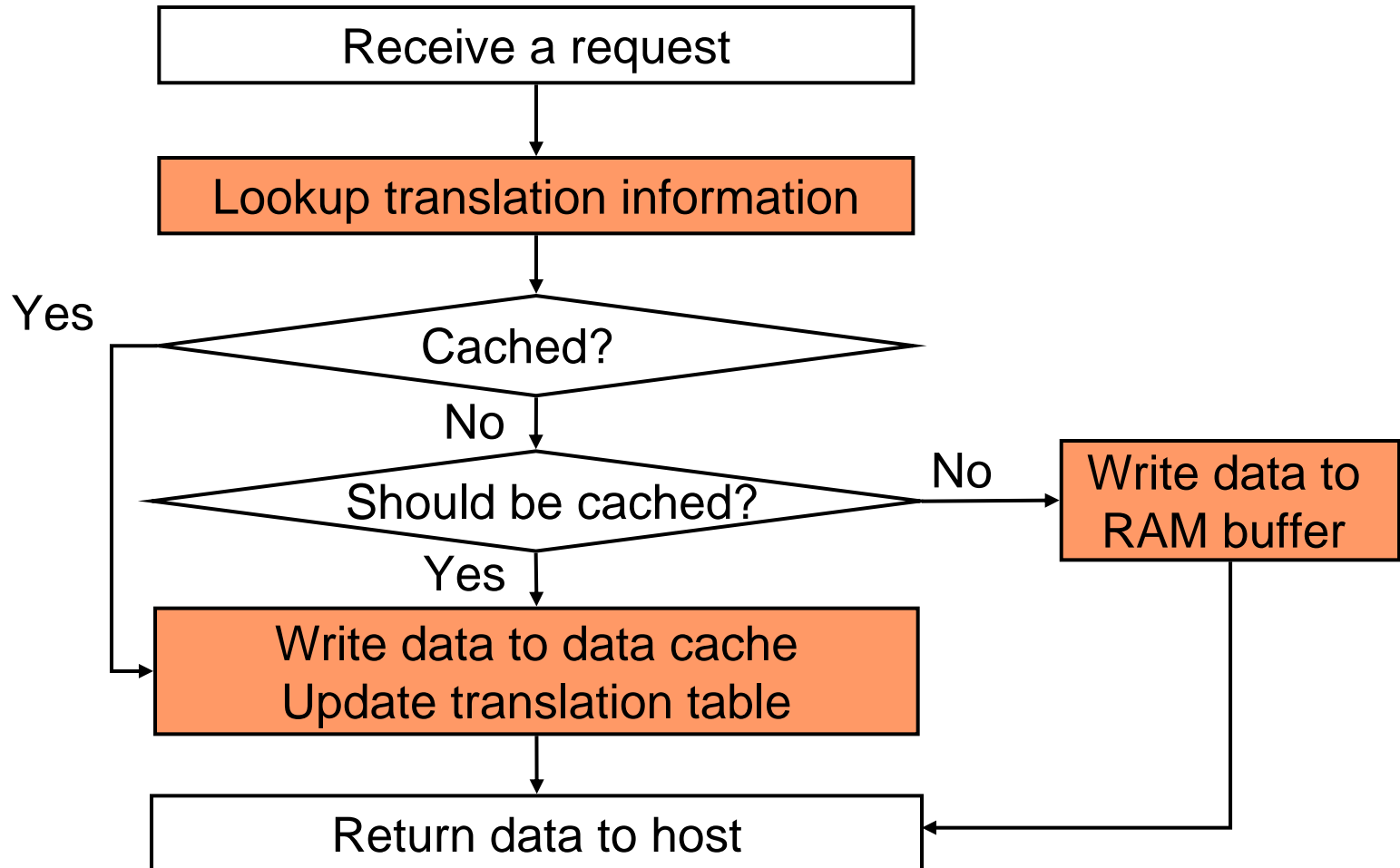


# Caching-Oriented FTL

- Allocator
  - Hybrid address translation architecture
  - Heap-like index structure
  - Adaptive caching policy
- Cleaner
  - Garbage identifying structure
  - Free space manager
- Objective
  - Improve the performance of read/write requests
    - Explore a high degree of parallelism for flash-memory accesses
    - Enhance the hit ratio of the data cache
  - Reduce overheads on the management of address translation



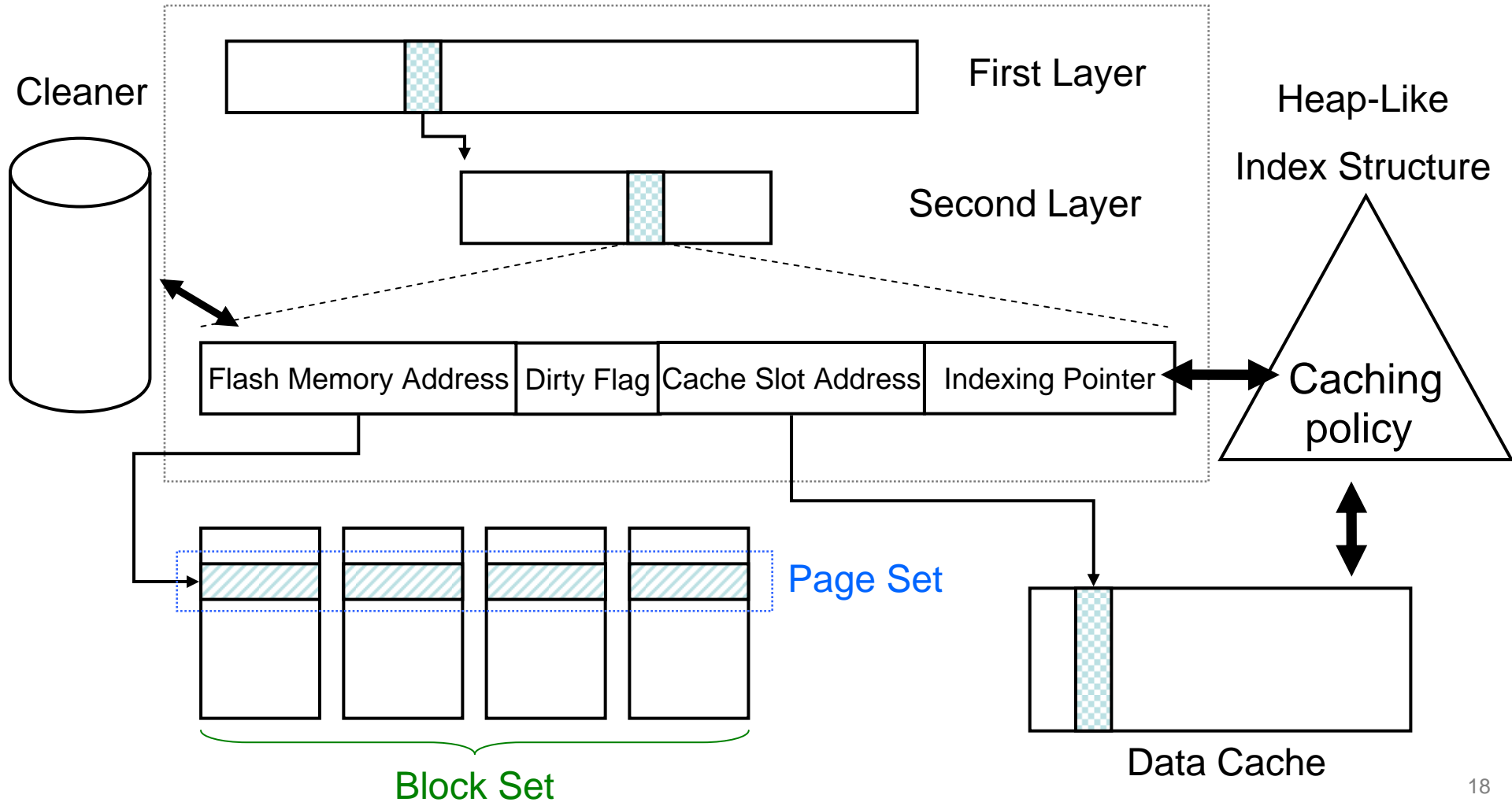
# Write Scenario





# Architecture of Caching-Oriented FTL

Hybrid Address Translation Architecture



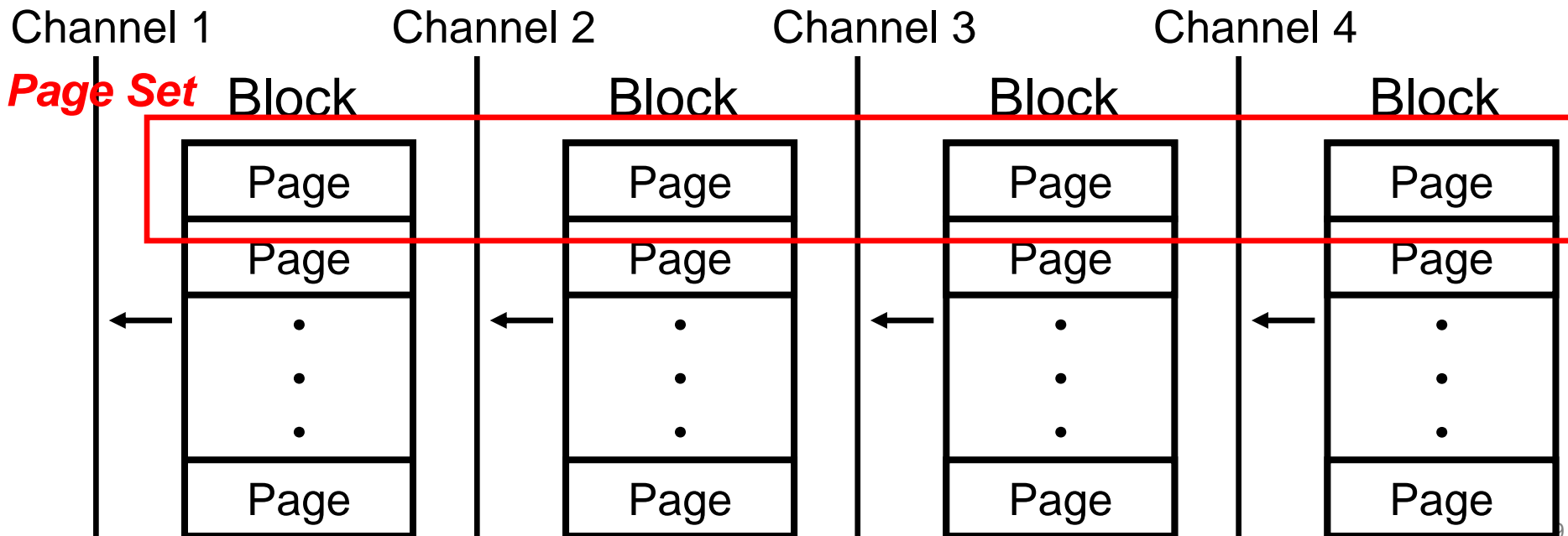


# Allocator

## – Translation and Access Unit

- Purpose

- High parallelism degree for flash-memory accesses
- Reasonable size for most requests



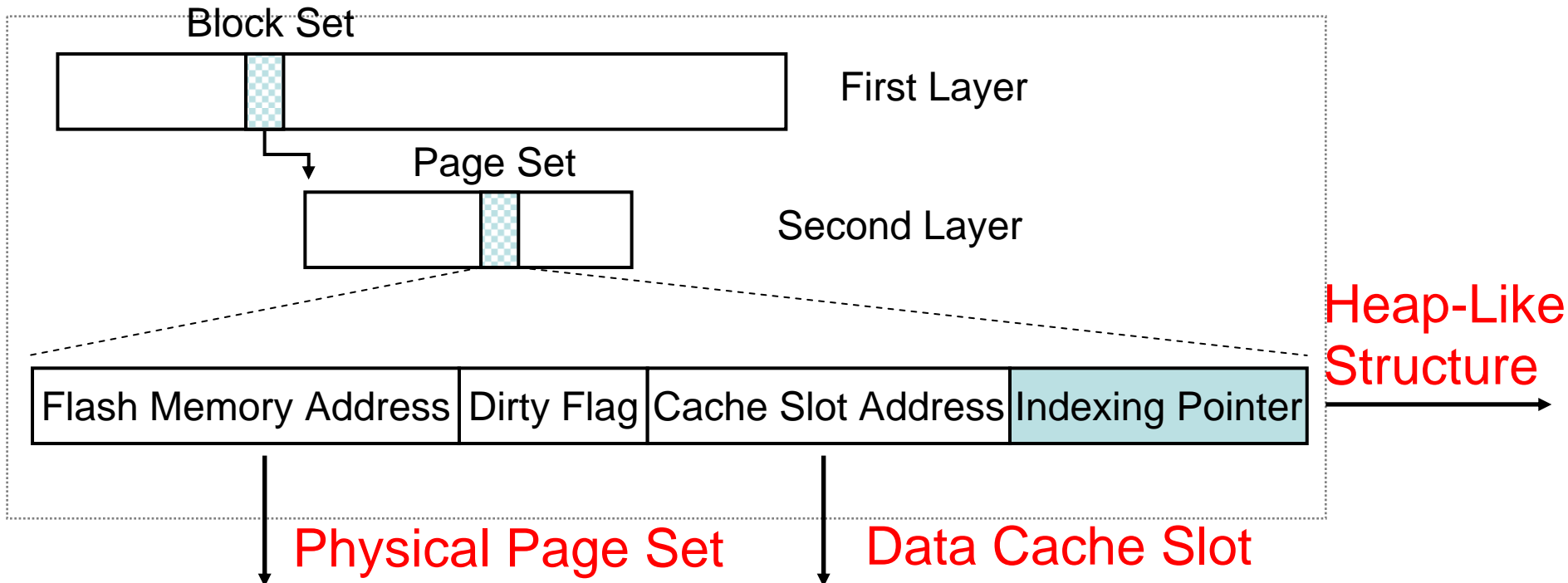


# Allocator

## – Hybrid Address Translation Architecture

- A two-layer translation table

- Translate an LBA to a flash **page address** or a **cache slot address**



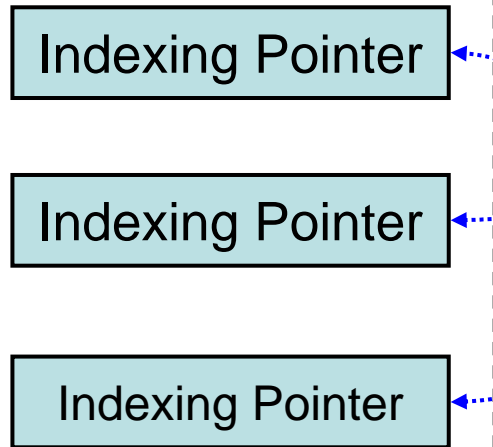


# Allocator

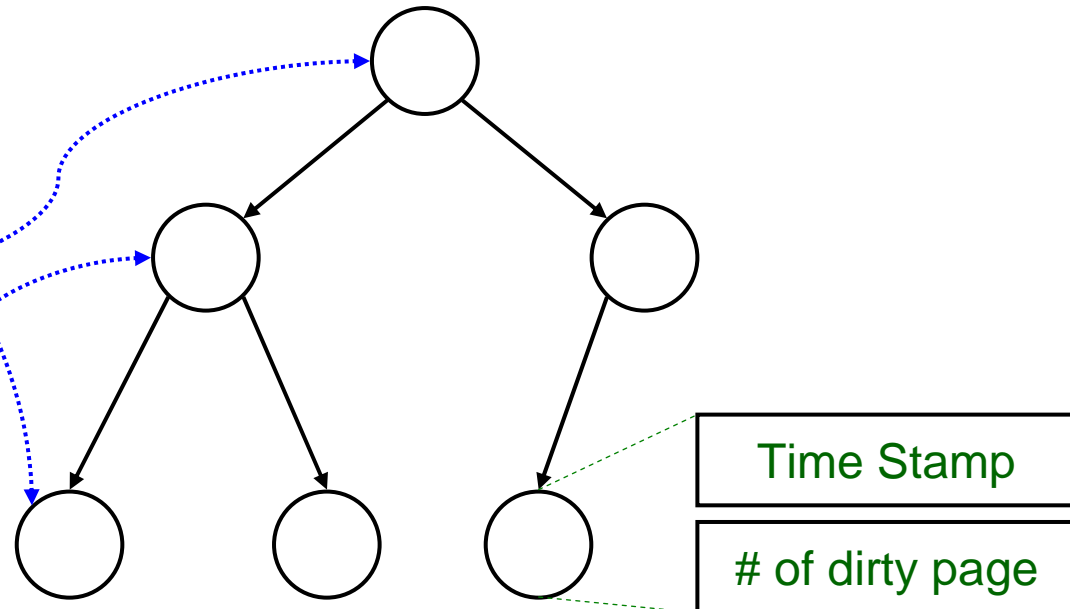
## – Heap-Like Index Structure

Maintain the information for the caching policy

### *Translation Table*



### *Index Structure*





# Allocator

## – An Adaptive Caching Policy

- Sequential Access

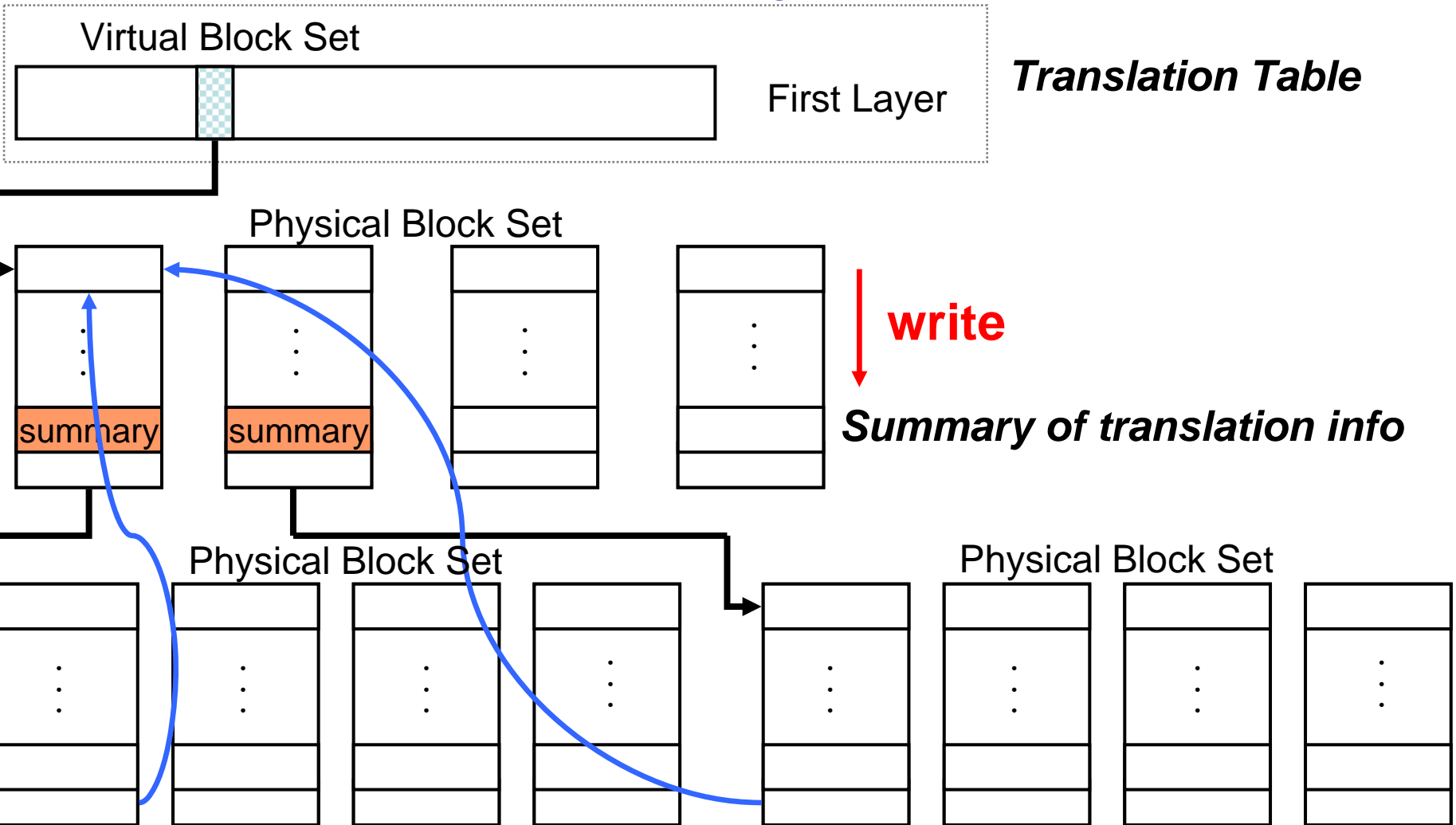
- Size of accessed data in a request is larger than a page set
- Frequently-read data is cached

- Non-Sequential Access

- Last referenced time
- Amount of modified(/dirty) pages in a page set



# Virtual Block Set vs. Physical Block Set





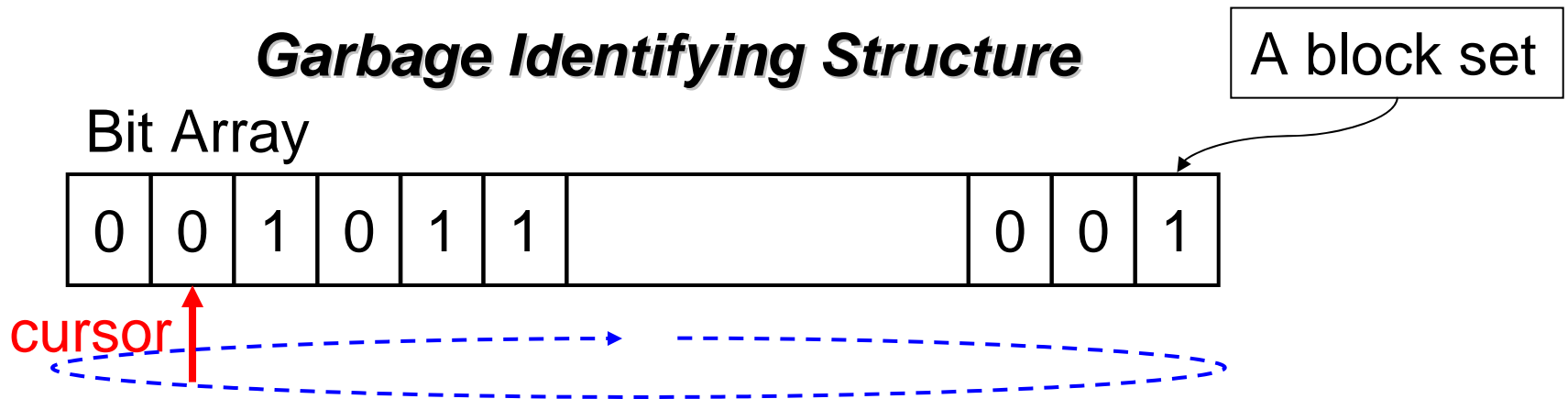
# Cleaner

- Purpose

- Merging of physical block sets that are mapped to the same virtual block

- Activation condition

- Number of physical block sets is lower than a predetermined threshold





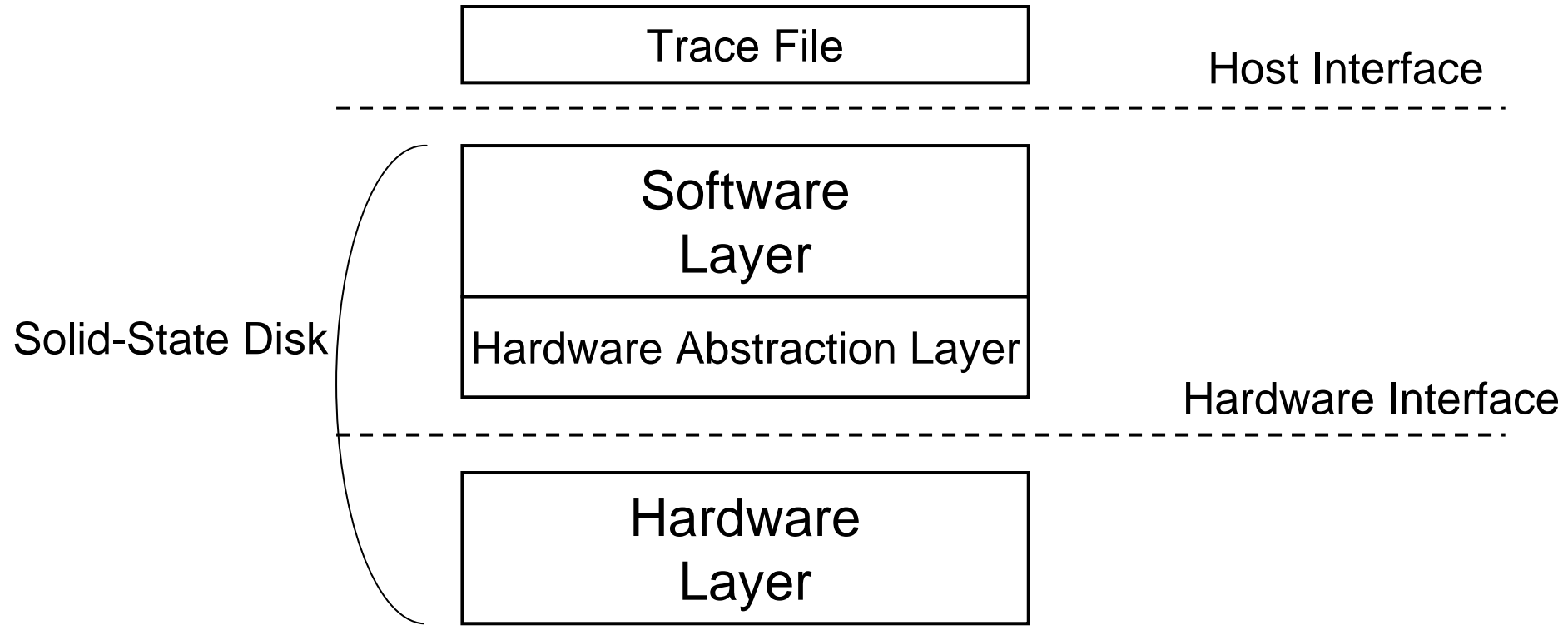
# Outline

- Introduction
- System Architecture
- A Multi-Chipped FTL with Caching Support
- **Experiments**
- Conclusion



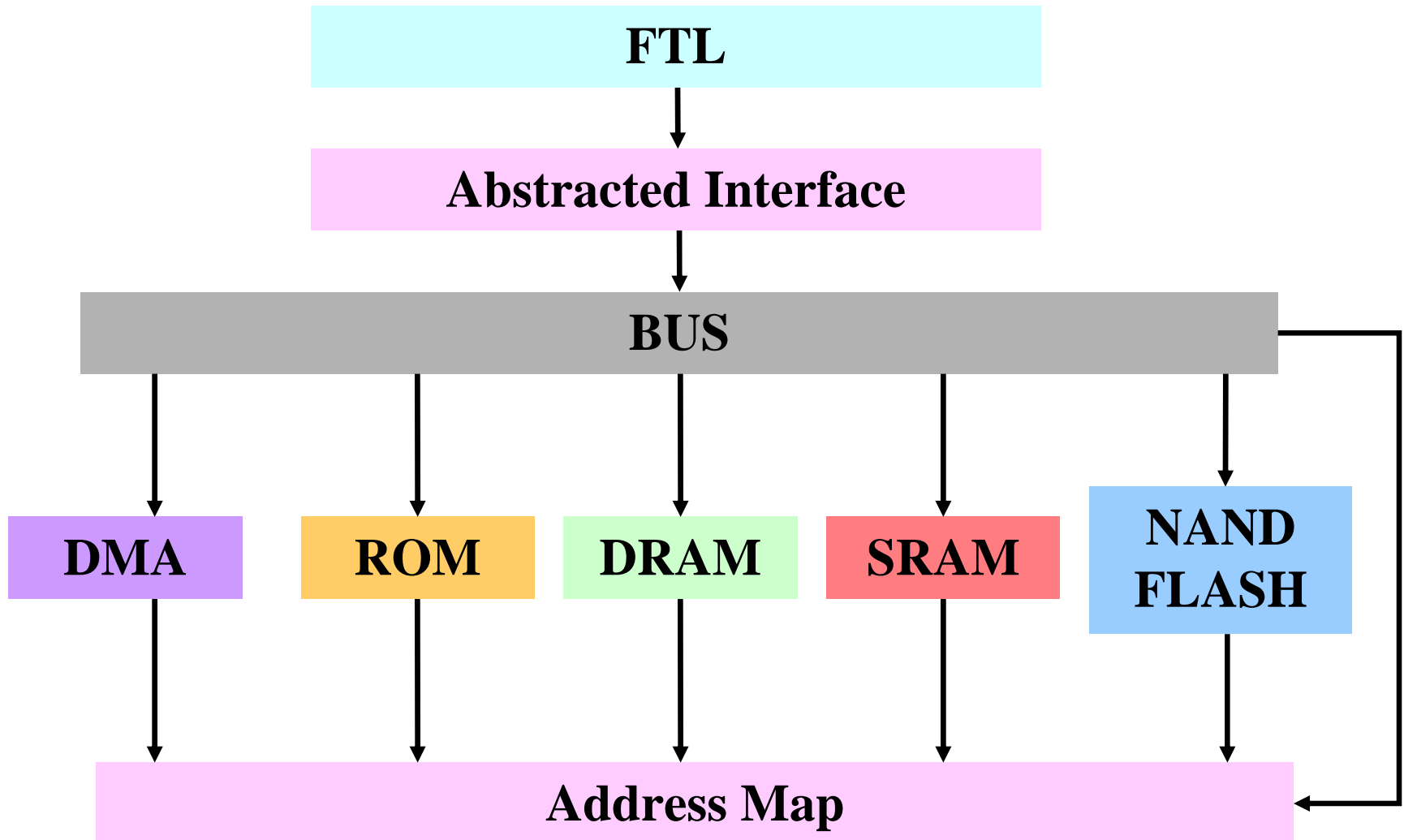
# Solid-State Disk Simulator

- Simulator Architecture





# FTL and Hardware Modules





# Workload Analysis

**95~98%**

	Trace 1	Trace 2	Trace 3	Trace 4	Trace 5
< 4KB	61.0937%	69.8761%	69.6798%	66.7316%	64.8221%
4~8KB	3.6870%	4.1295%	2.8431%	4.1196%	4.1471%
8~16KB	5.1224%	5.3483%	3.9492%	5.2693%	5.5429%
16~32KB	7.6706%	6.8395%	5.4788%	7.0201%	8.8467%
32~64KB	21.2955%	9.1417%	15.9027%	13.0844%	15.5289%
64~128KB	0.1997%	0.3361%	0.3051%	0.2438%	0.6276%
128~256KB	0.0809%	2.8223%	1.2196%	2.7138%	0.0890%
256~512KB	0.8292%	1.4917%	0.6134%	0.8032%	0.3783%
512~1024KB	0.0115%	0.0096%	0.0044%	0.0080%	0.0104%
> 1MB	0.0095%	0.0052%	0.0038%	0.0062%	0.0070%

*Applications: Media Player, Avira AntiVir, FileZilla Server, BitTorrent, Disk Defragmenter*



# Caching-Oriented FTL vs. BL

- Access unit:
  - Caching-oriented FTL
    - *Page size x channel = 4K x 4 = 16 KB*
  - BL
    - *Block size = 4K x 128 = 512 KB*
- Write performance
  - The size of most read/write requests is no more than **64KB**
    - Caching-oriented FTL: 1~4 write operations
    - BL: 128 write operations (for MLC flash)



# Outline

- Introduction
- System Architecture
- A Multi-Chipped FTL with Caching Support
- Experiments
- **Conclusion**



# Conclusion

- We propose a multi-chipped FTL design with caching support for solid-state disks.
- A SystemC-based SSD simulator was implemented to evaluate the performance of different FTL designs with different hardware configurations.
- Future work
  - Explore the crash recoverability for the adaptive caching policy
  - Exploit the possibility to enhance the caching performance with limited cache size



**Thanks for your listening!**

**Q&A**