Solid-State Drives (SSDs)

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

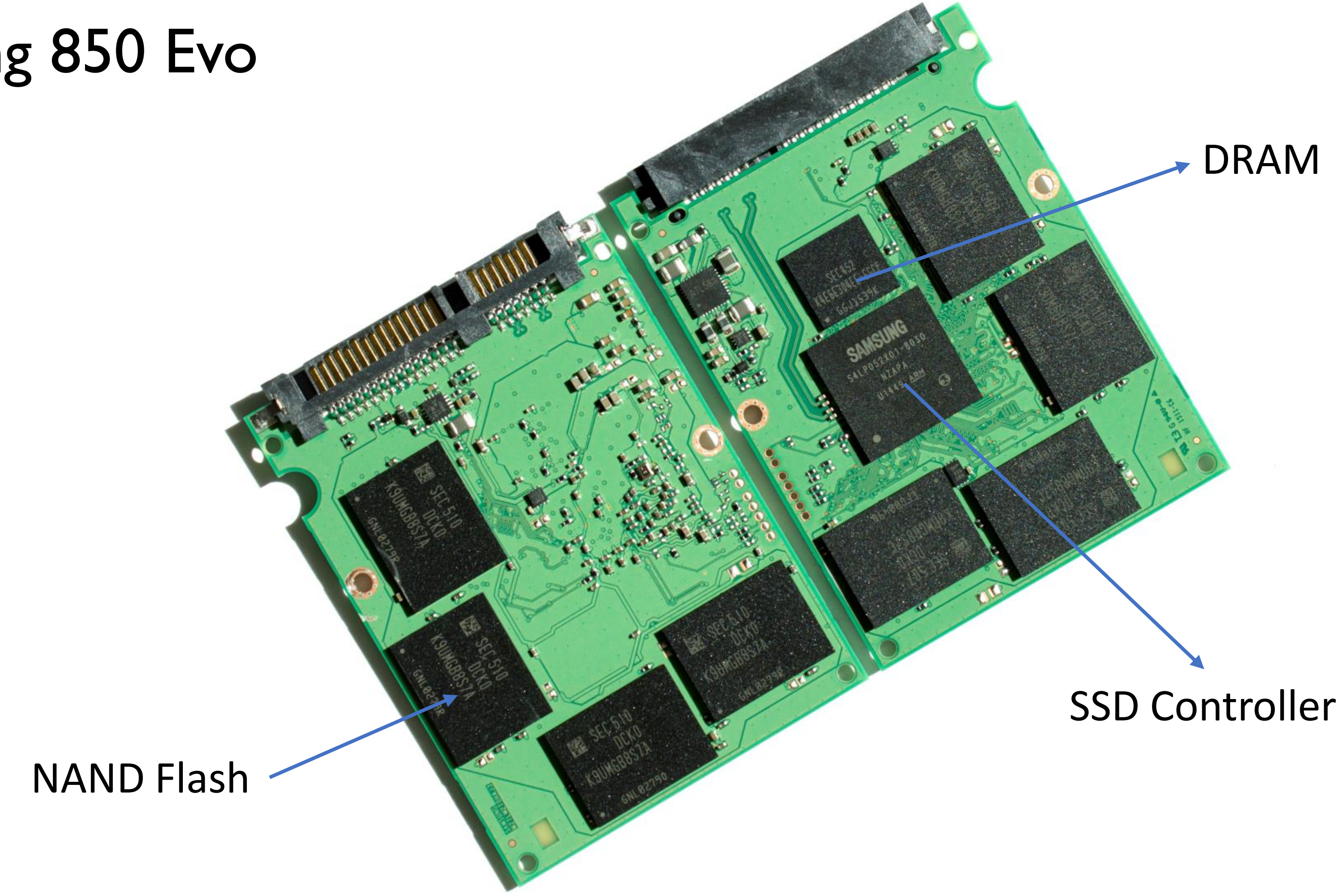
Systems Software &
Architecture Lab.

Seoul National University

Spring 2019

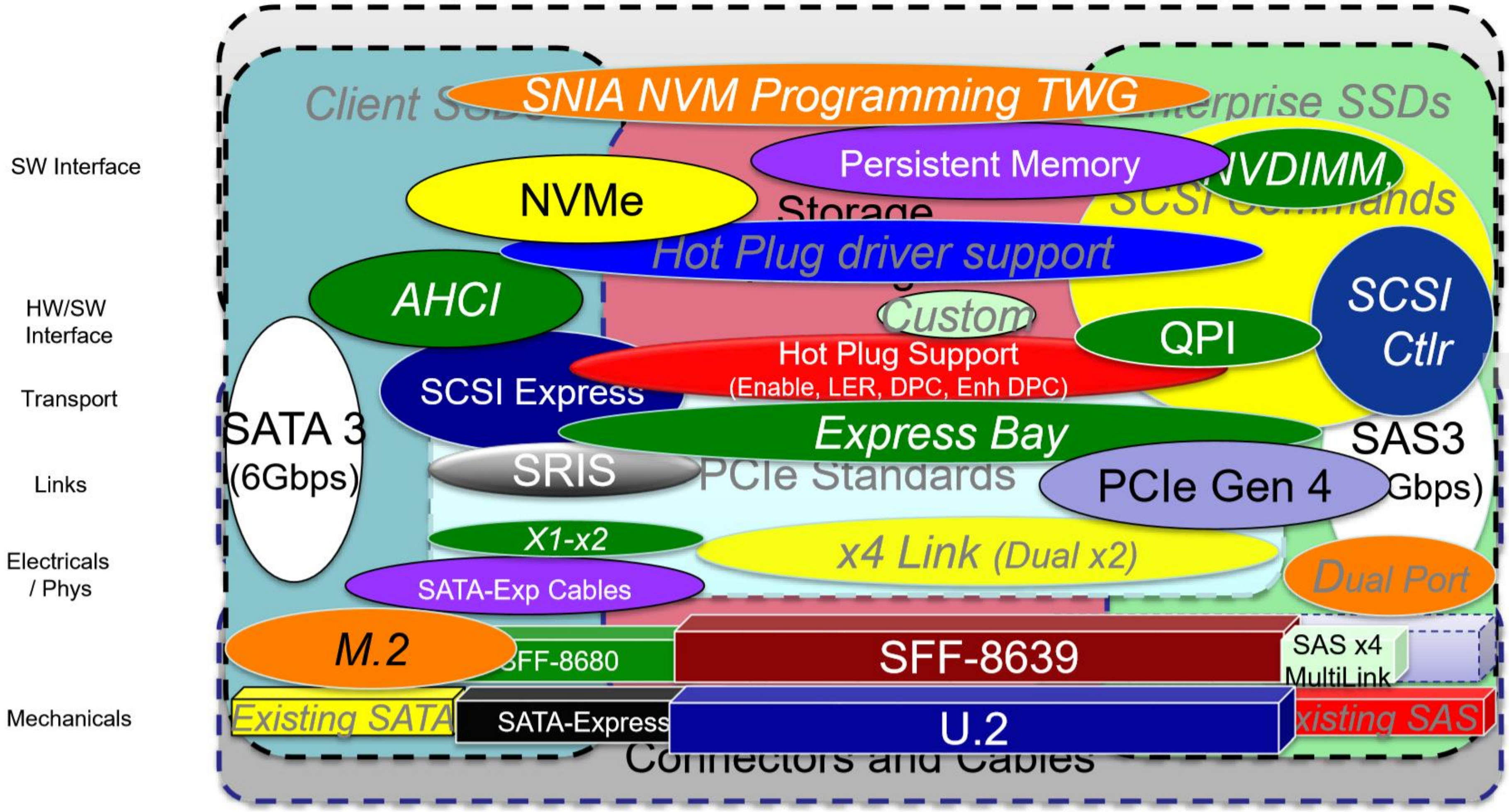
Anatomy of an SSD

- Samsung 850 Evo



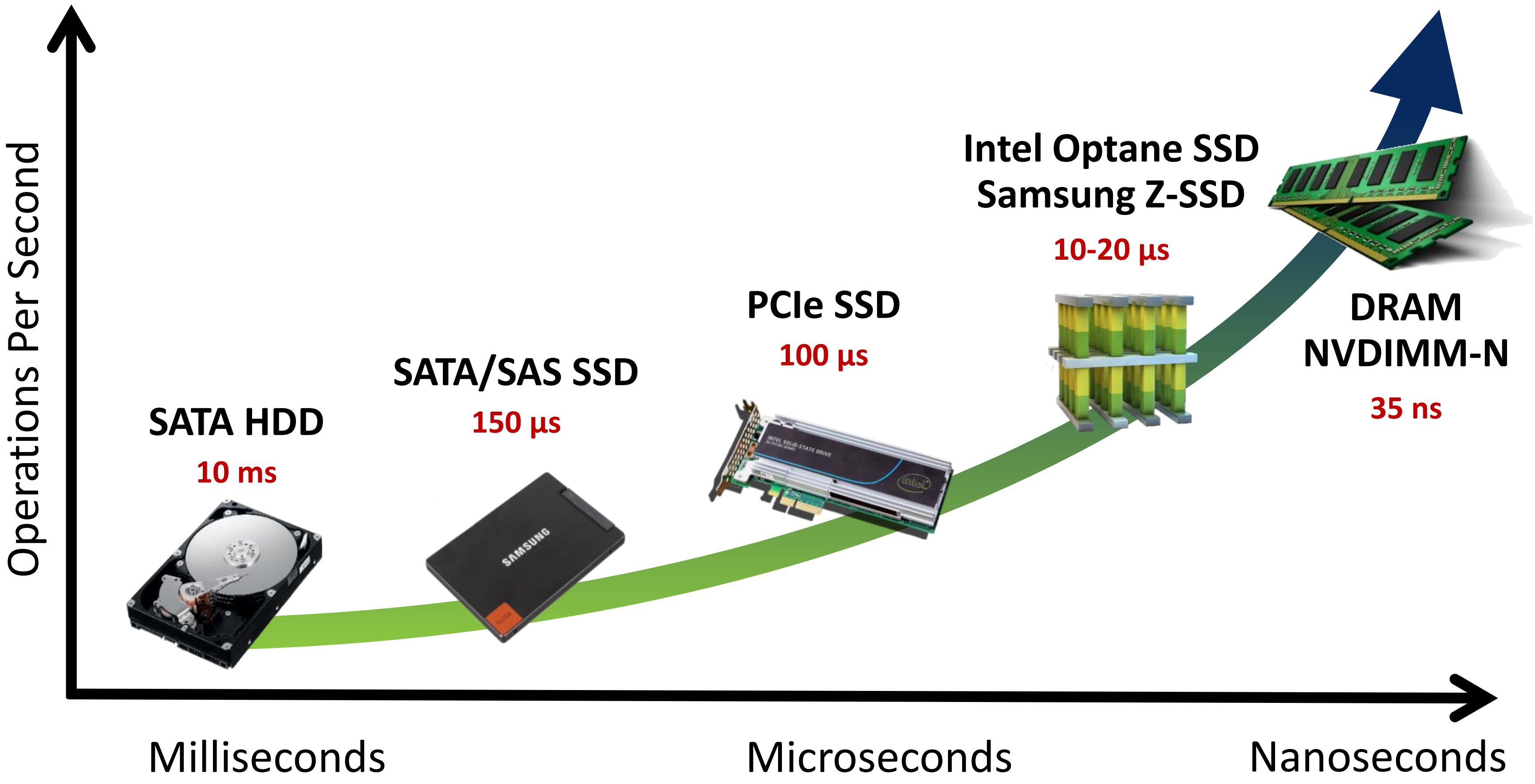
http://www.anandtech.com/show/9451/the-2tb-samsung-850-pro-evo-ssd-review

(Messy) Storage Interfaces



J. Pappas, Annual Update on Interfaces, FMS, 2015.

Moving Closer to the Processor

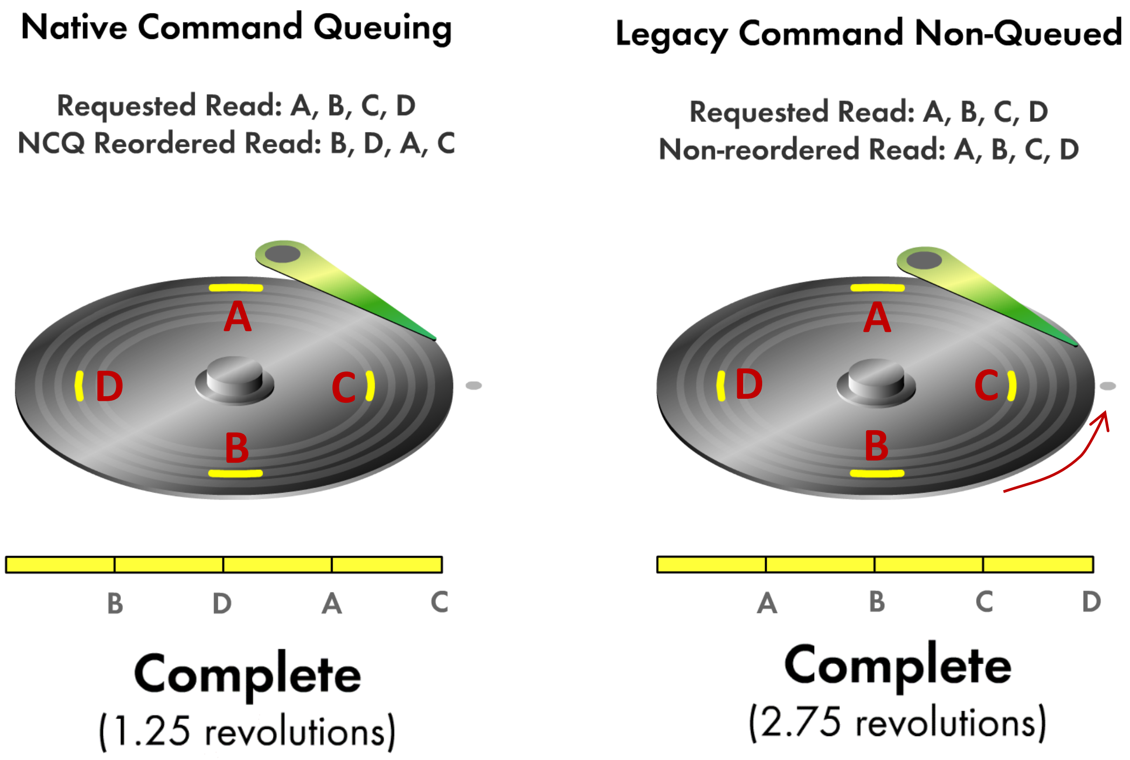


Serial ATA (SATA)

- Primary internal storage interconnect for desktop and mobile PCs
  - Evolved from (Parallel) ATA
  - More than 1.1 billion SATA drives shipped during 2001-2008
  - Market share (as of 2008): Desktop (99%), Mobile PC (97.7%), Enterprise (27.6%)
- Serial, point-to-point, half duplex
- Why SATA?
  - Lower pin count (cost, space), Lower voltage support (5V → 0.7V)
  - Higher performance: SATA 3 – 600MB/s @ 6Gbps
  - Simple drive configuration (no slave)
  - Greater reliability (CRC/packet)
  - Migration to servers (hot plug, NCQ, …)

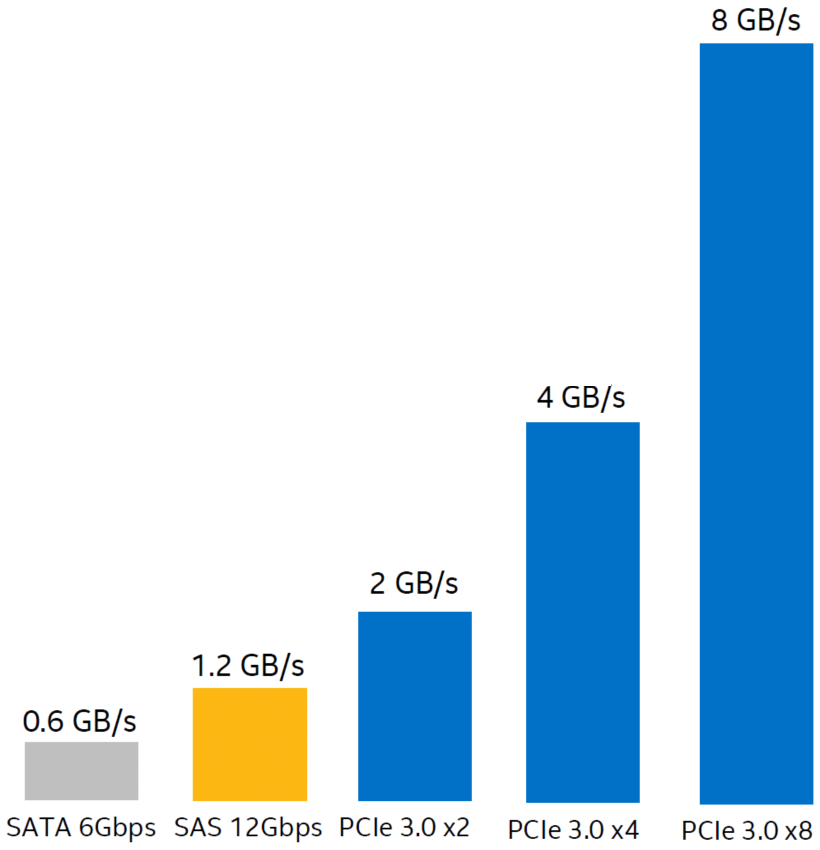
SATA NCQ

- Enqueue up to 32 commands in the drive
- Process them in an out-of-order fashion



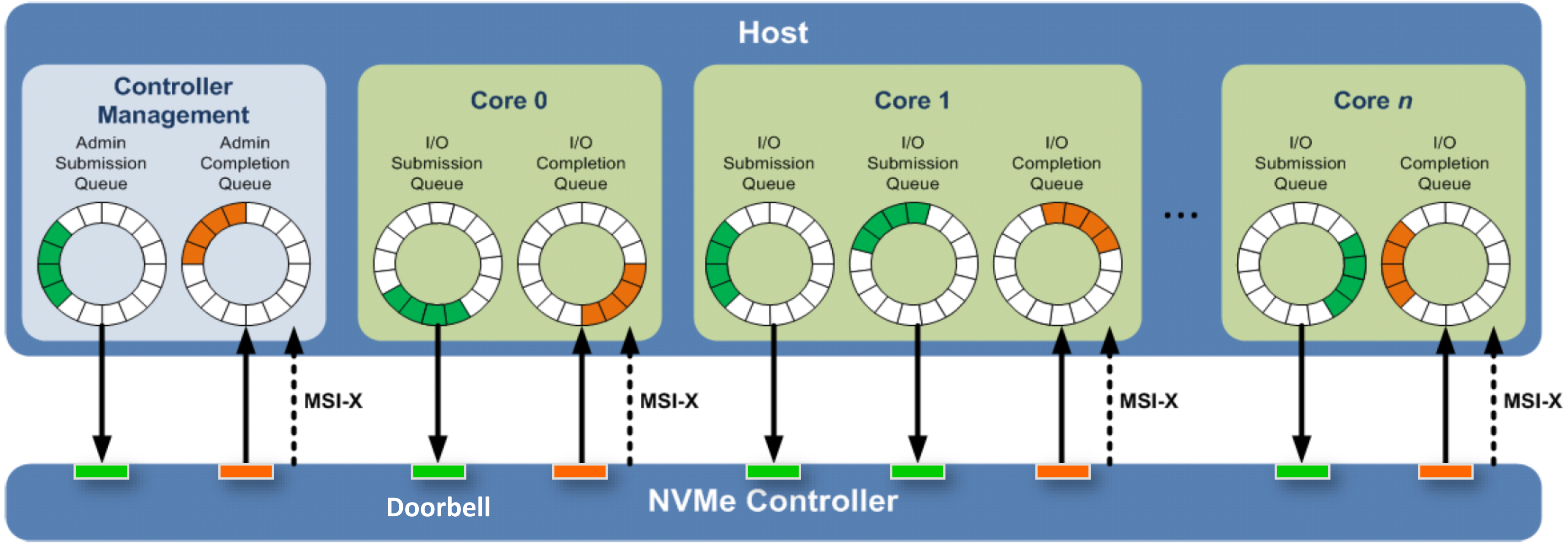
NVMe (NVM Express)

- The industry standard interface for high-performance NVM storage
  - NVMe 1.0 in 2011 by NVM Express Workgroup
  - NVMe 1.2 in 2014
- PCIe-based
- Lower latency
  - Direct connection to CPU
  - No HBA (Host Bus Adapter) required: reduced power and cost
- Scalable bandwidth
  - 1GB/s per lane (PCIe Gen3)
  - Up to 32 lanes



NVMe Overview

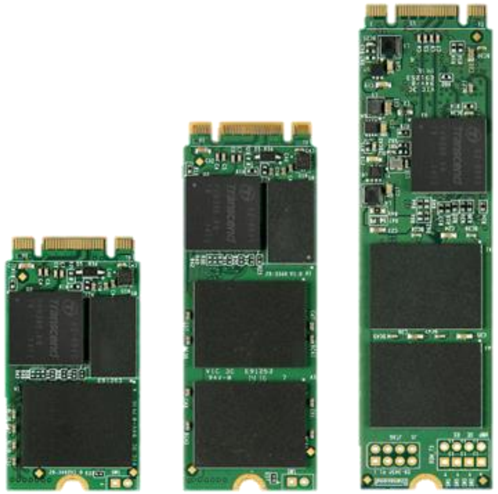
- Deep queue: 64K commands per queue, up to 64K queues
- Streamlined command set: only 13 required commands
- One register write to issue a command (“doorbell”)
- Support for MSI-X and interrupt aggregation



NVMe SSD Form Factors

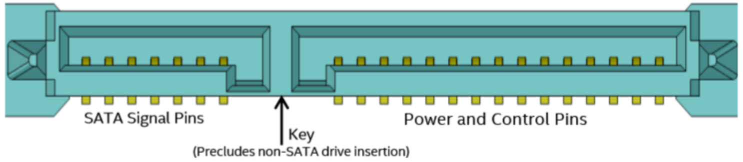
Add-in-card (AIC)

U.2
(SFF-8639:
Up to x4)

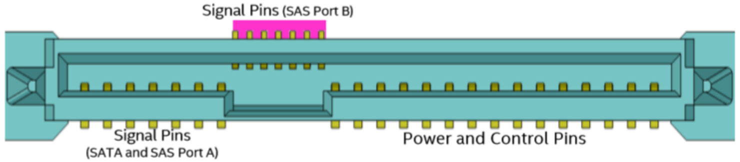


2242 2260 2280
M.2 (PCIe: Up to x4)

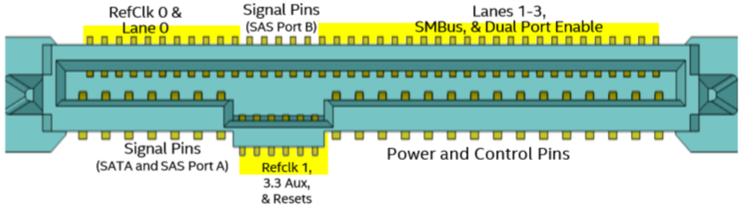
SATA



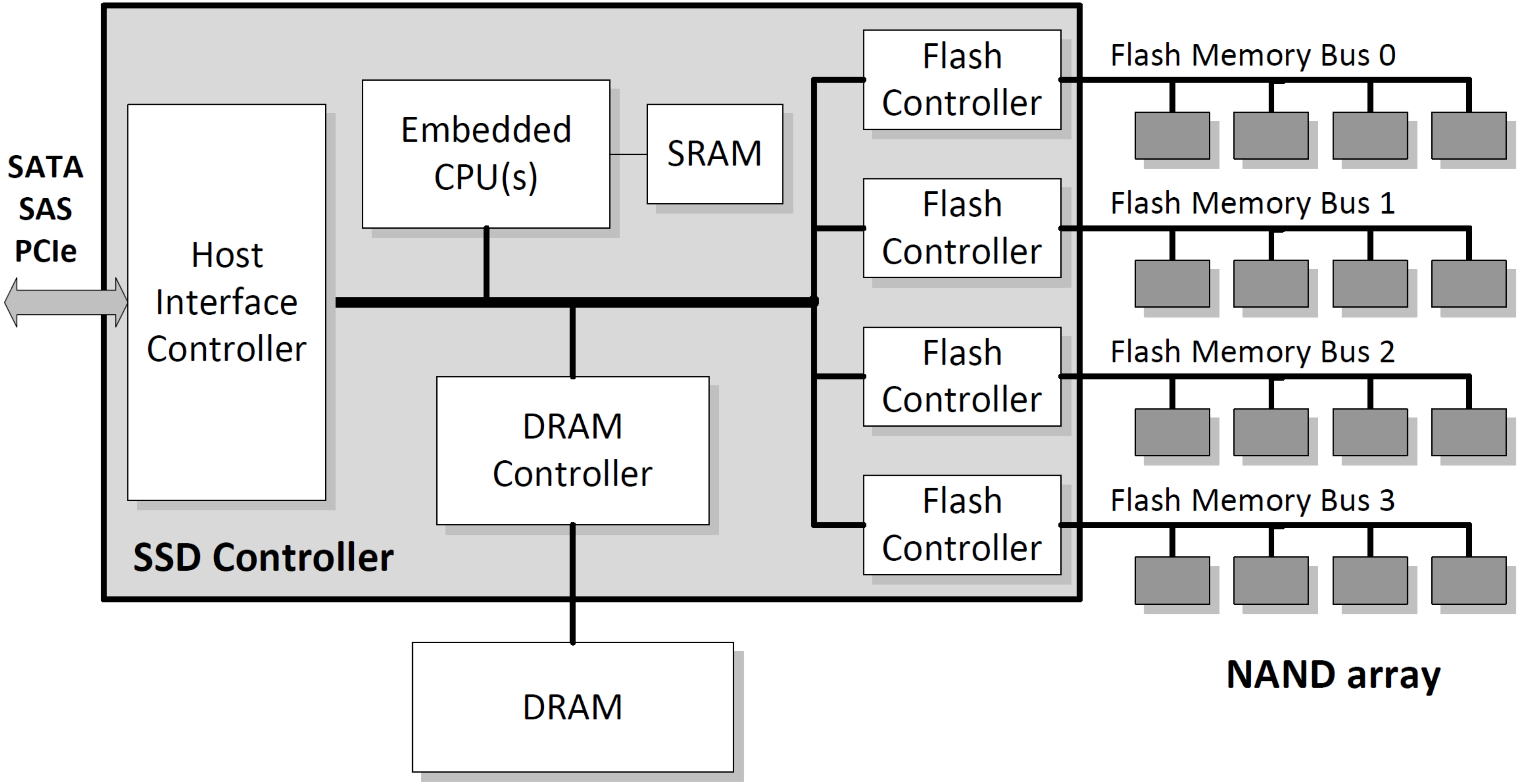
SAS



U.2



SSD Internals



Block Management in Solid-State Devices

(A. Rajimwale et al., USENIX ATC, 2009)

Some of slides are borrowed from the authors' presentation.

SSD – A Different Beast

- SSDs differ from disks
  - No mechanical or moving parts
  - Contain multiple flash elements
  - Different internal architecture
  - Complex internal operations
- SSDs differ among themselves
  - Low, medium, and high end devices
  - Firmware, interconnections, mapping, striping, ganging
- Will the existing file system assumptions hold?

Problem

- Several assumptions are no longer valid

Assumptions	Disks	SSDs
Sequential accesses much faster than random	✔	✘
No write amplification	✔	✘
Little background activity	✔	✘
Media does not wear down	✔	✘
Distant LBNs lead to longer access time	✔	✘

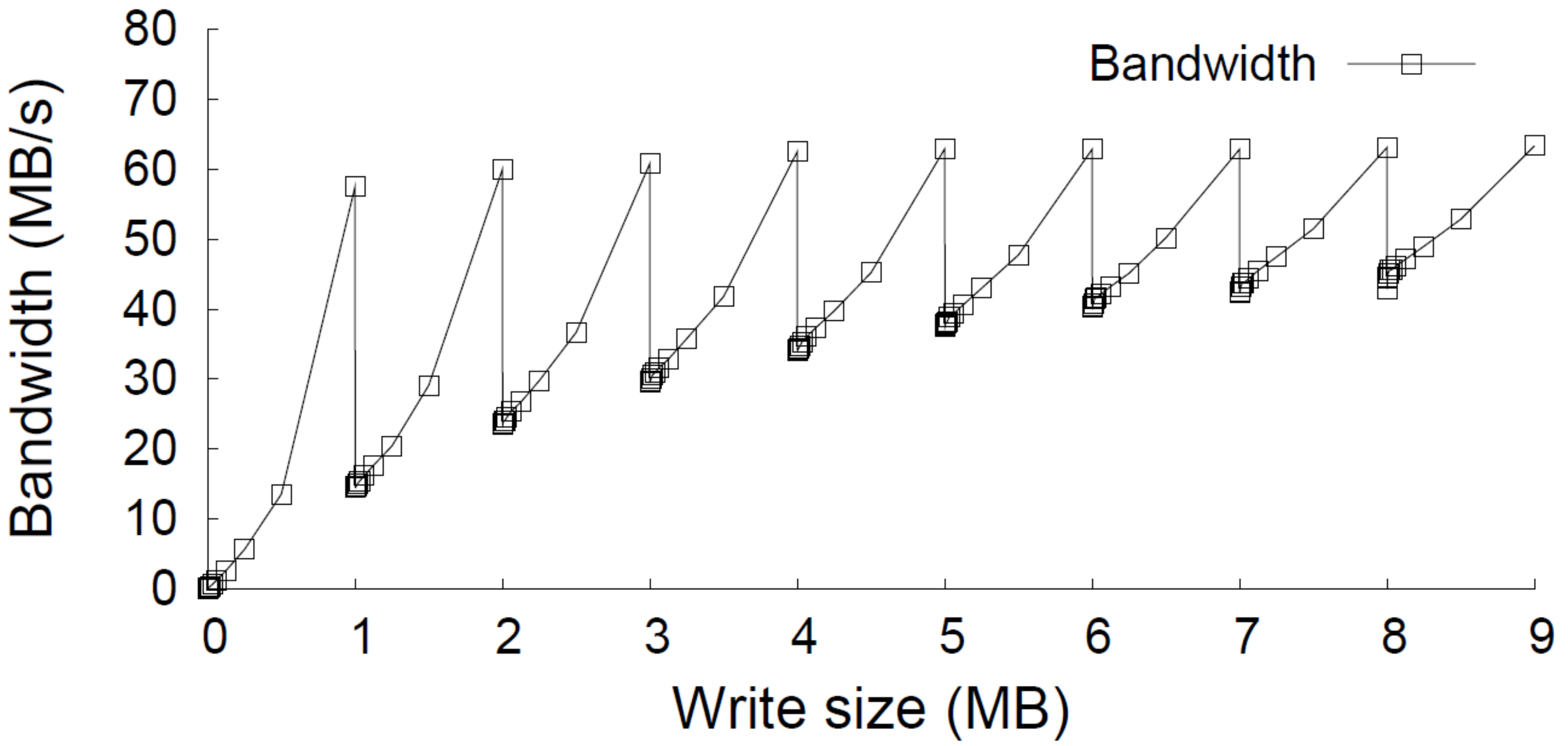
Sequential vs. Random

- Random I/O is only a few times slower than sequential I/O
- File systems must reconsider the policies for block-level sequentiality
- Block management should be moved to SSD

Device	Read			Write		
	Seq	Rand	Ratio	Seq	Rand	Ratio
HDD	86.2	0.6	143.7	86.8	1.3	66.8
$S1_{slc}$	205.6	18.7	11.0	169.4	53.8	3.1
$S2_{slc}$	40.3	4.4	9.2	32.8	0.1	328.0
$S3_{slc}$	72.5	29.9	2.4	75.8	0.5	151.6
$S4_{slc\_sim}$	30.5	29.1	1.1	24.4	18.4	1.3
$S5_{mlc}$	68.3	21.3	3.2	22.5	15.3	1.5

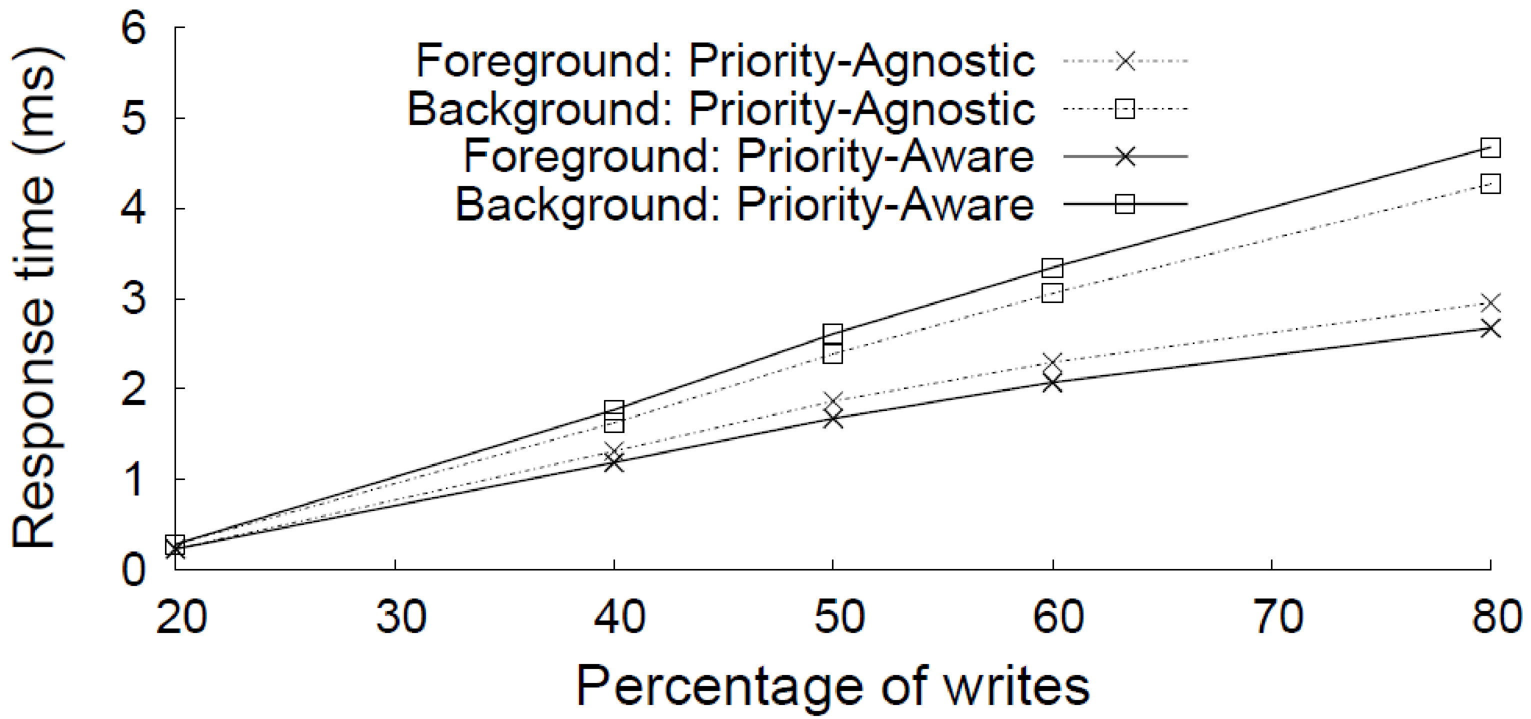
Write Amplification

- When write size is smaller than stripe size (read-modify-write)
- Also due to garbage collection



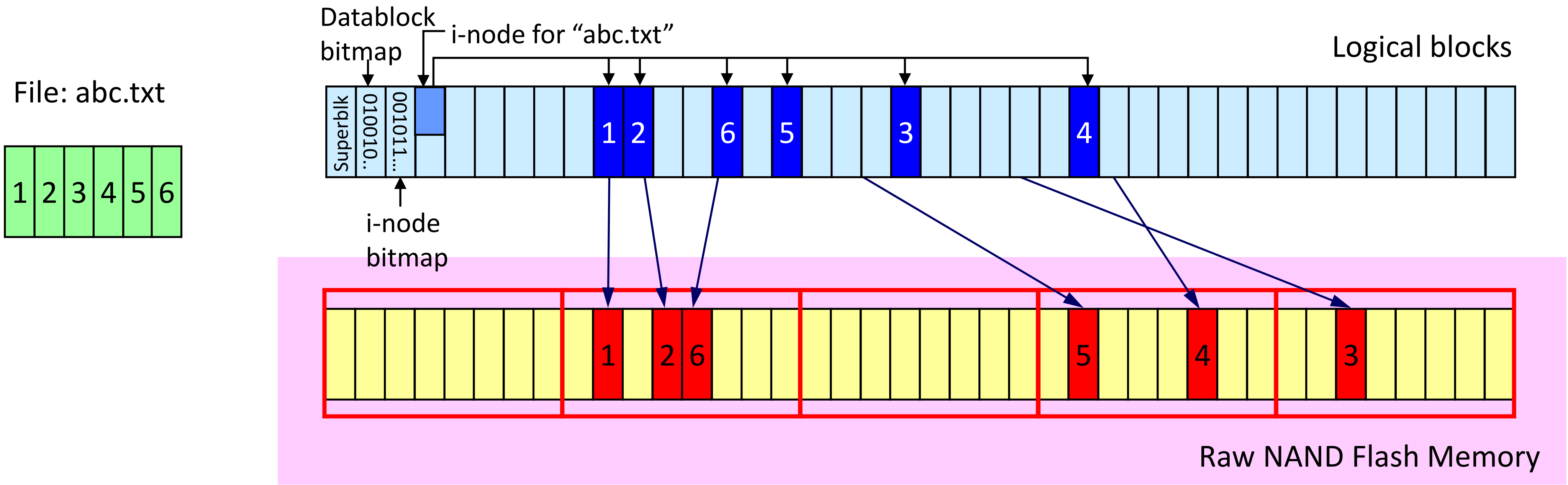
Background Activity

- SSDs perform a considerable amount of background activity (e.g., cleaning, wear-leveling, etc.)
- Improving QoS: Priority-aware cleaning



Block Wear

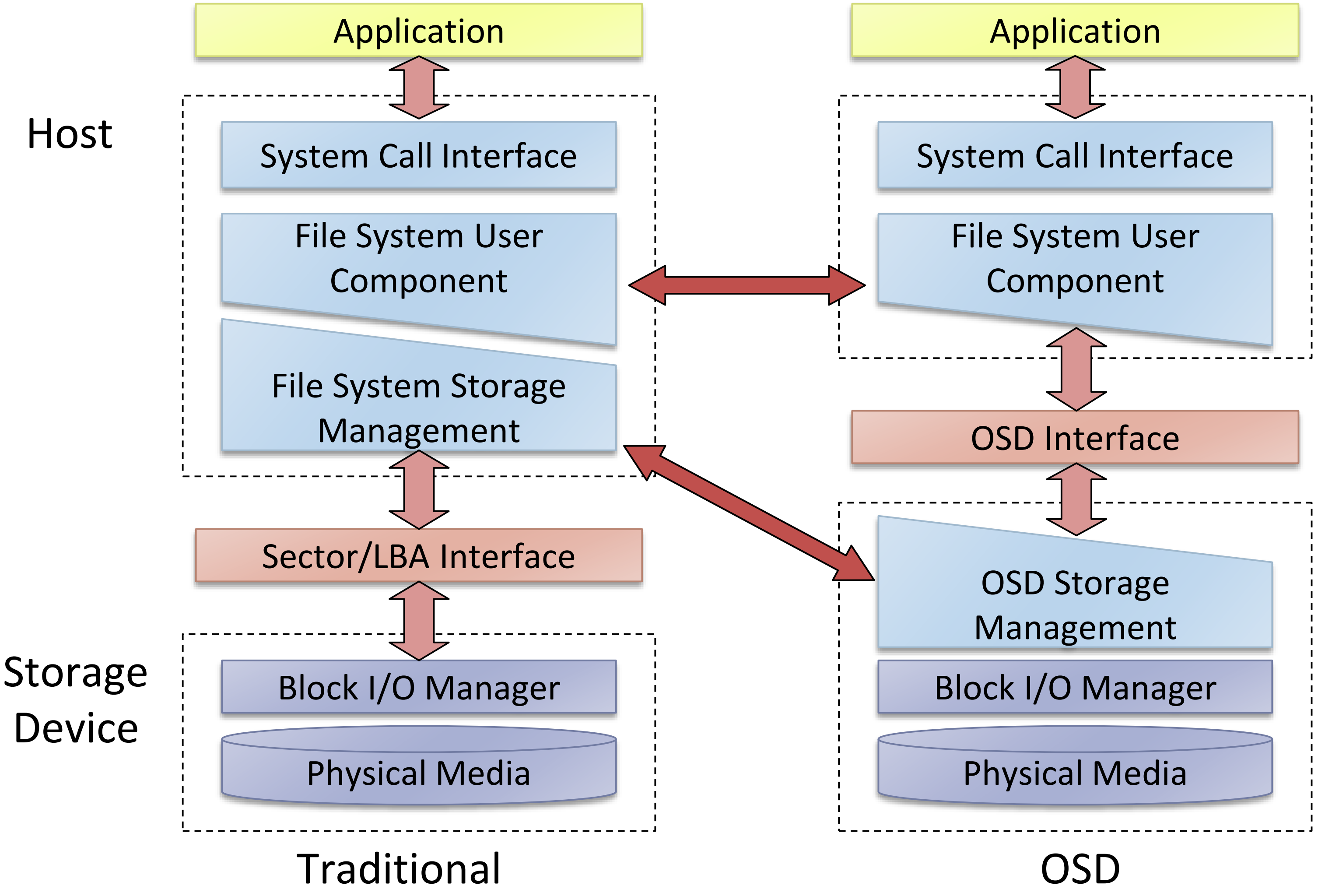
- Flash blocks have a limited P/E (Program/Erase) cycles
- The effectiveness of cleaning and wear-leveling can be improved by using file-system-level semantics
- Informed cleaning: e.g. TRIM or DISCARD command



Logical vs. Physical Distance

- Nearby LBNs do not mean physical proximity due to indirection in SSD (logical-to-physical mapping)
- File system accesses must be in terms of objects
- SSD must handle the low-level sector-specific scheduling

Object-based Storage Device?

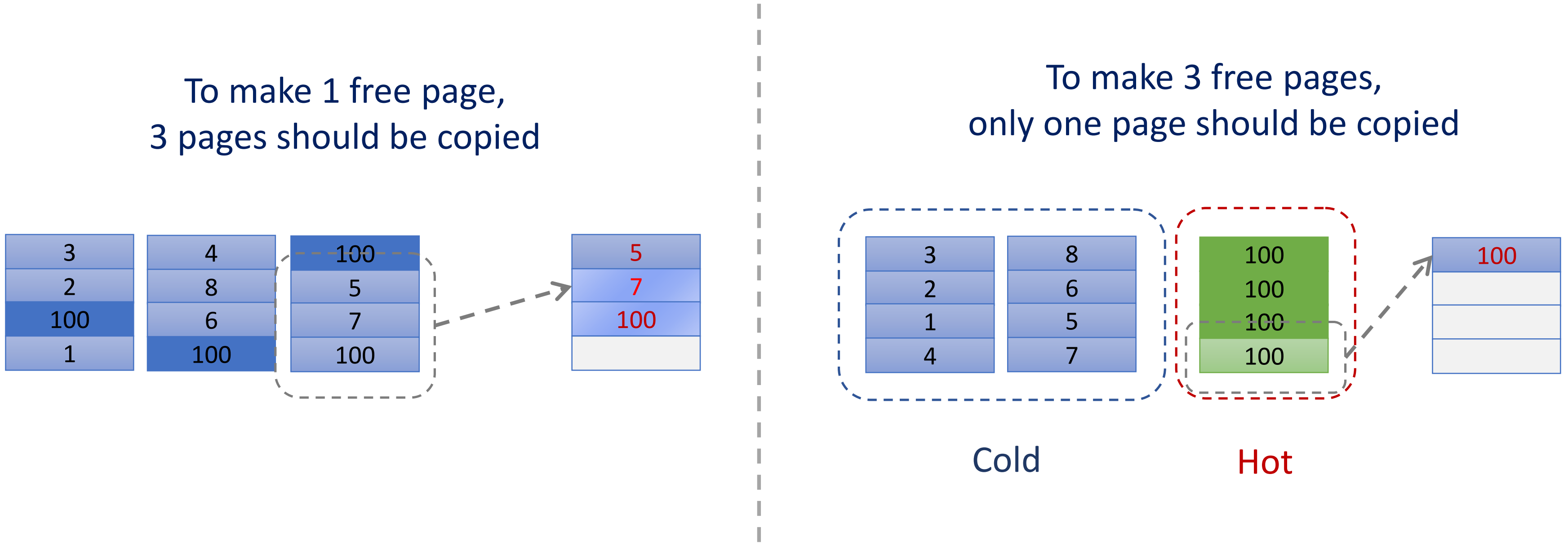


Using Data Clustering to Improve Cleaning Performance for Flash Memory

(M.-L. Chaing et al., SP&E, 1999)

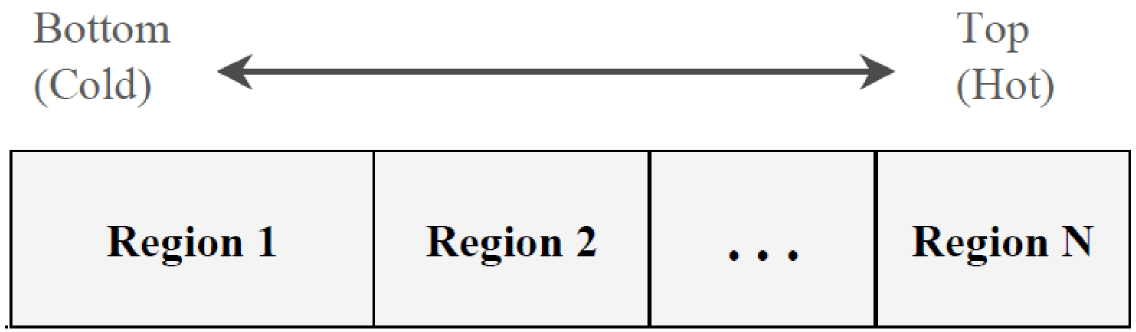
Separating Hot/Cold data

- Separating hot data from cold data can reduce garbage collection overhead



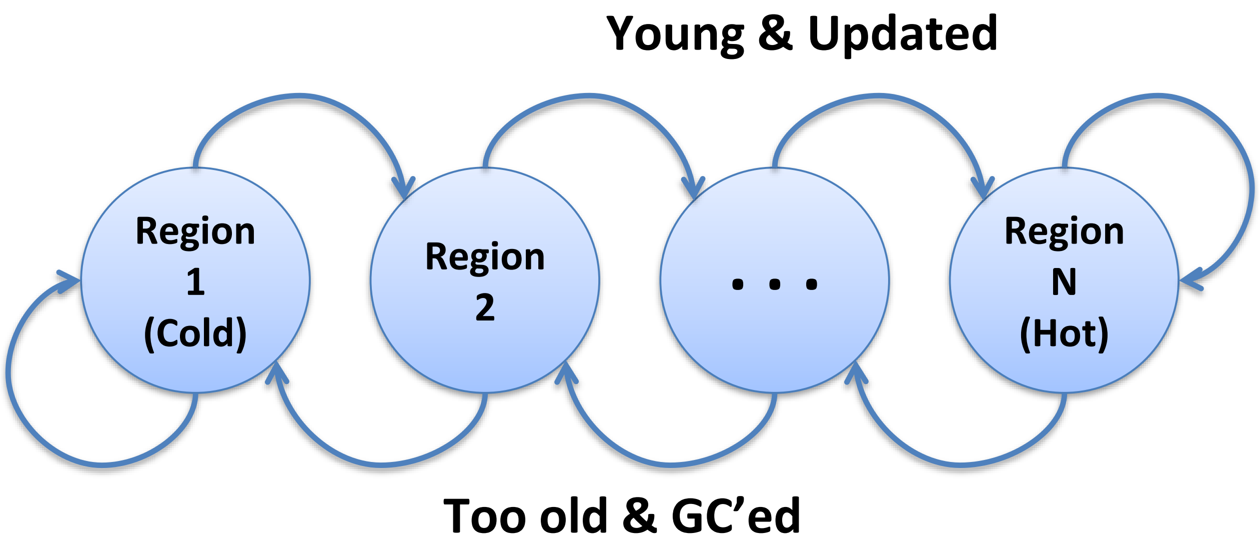
DAC

- Dynamic dAta Clustering
  - Logically partitioning flash memory into regions
  - A separate update block for each region
  - Separate hot pages from cold pages
  - Cluster pages of similar write access frequencies in the same region



Data Clustering

- Key idea
  - Dynamically clusters data not only during garbage collection, but also during data update
  - Similar to generational garbage collection



State Transitions

- Move data toward the Top region
  - Update frequencies increase
  - When the data is updated within a time threshold
- Move data toward the Bottom region
  - Update frequencies decrease
  - When cleaning the data after a time threshold (garbage collection)
- Victim selection in GC
  - Greedy, Cost-benefit, or CAT

Implementation in Page Mapping

- Page-level mapping table
  - Current region number for each entry
  - Timestamp (for time threshold)

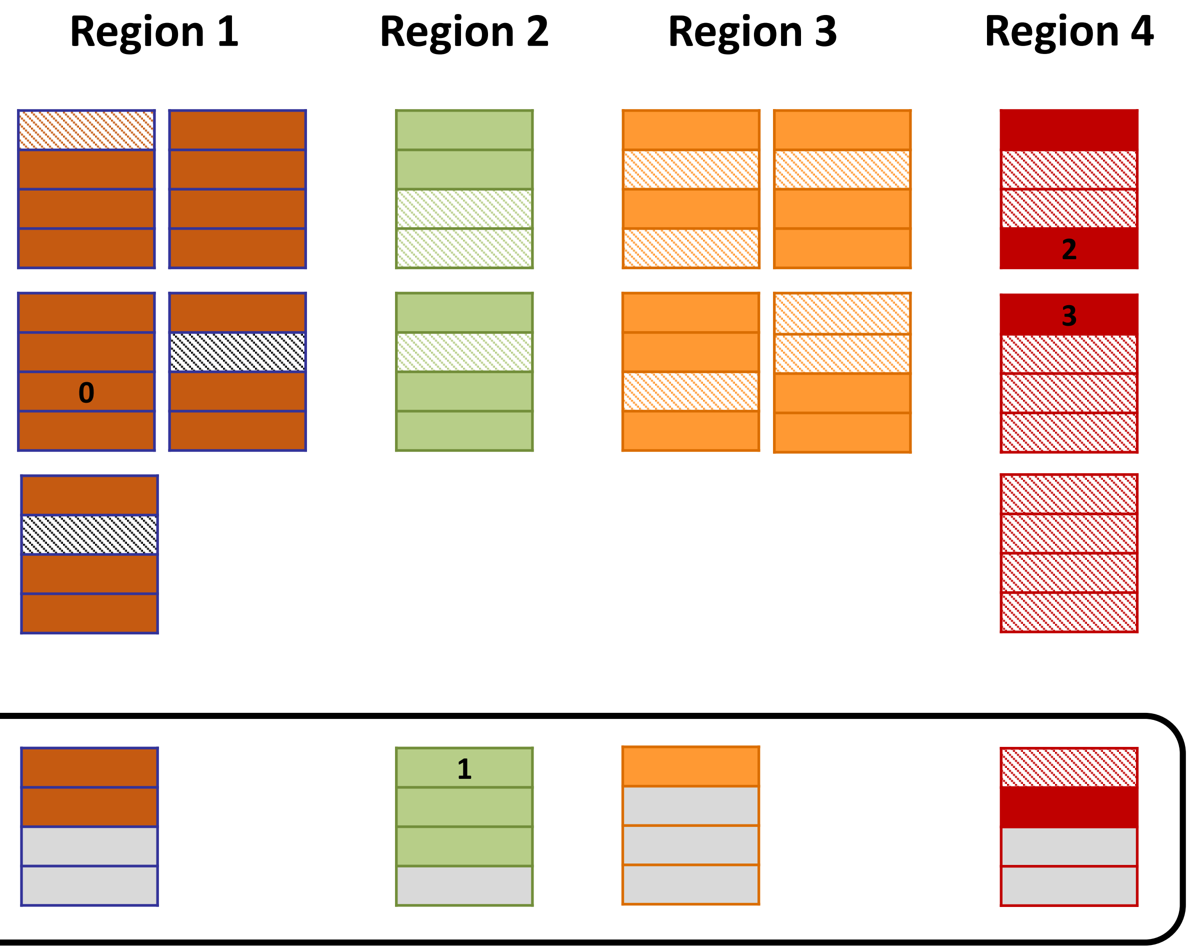
- Update blocks
  - Maintained separately for each region

- Per-block information
  - Region number (optional)
  - Timestamp (for Cost-benefit and CAT)

DAC Architecture

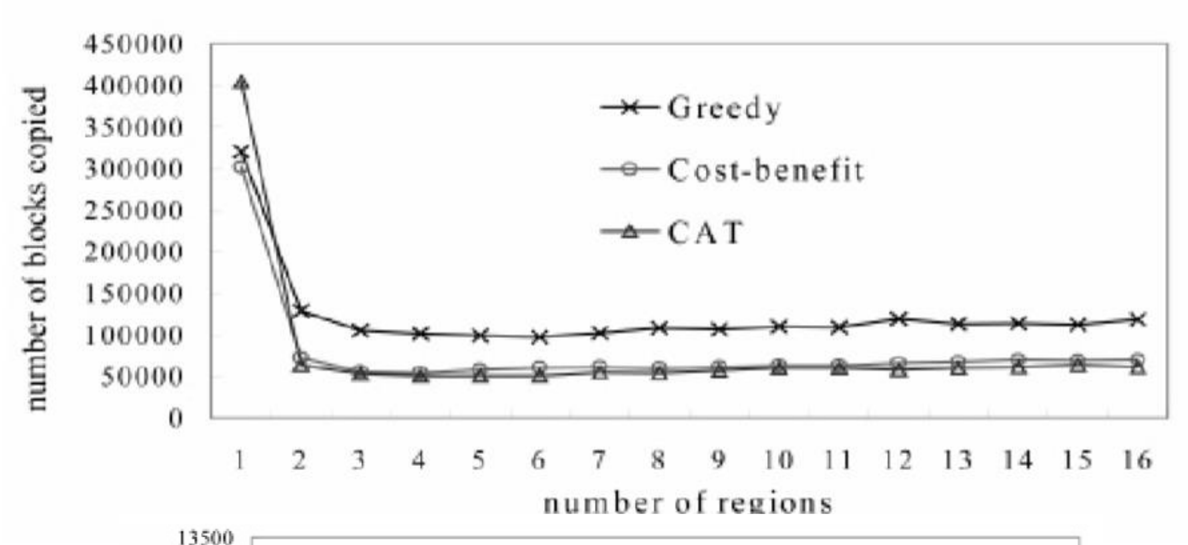
Page Map Table

	PPN	Region	TS
0	12	1	
1	68	2	
2	103	4	
3	104	4	

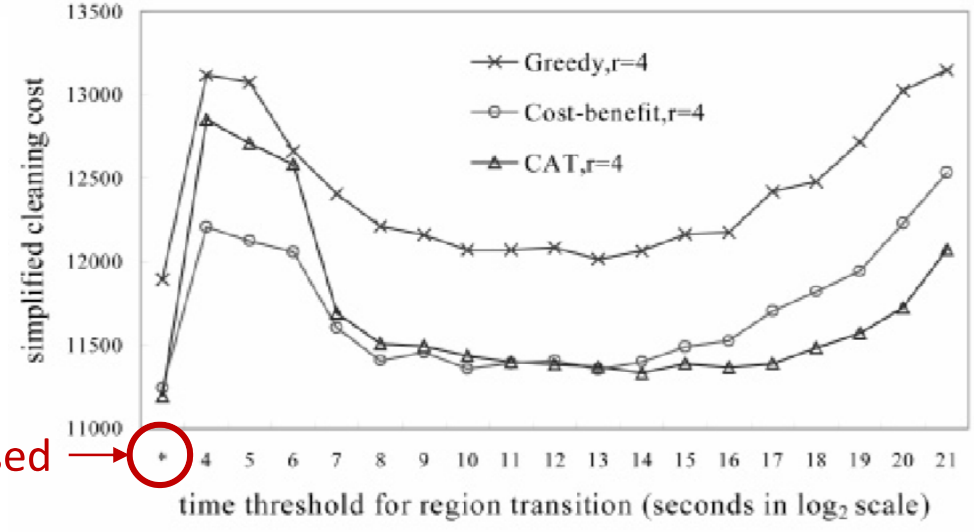


DAC Performance

- hplajw trace
  - without time threshold



- The effects of time threshold in 4 regions



No time threshold used

DAC

- Pros
  - Data are clustered with a low overhead during data update and GC
  - Data classification is more fine-grained than the traditional hot vs. cold classification
  - Easy to integrate in the page mapping

- Cons
  - The optimal number of regions depends on the workload
  - The effectiveness of the time threshold depends on the workloads

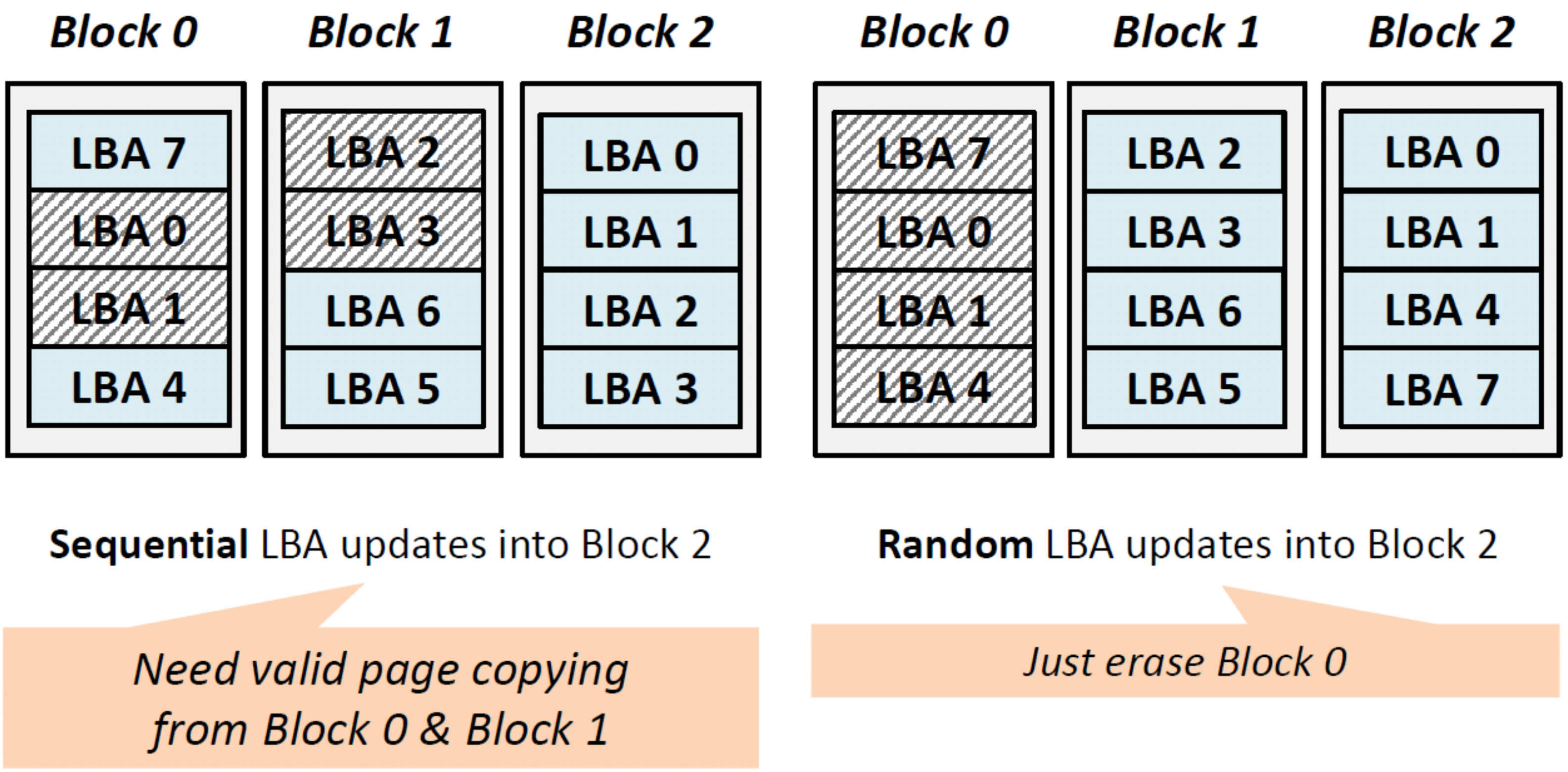
The Multi-streamed Solid-State Drive

(J.-U. Kang et al., HotStorage, 2014)

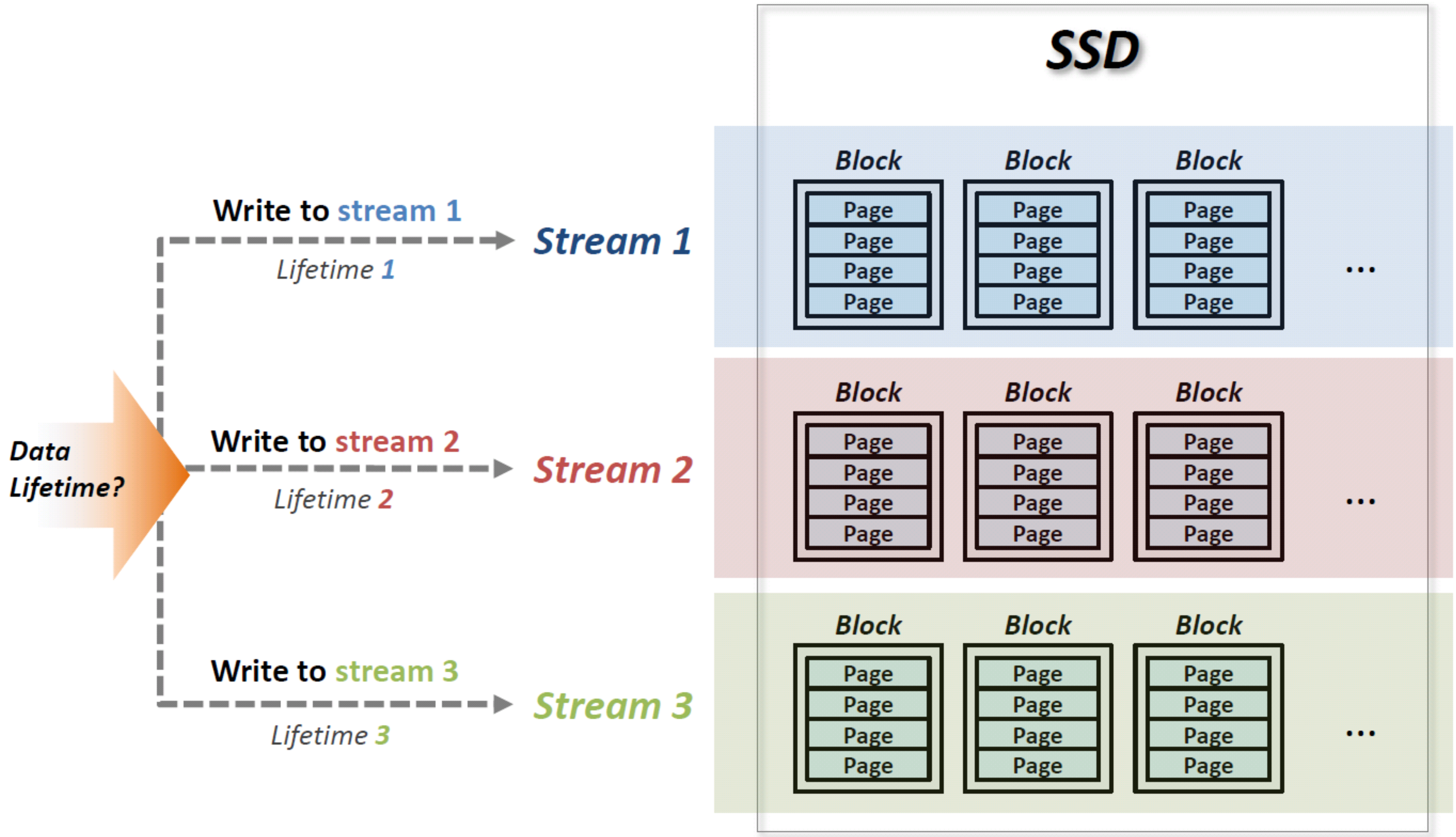
Some of slides are borrowed from the authors' presentation.

Effects of Write Patterns

- Previous write patterns (= current state) matter



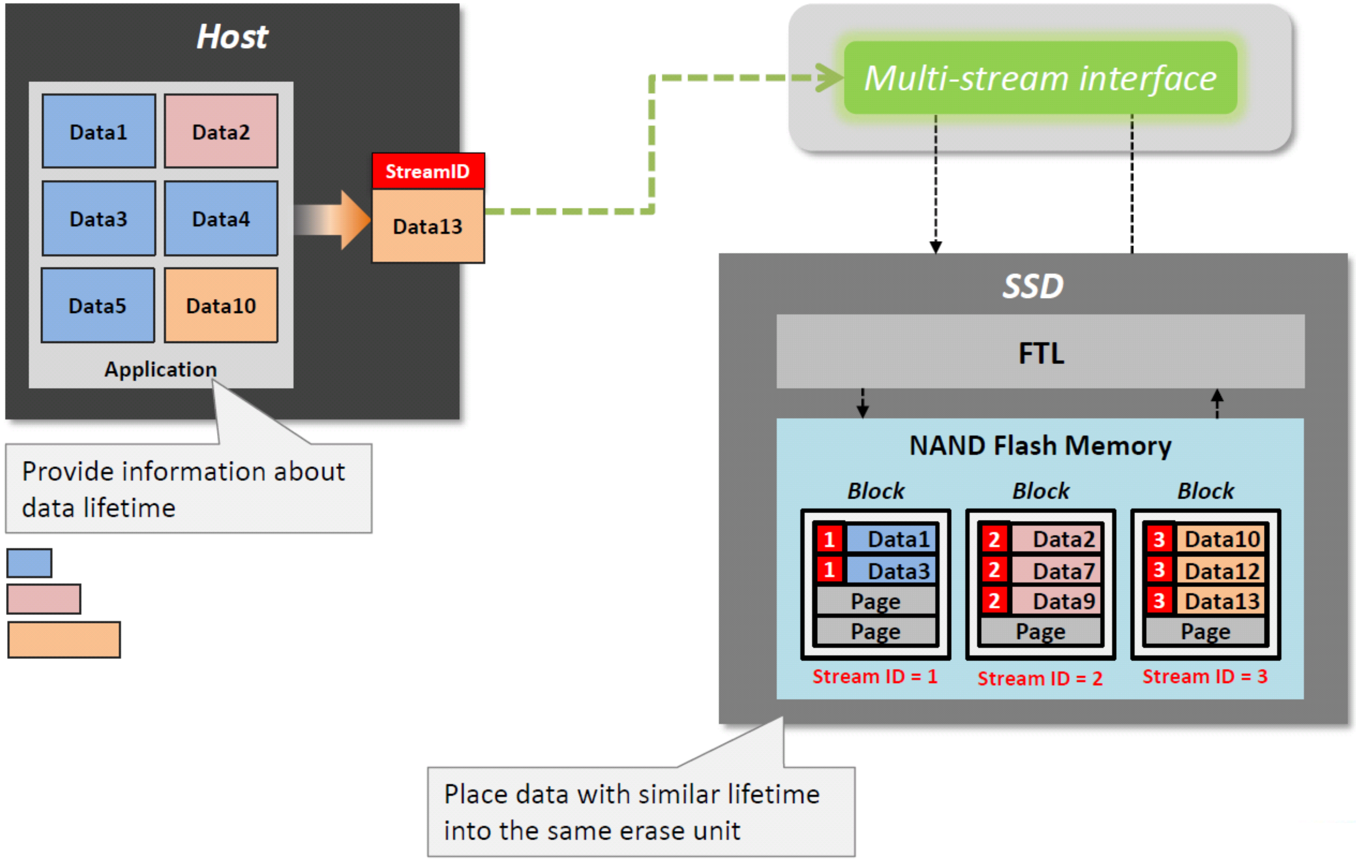
Stream



SSD
Data
Lifetime?
Write to stream 1
Lifetime 1
Stream 1
Write to stream 2
Lifetime 2
Stream 2
Write to stream 3
Lifetime 3
Stream 3
Block
Page
Page
Page
Page
...

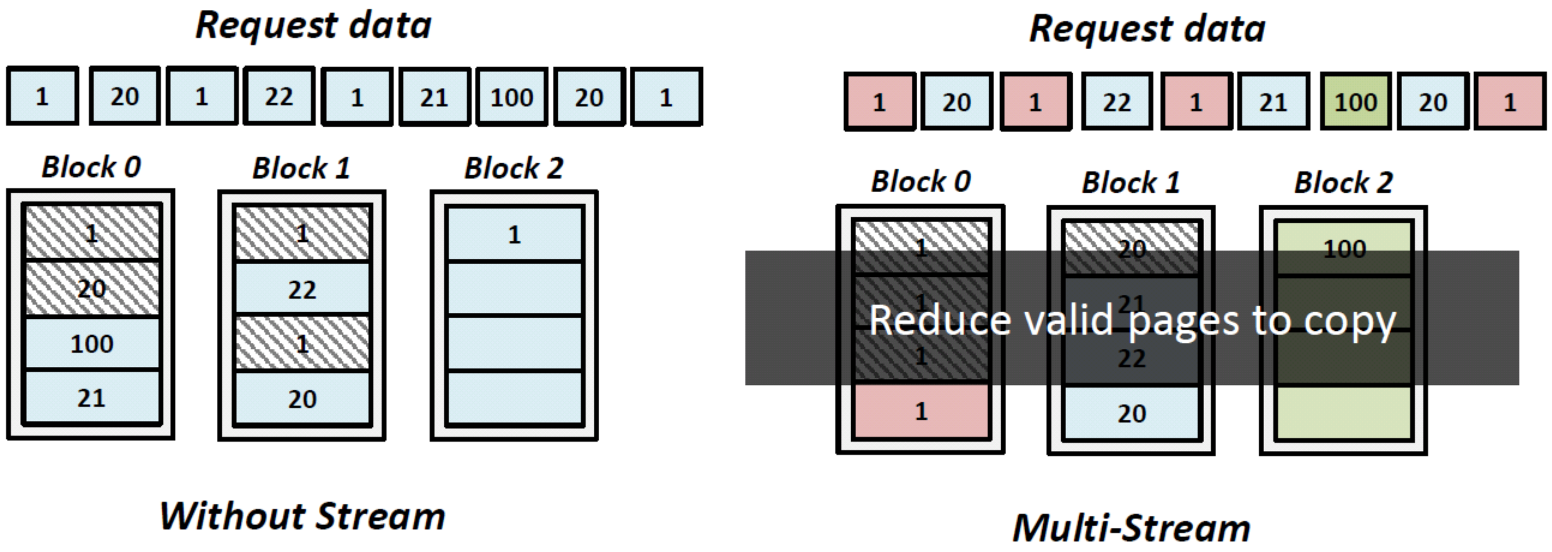
The Multi-streamed SSD

- Mapping data with different lifetime to different streams



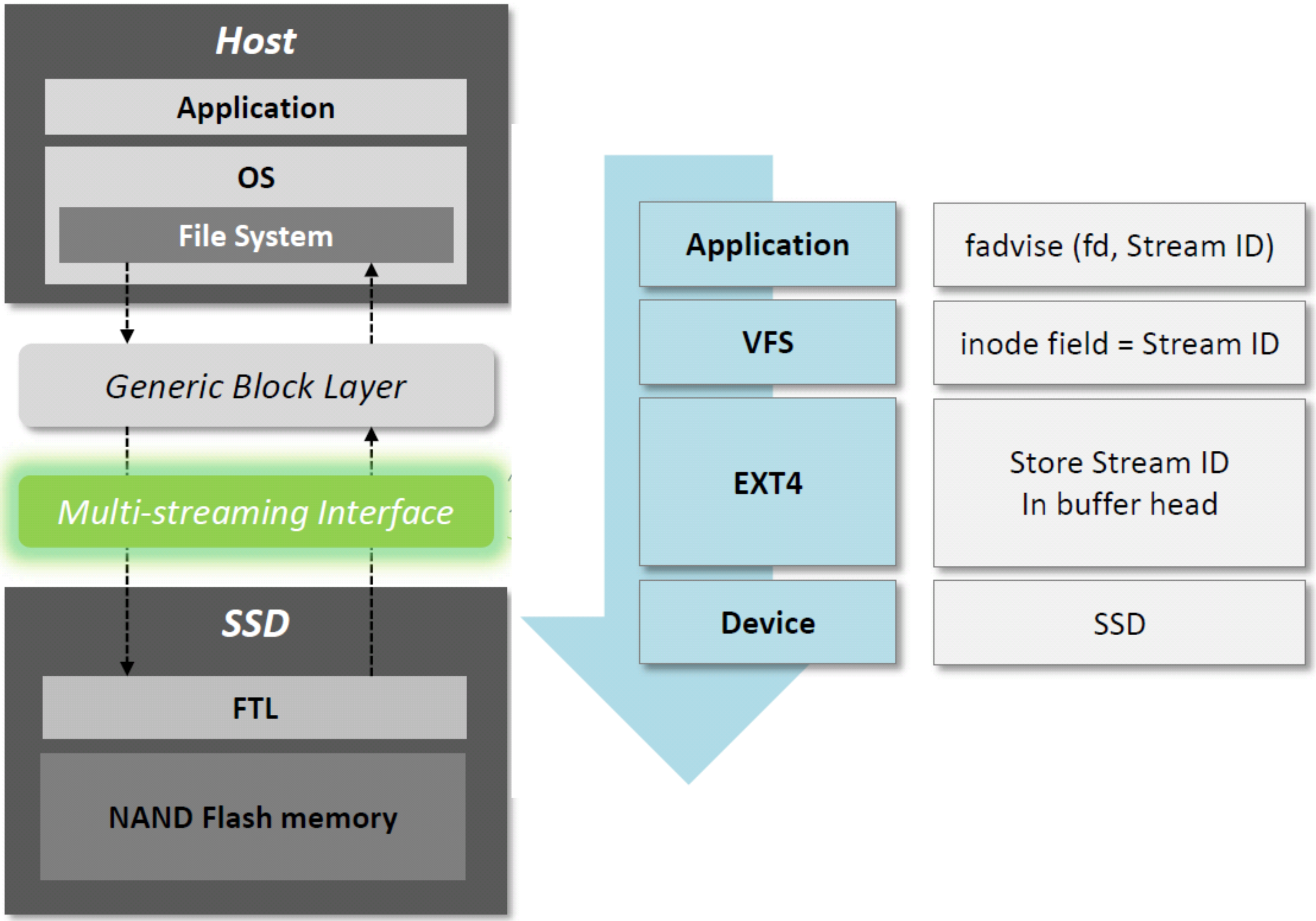
Working Example

- High GC efficiency → Performance improvement

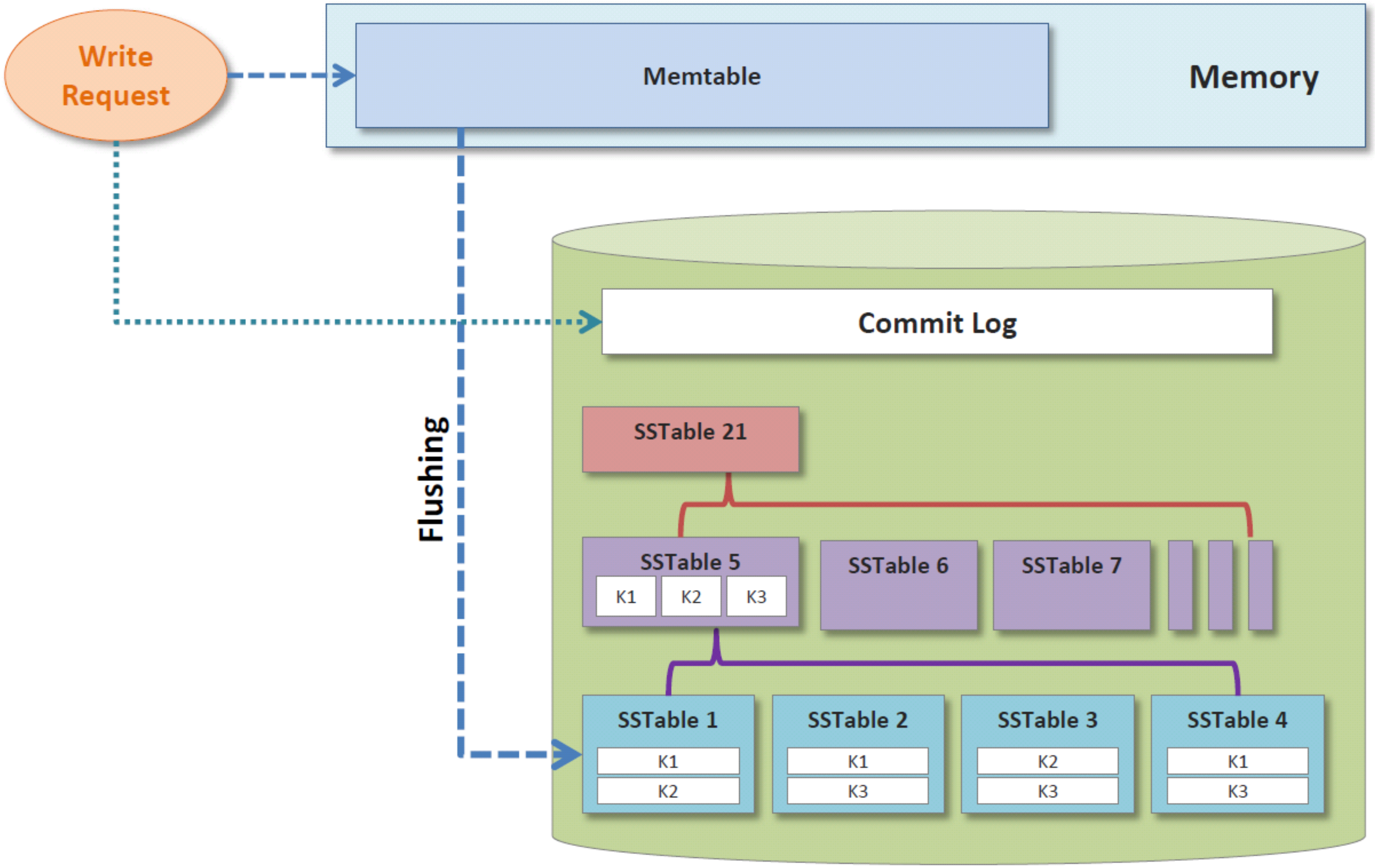


For effective multi-streaming,
proper mapping of data to streams is essential!

Architecture

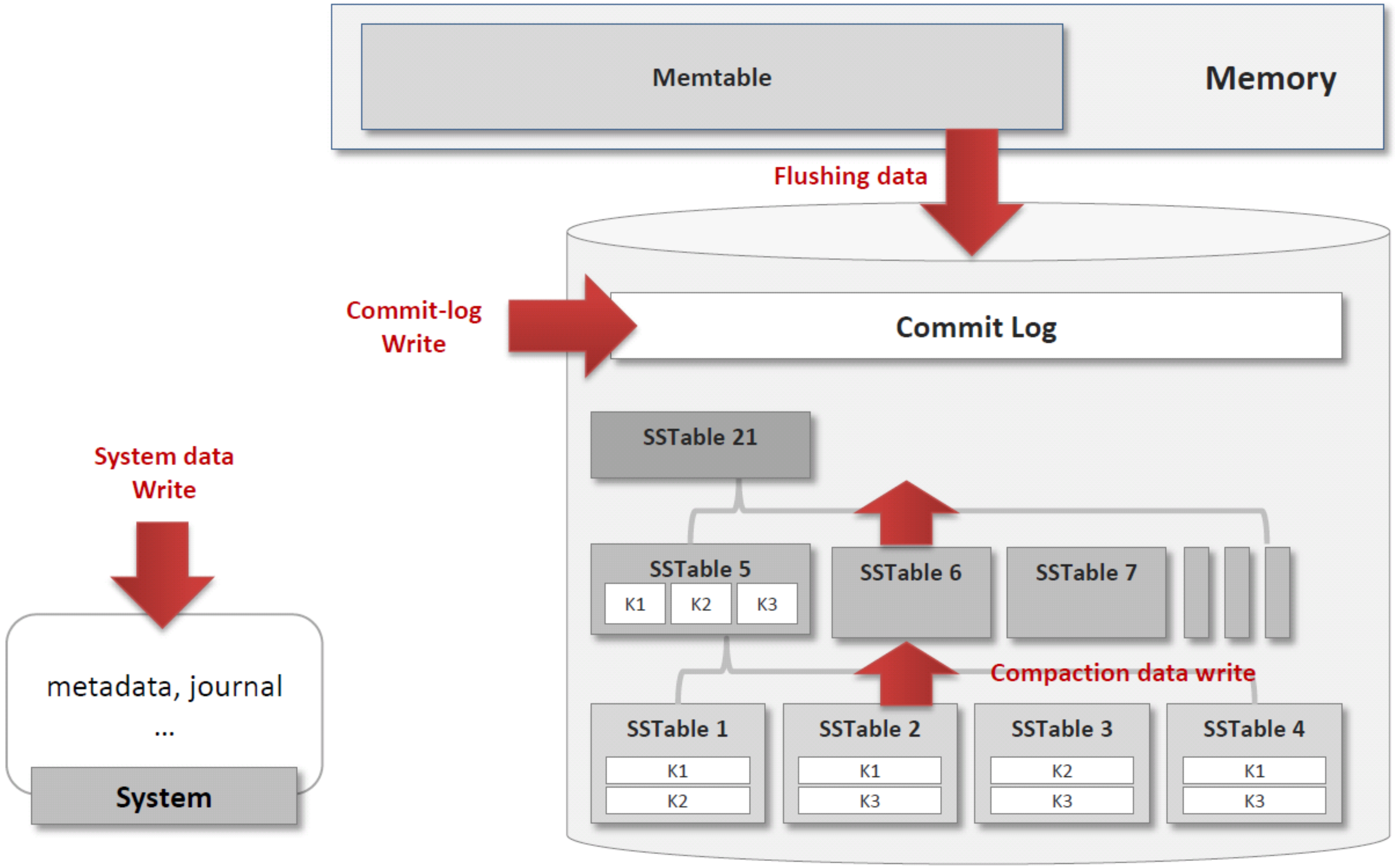


Case Study: Cassandra



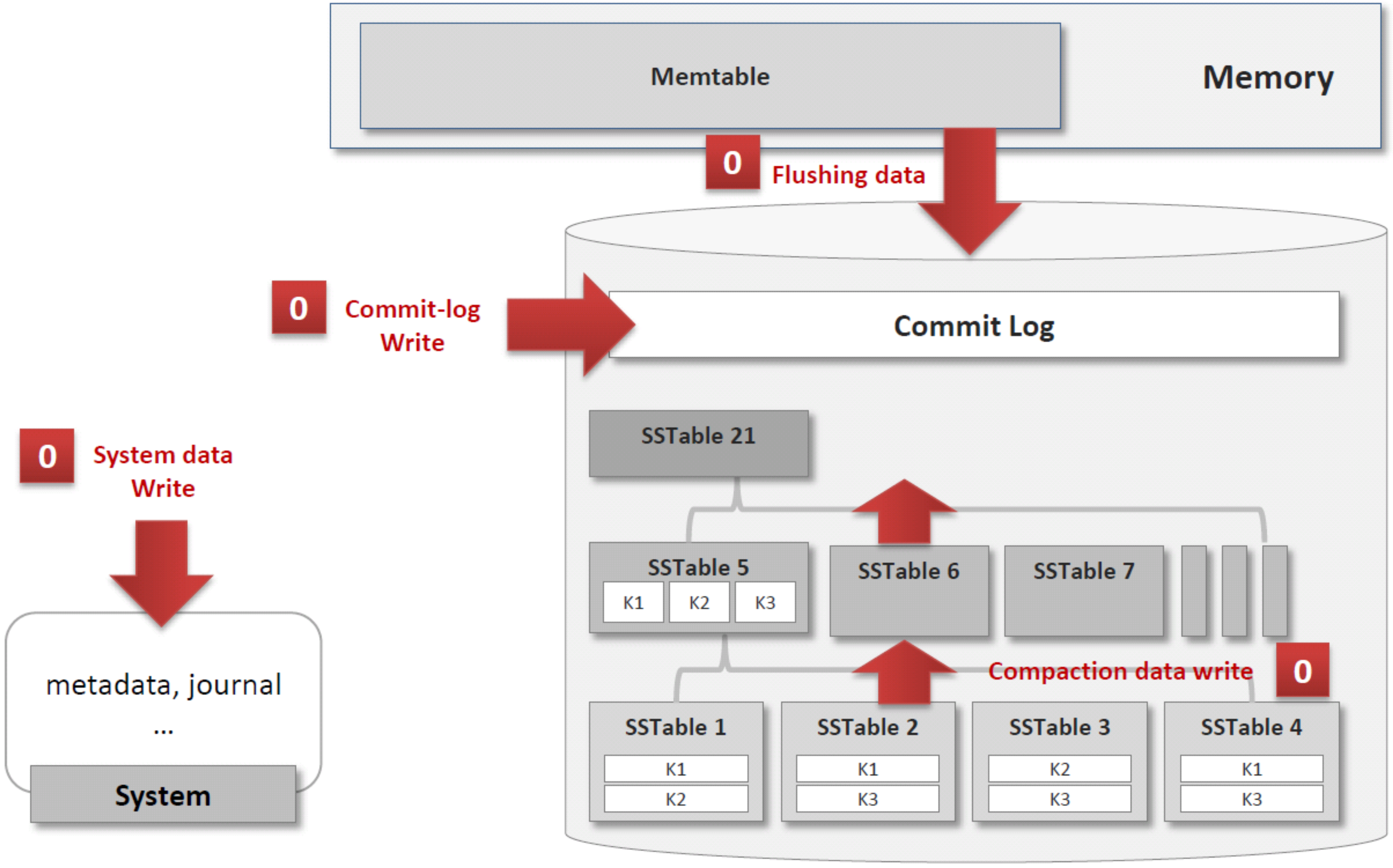
Cassandra's Write Patterns

- Write operations when Cassandra runs



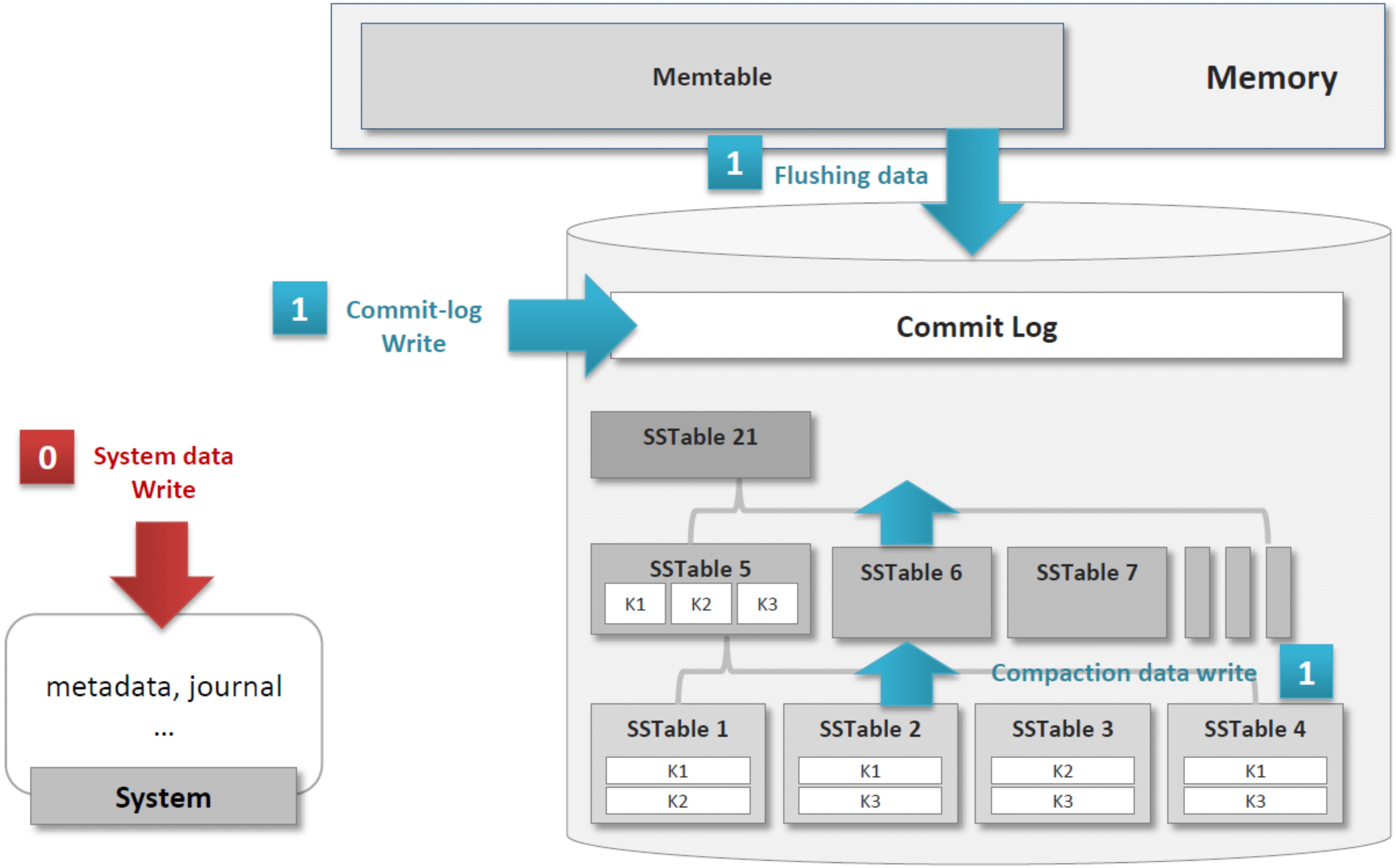
Mapping #1: Conventional

- Just one stream ID (= conventional SSD)



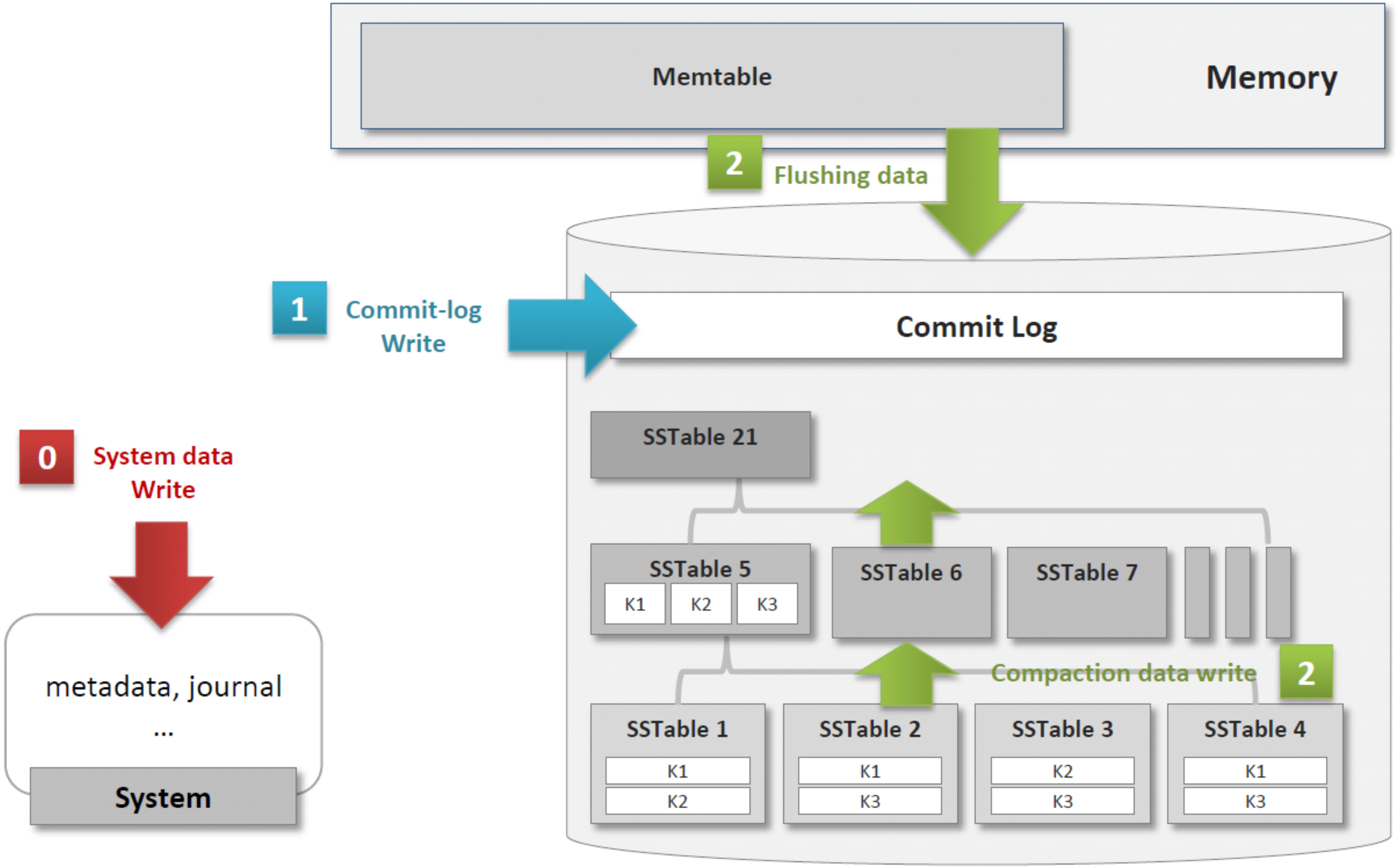
Mapping #2: Multi-App

- Separate application writes (ID 1) from system traffic (ID 0)



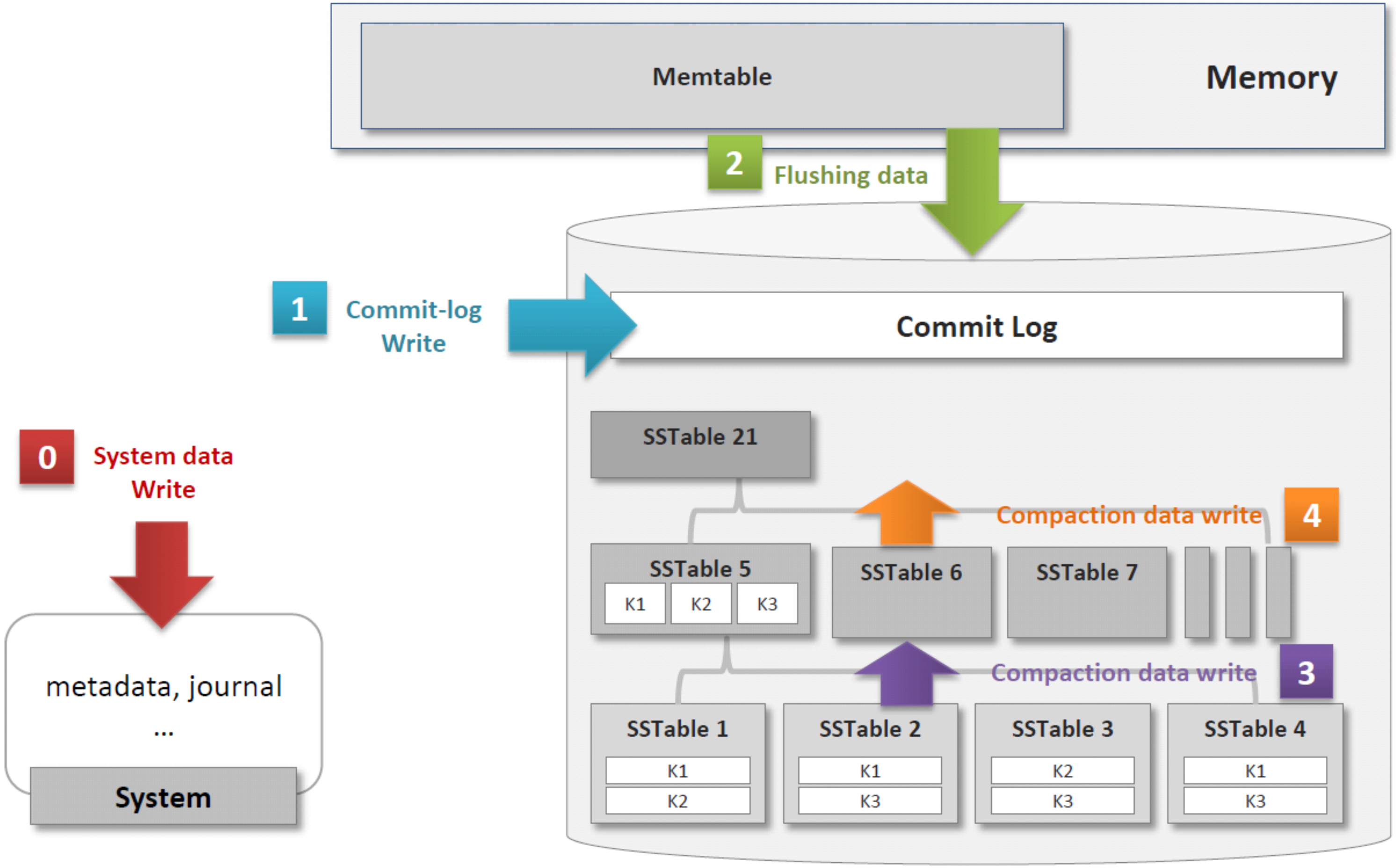
Mapping #3: Multi-Log

- Use three streams; further separate Commit Log



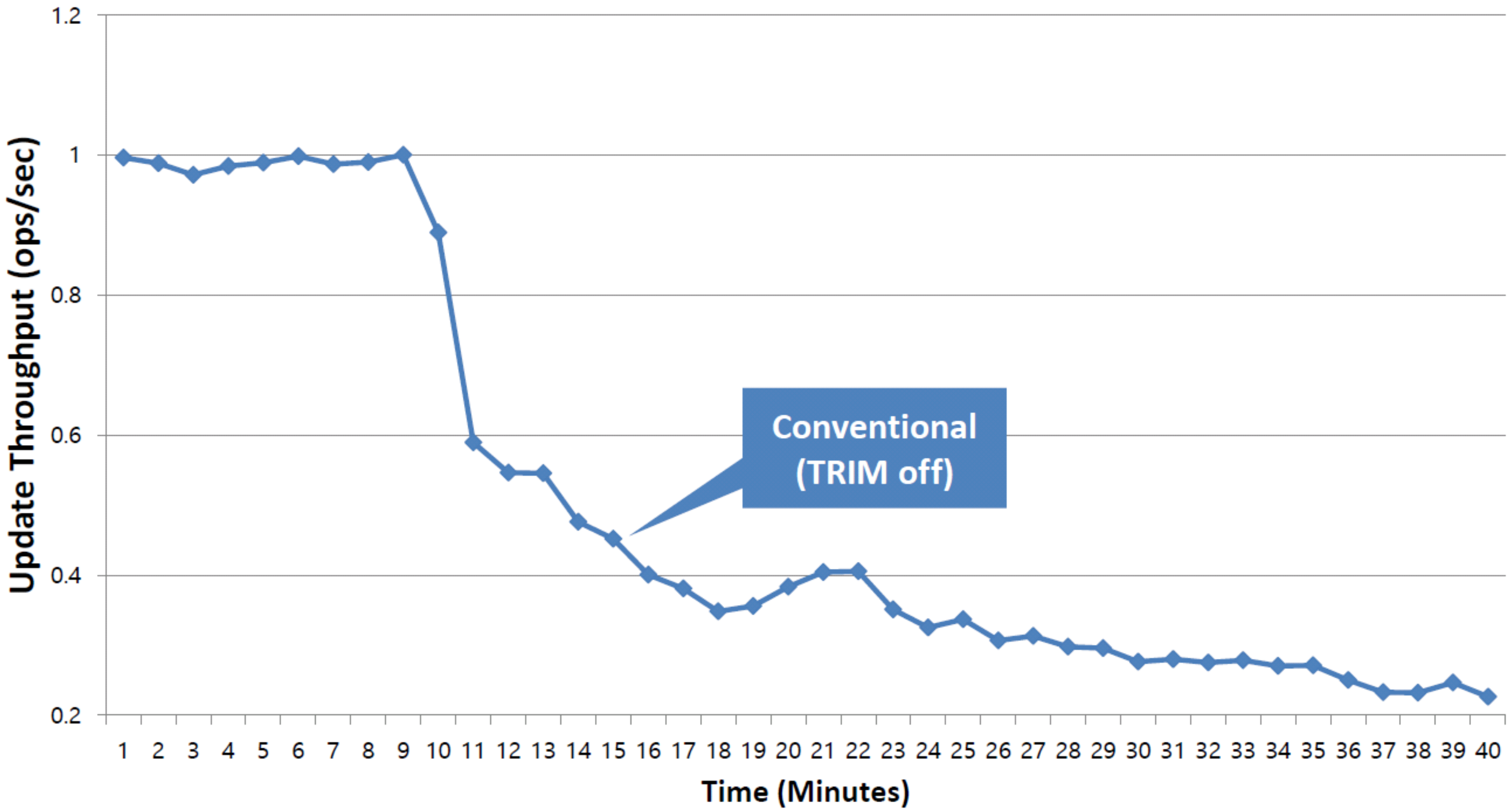
Mapping #4: Multi-Data

- Give distinct streams to different tiers of SSTables



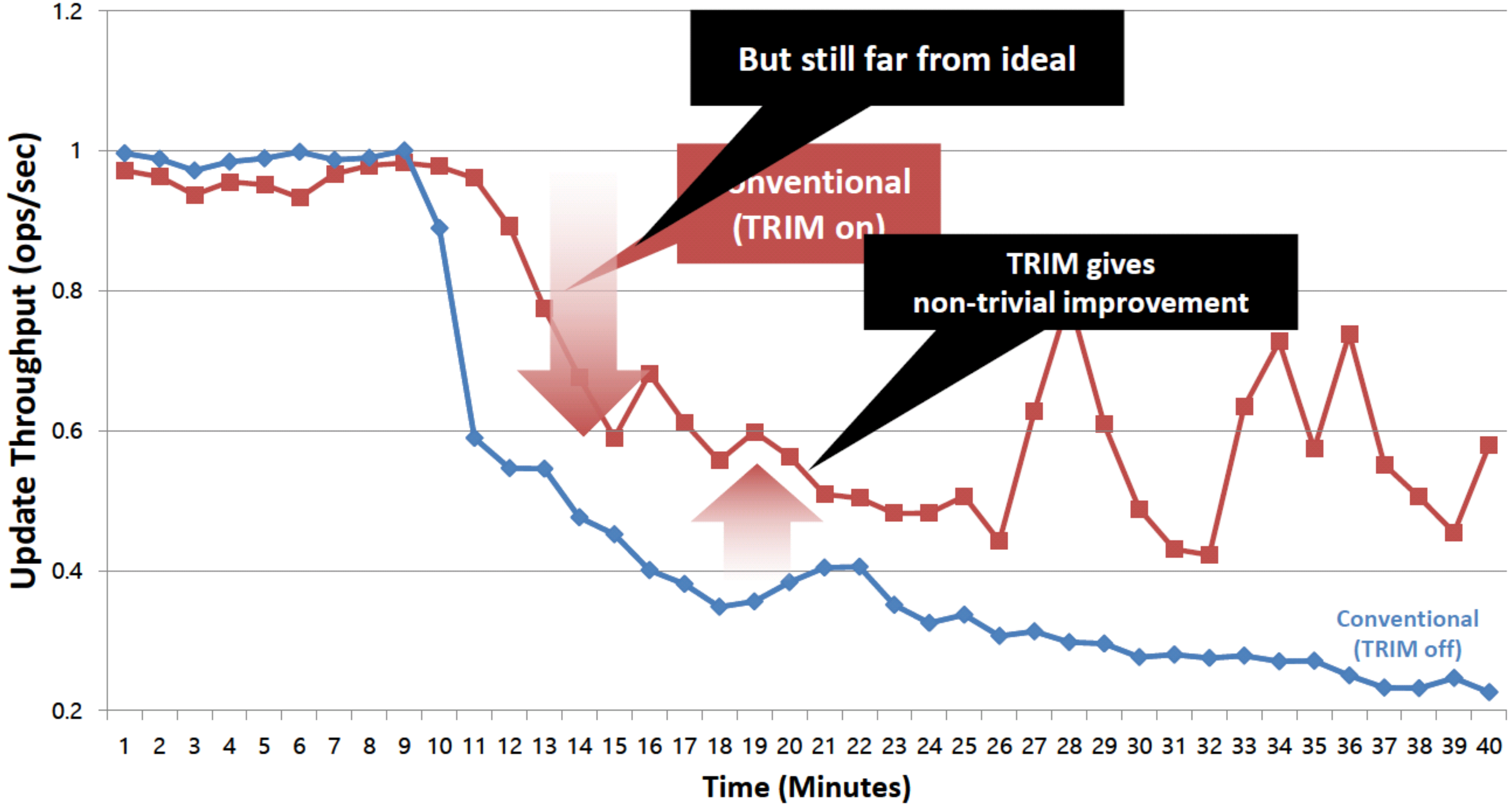
Results: Conventional

- Cassandra's normalized update throughput
  - Conventional "TRIM off"



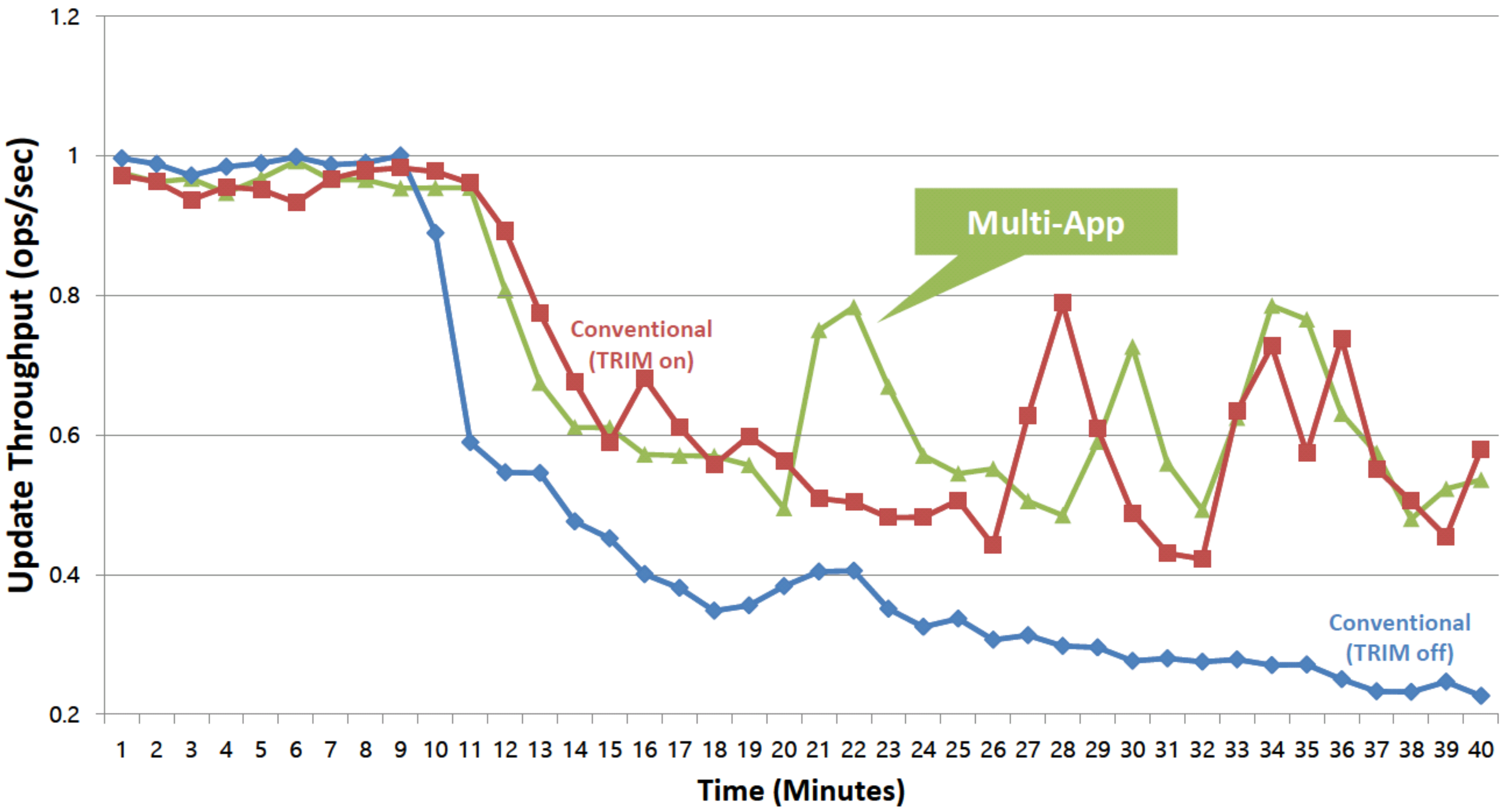
Results: Conventional with TRIM

- Cassandra's normalized update throughput
  - Conventional "TRIM on"



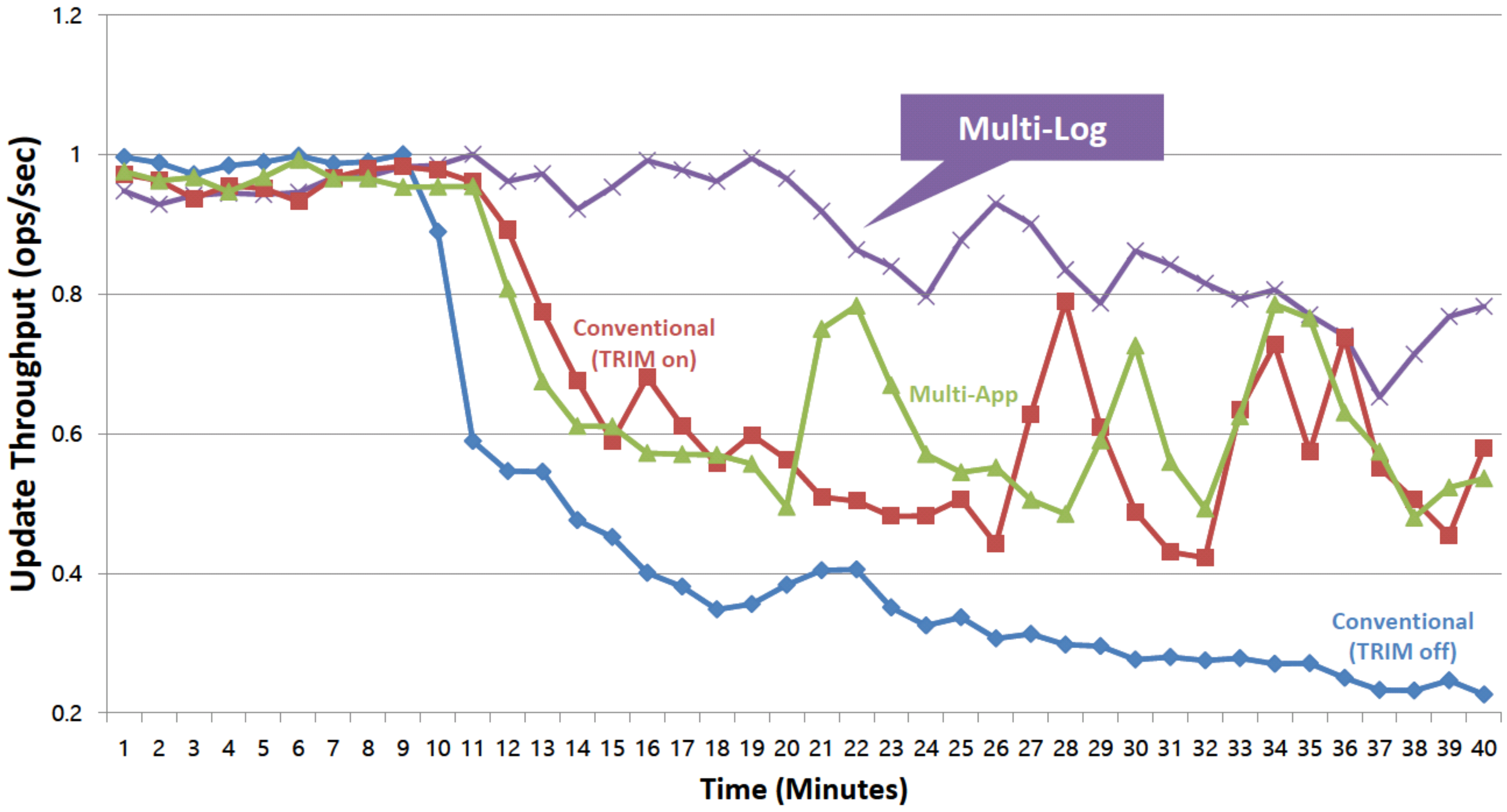
Results: Multi-App

- Cassandra's normalized update throughput
  - "Multi-App" (System data vs. Cassandra data)



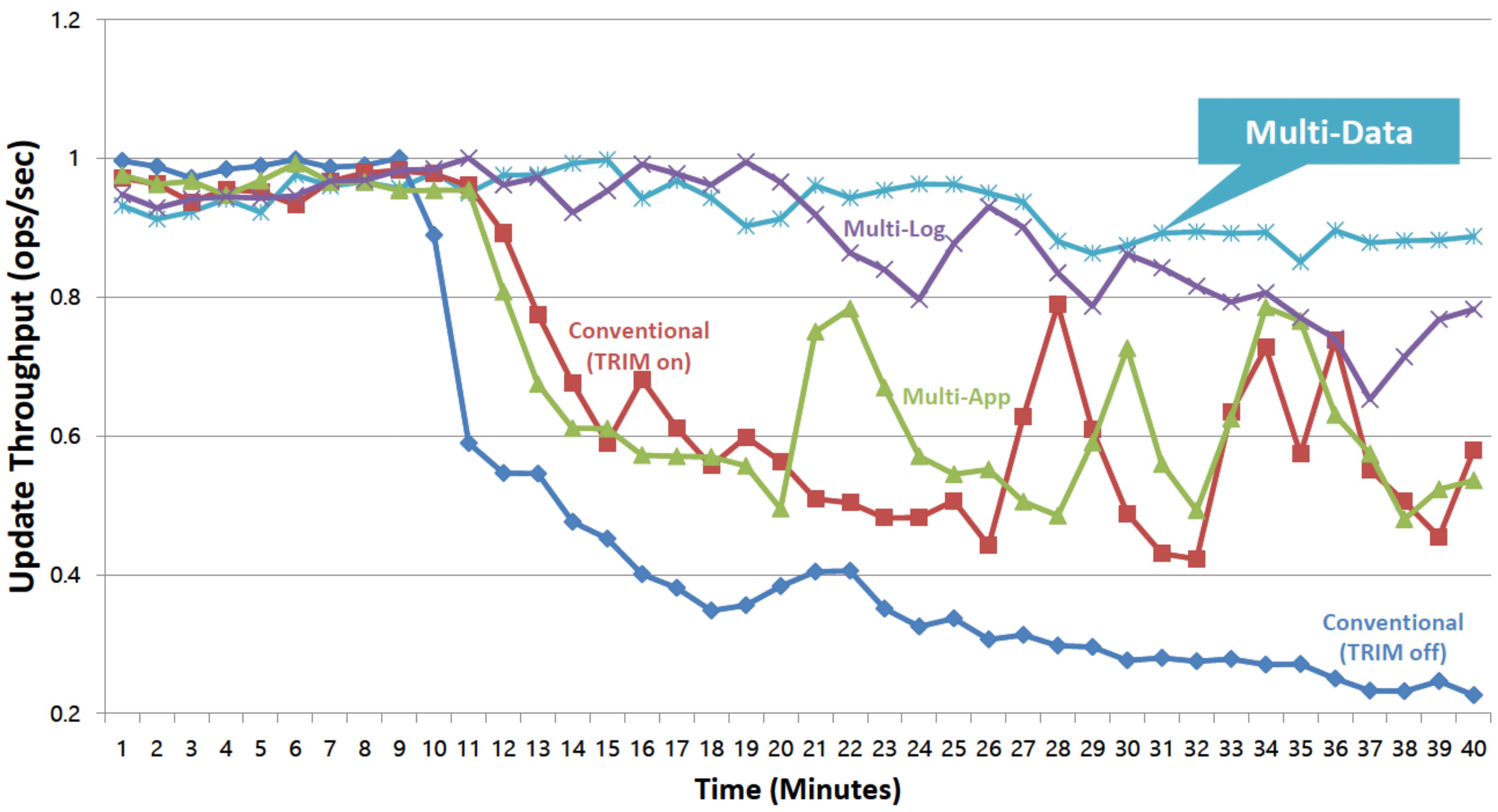
Results: Multi-Log

- Cassandra's normalized update throughput
  - "Multi-Log" (System data vs. Commit-Log vs. Flushed data)



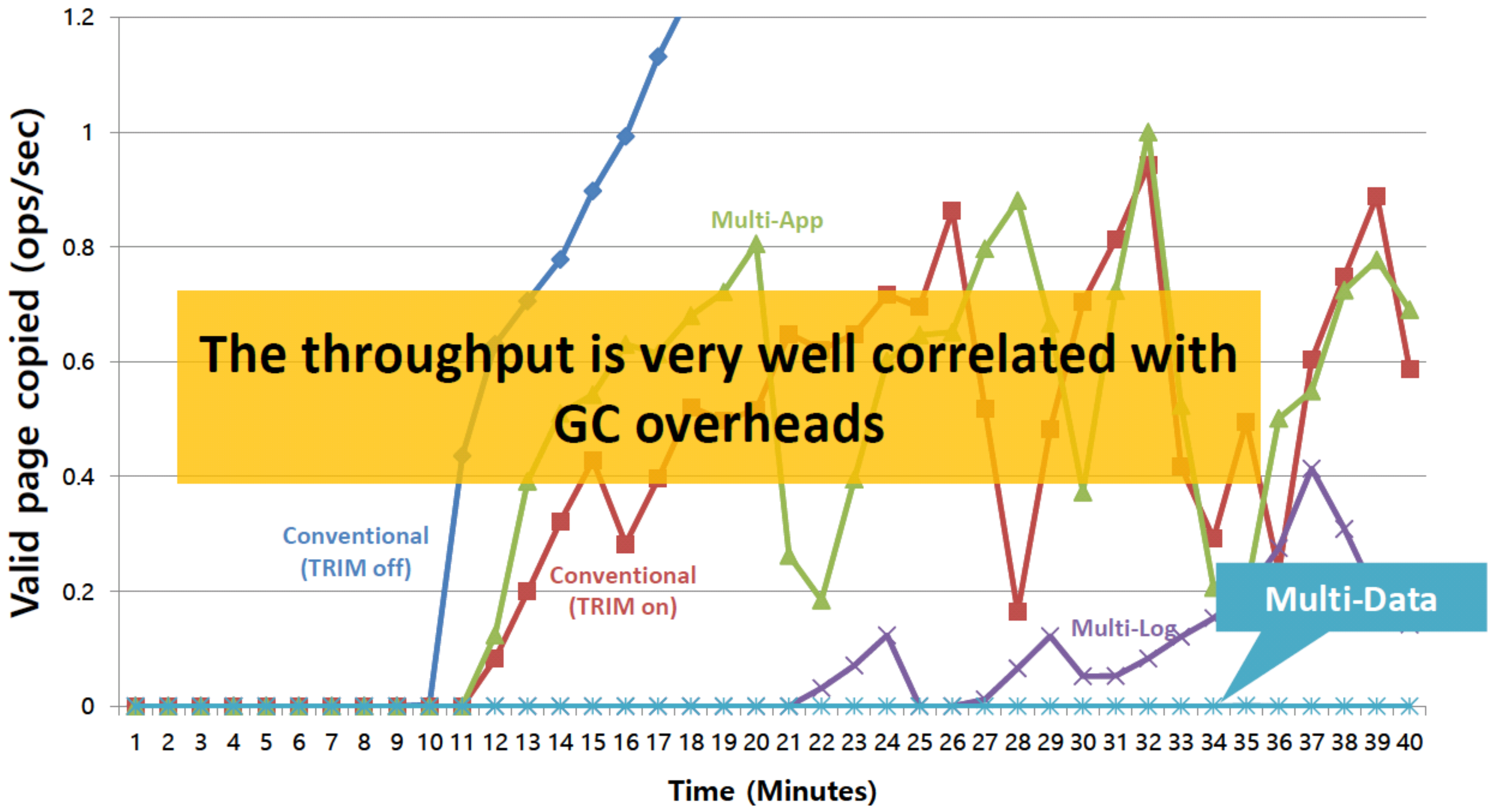
Results: Multi-Data

- Cassandra's normalized update throughput
  - "Multi-Data" (System data vs. Commit-Log vs. Flushed data vs. Compaction Data)



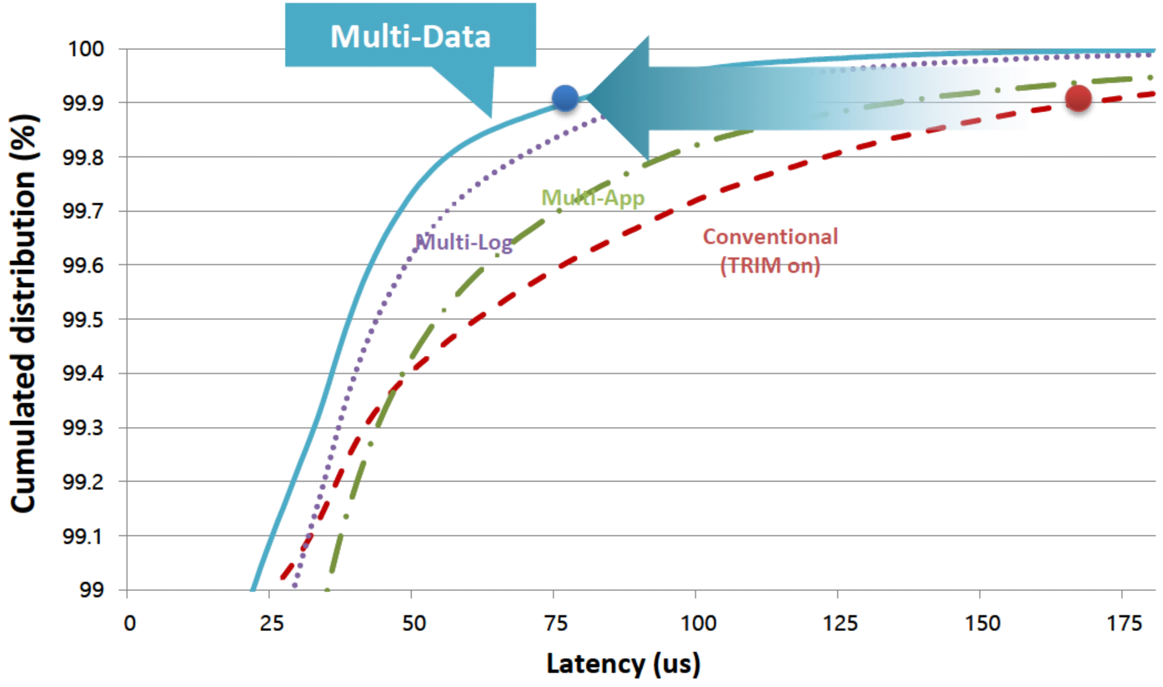
Results: GC Overheads

- Cassandra's GC overheads



Results: Latency

- Cassandra's cumulated latency distribution
  - Multi-streaming improves write latency
  - At 99.9%, Multi-Data lowers the latency by 53% compared to Normal

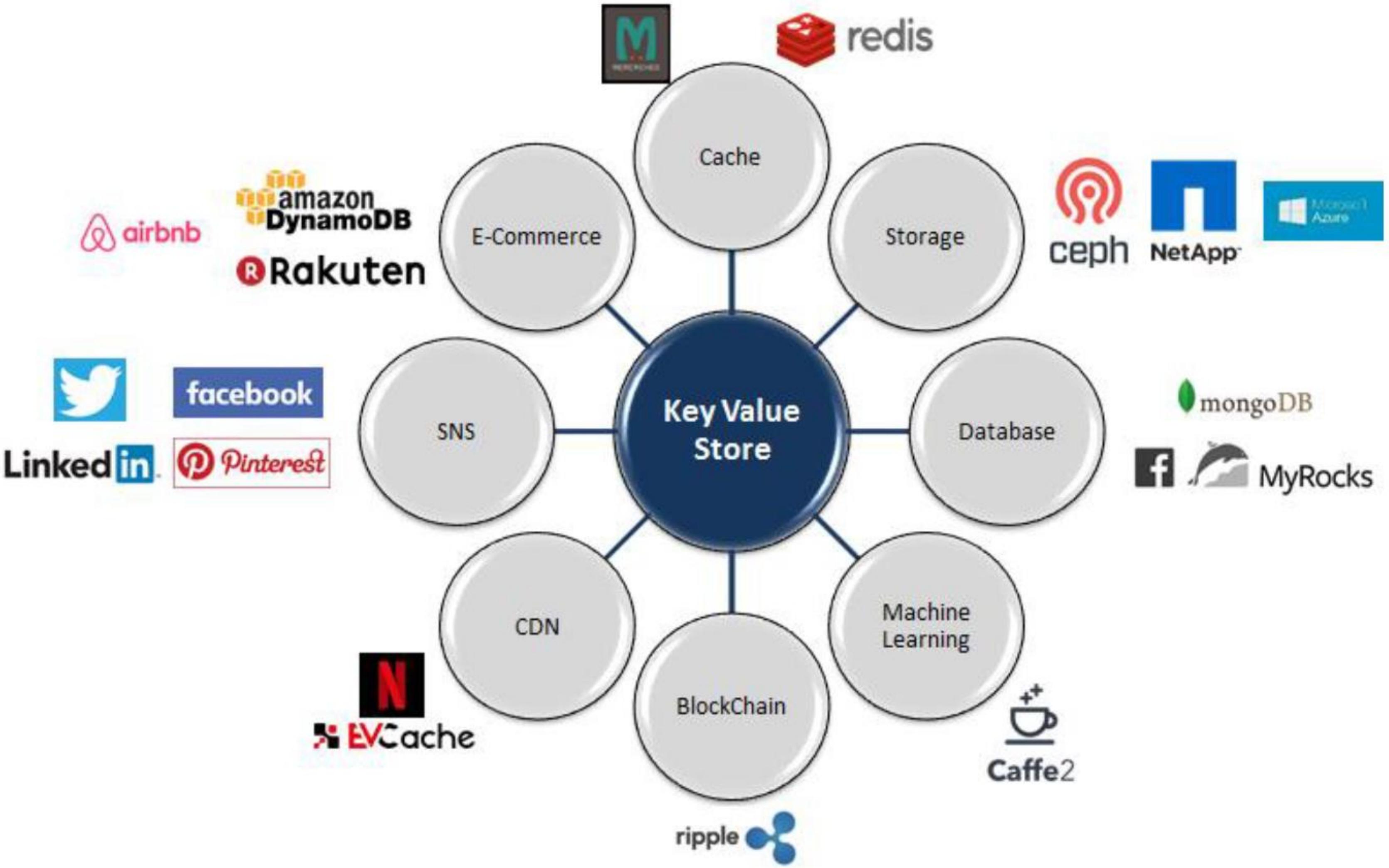


Summary

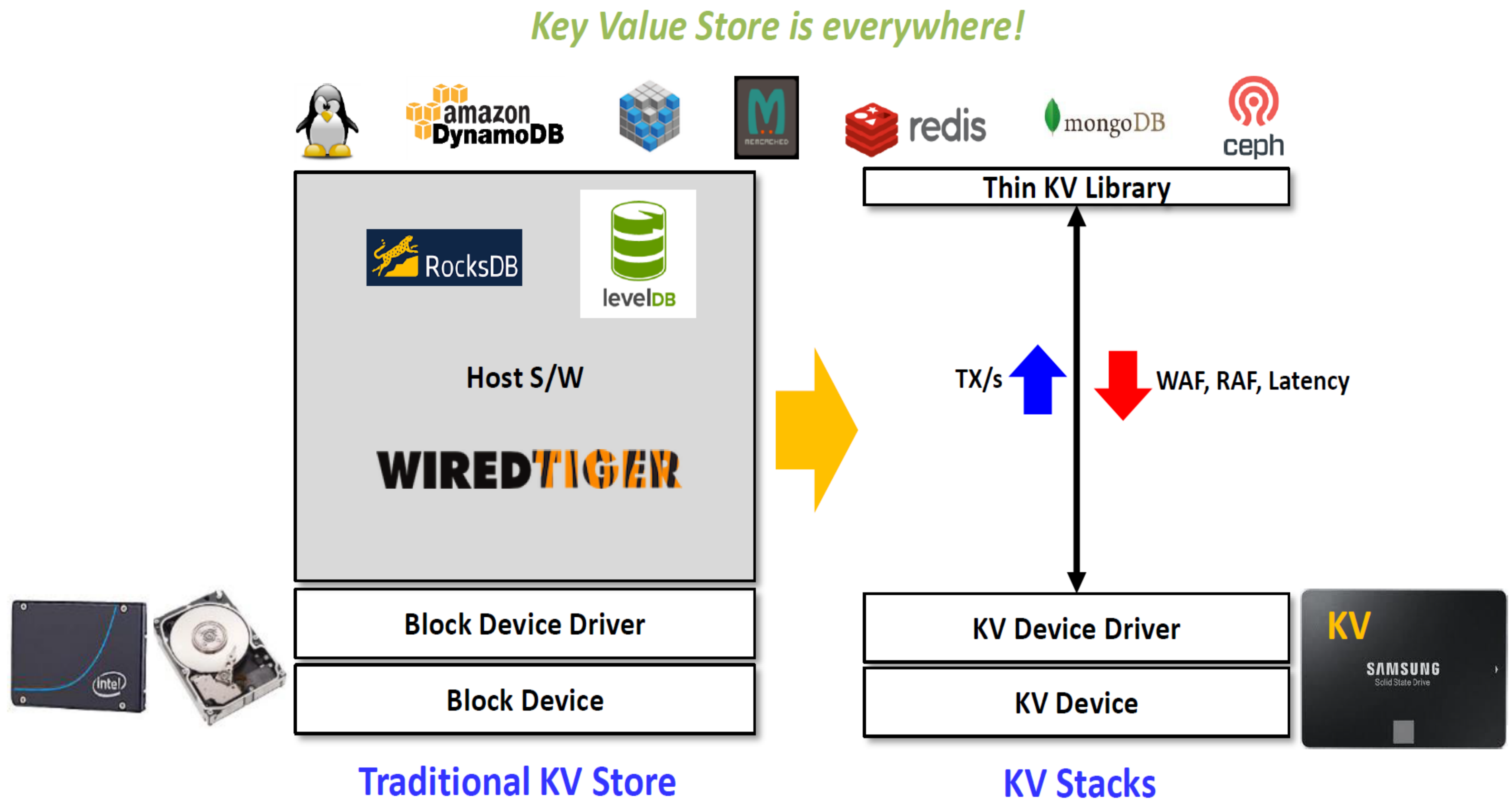
- Mapping application and system data with different lifetimes to SSD streams
  - Higher GC efficiency, lower latency
- Multi-streaming can be supported on a state-of-the-art SSD and co-exist with the traditional block interface
- Standardized in T10 SCSI (SAS SSDs) in 2015
- Standardized in NVMe 1.3 in 2017

Key-Value SSD (KVSSD)

KV Stores Common in Systems at Scale



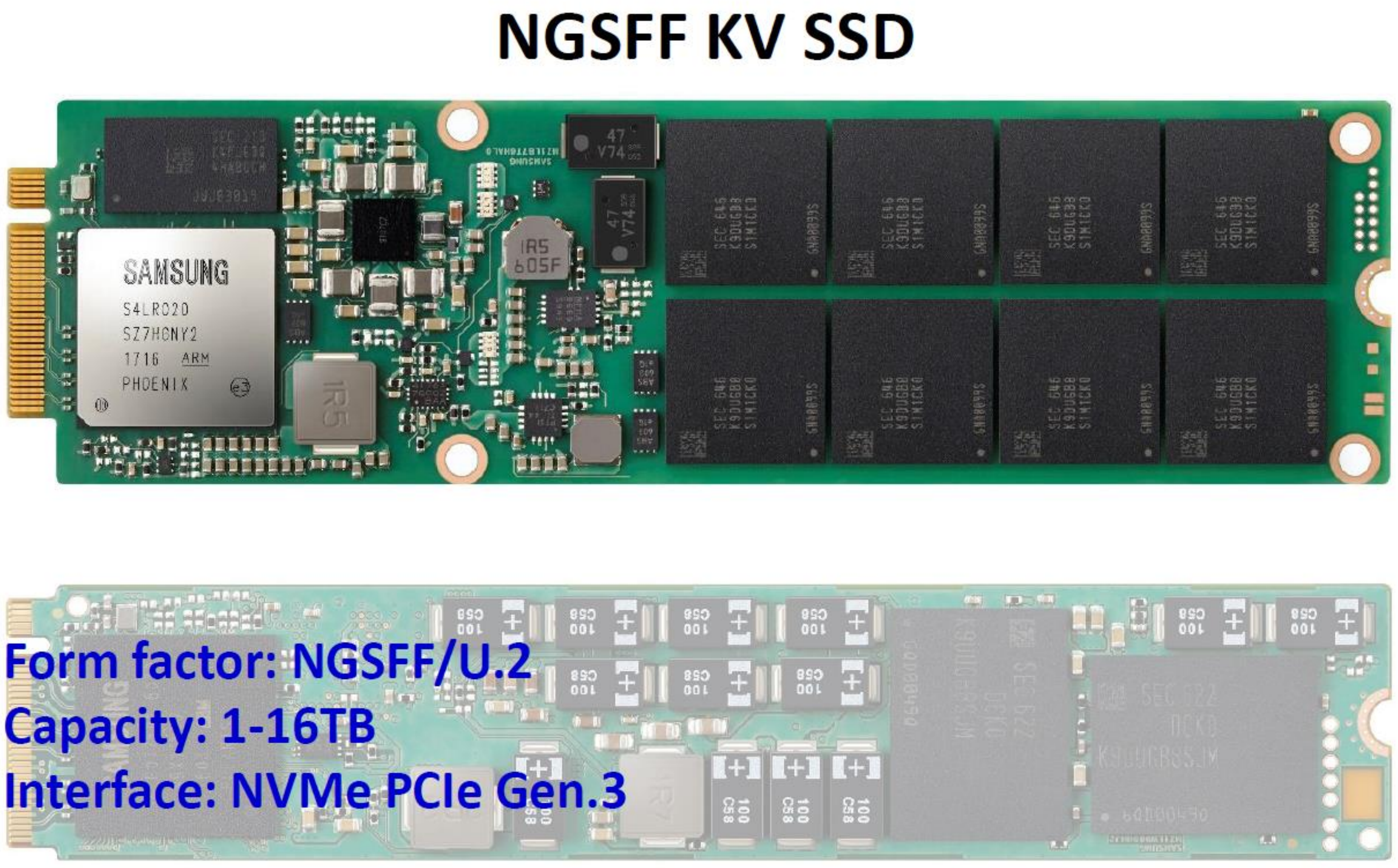
Why KVSSD?



Key Value Store is everywhere!
amazon DynamoDB
redis
mongoDB
ceph
RocksDB
levelDB
Host S/W
WIREDTIGER
Block Device Driver
Block Device
Thin KV Library
TX/s
WAF, RAF, Latency
KV Device Driver
KV Device
KV
SAMSUNG
Traditional KV Store
KV Stacks

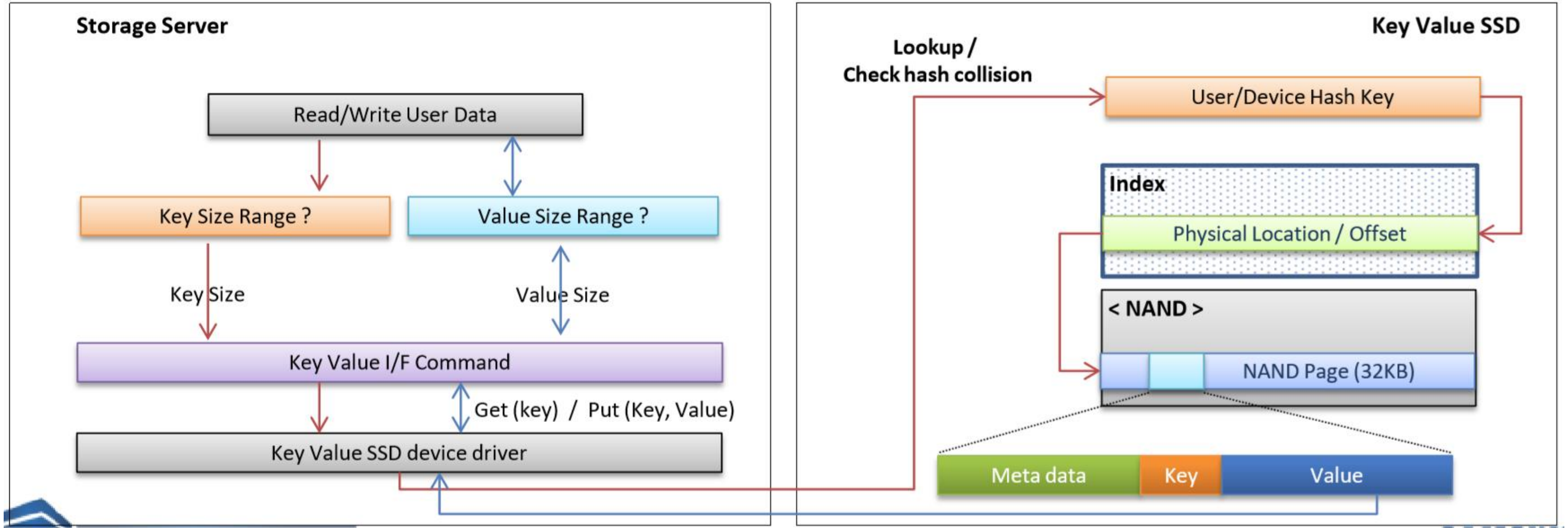
KVSSD Prototype

- Samsung KV-PM983



KVSSD Design

- Key size: up to 255B
- Value size: up to 2MB
- https://github.com/OpenMPDK/KVSSD



Open-Channel SSD (OCSSD)

Why OCSSD?

I/O Isolation

Enable I/O isolation between tenants by allocating your SSD into separate parallel units.

Predictable Latency

No more guessing when an IO completes. You know which parallel unit is accessed on disk.

Data Placement & I/O Scheduling

Manage your non-volatile memory as a block device, through a file-system or inside your application.

https://platformlab.stanford.edu/Seminar%20Talks/Matias_Bj_rling___Javier_Gonzales.pdf

OCSSD Architecture