The Multi-streamed Solid-State Drive

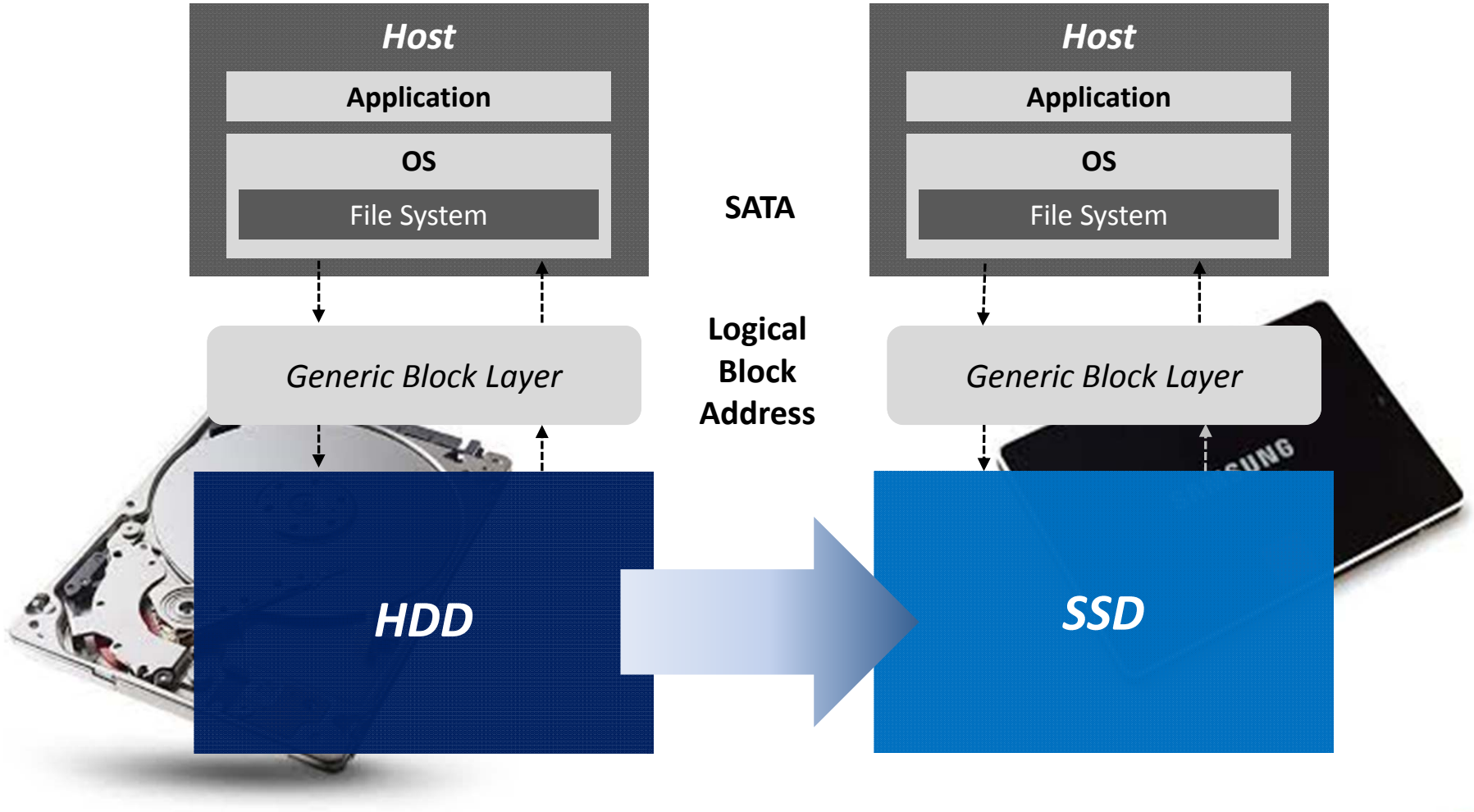
Jeong-Uk Kang*, Jeeseok Hyun, Hyunjoo Maeng, and Sangyeun Cho

Memory Solutions Lab.
Memory Division, Samsung Electronics Co., Ltd

SAMSUNG

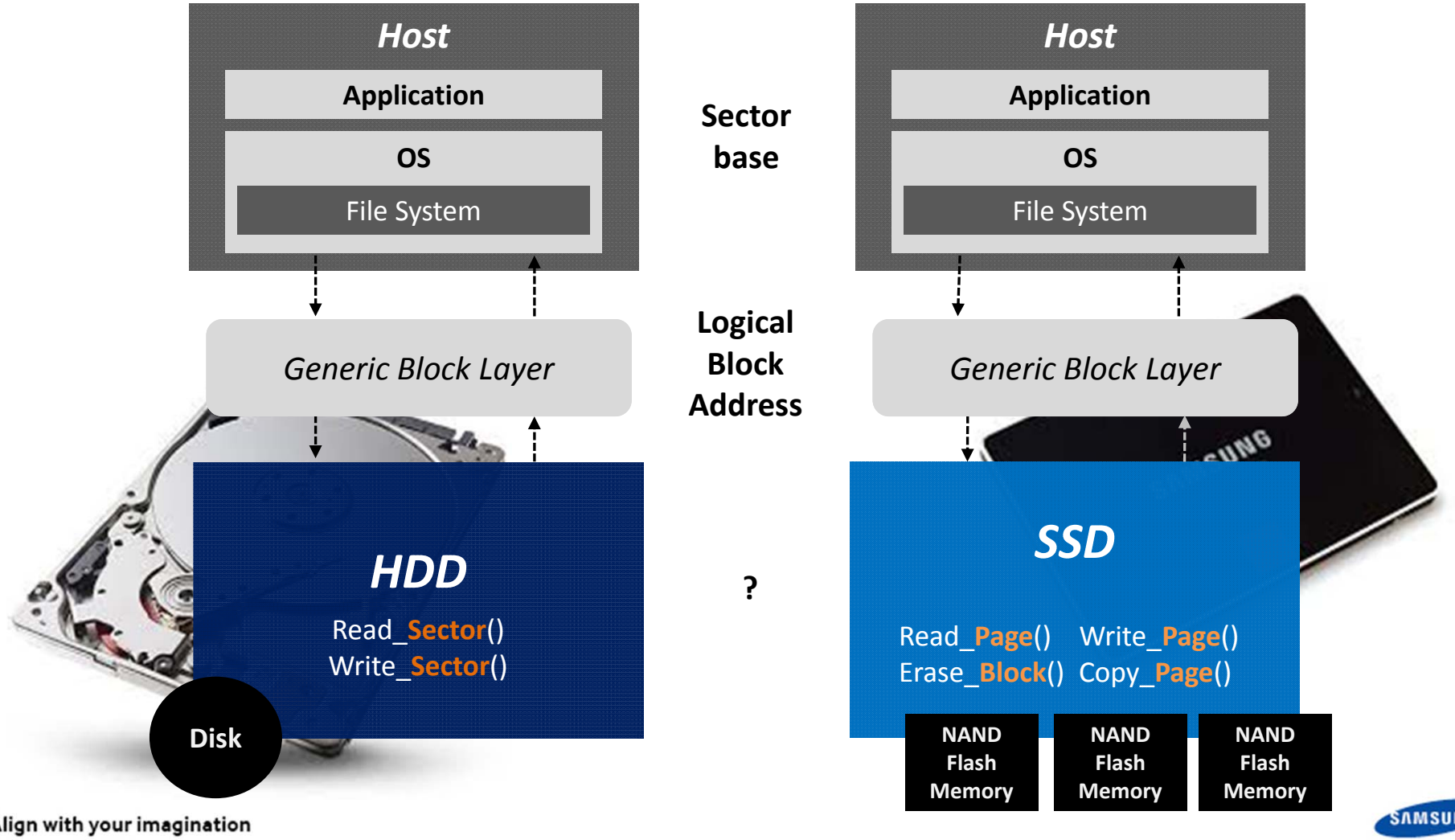
SSD as a Drop-in Replacement of HDD

- **SSD shares a common interface with HDD**
  - **The block device abstraction paved the way for wide adoption of SSDs**



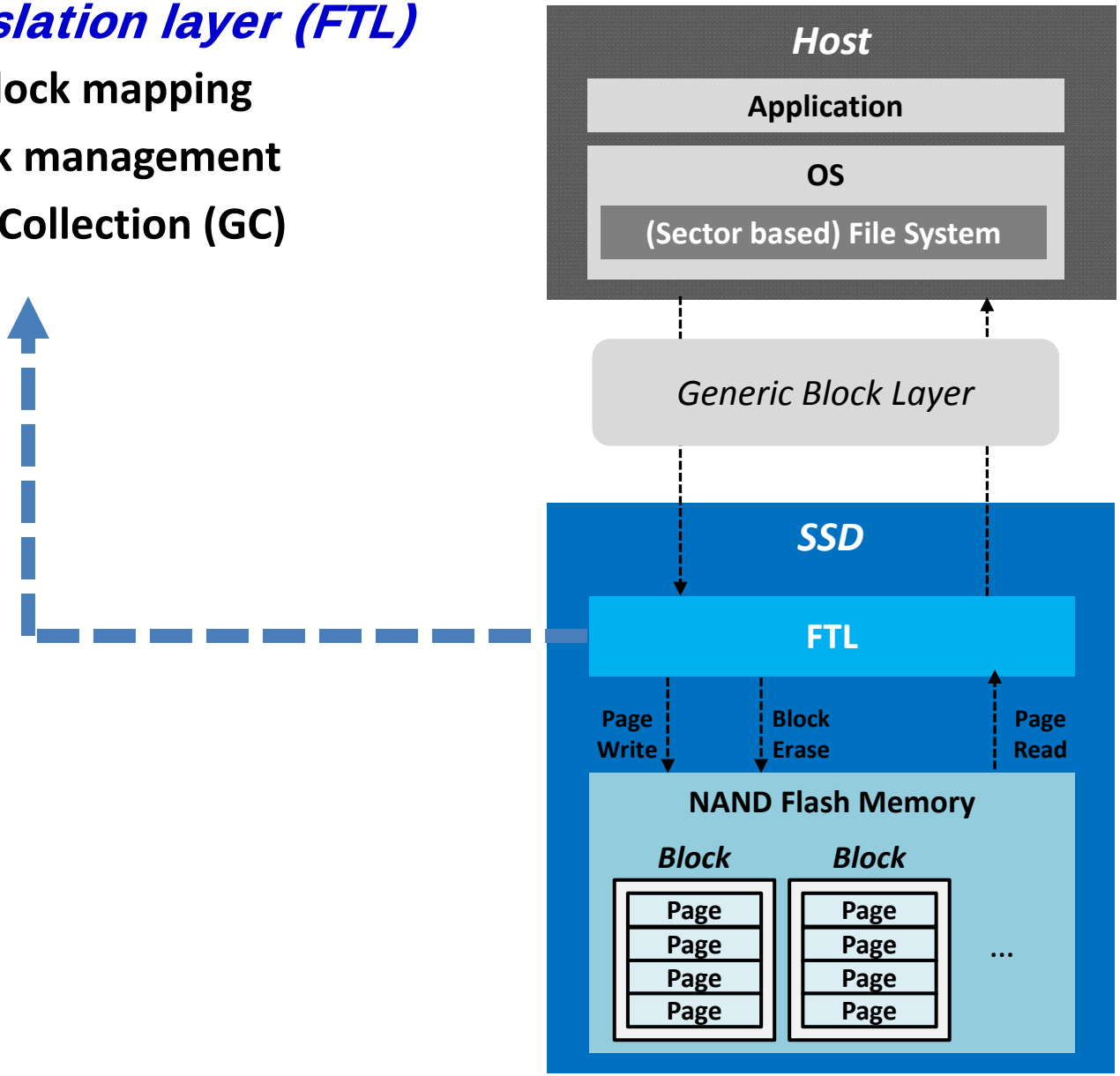
Great, BUT…

- **Rotating media and NAND flash memory are very different!**



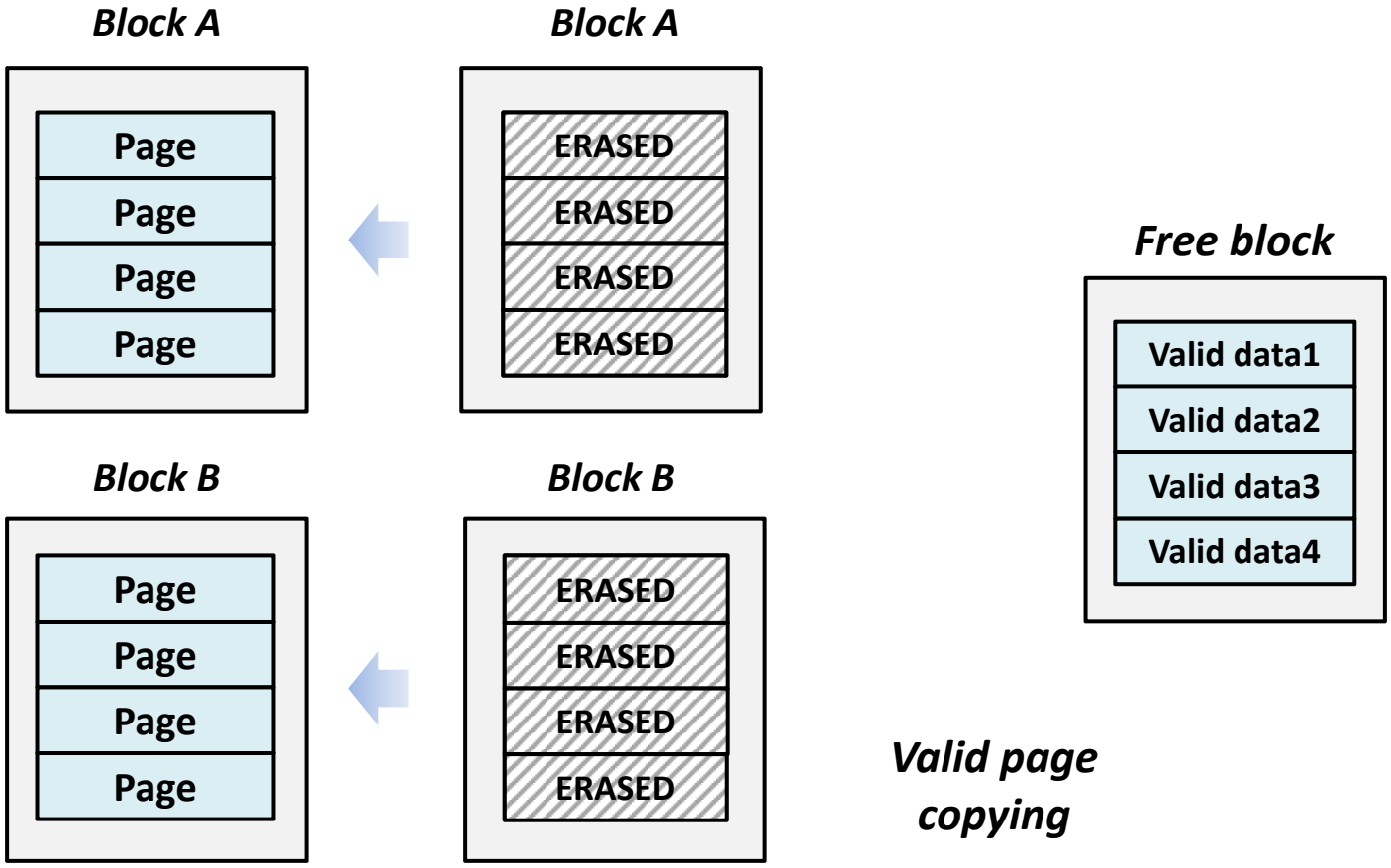
The Trick is FTL!

- ***Flash translation layer (FTL)***
  - Logical block mapping
  - Bad block management
  - Garbage Collection (GC)



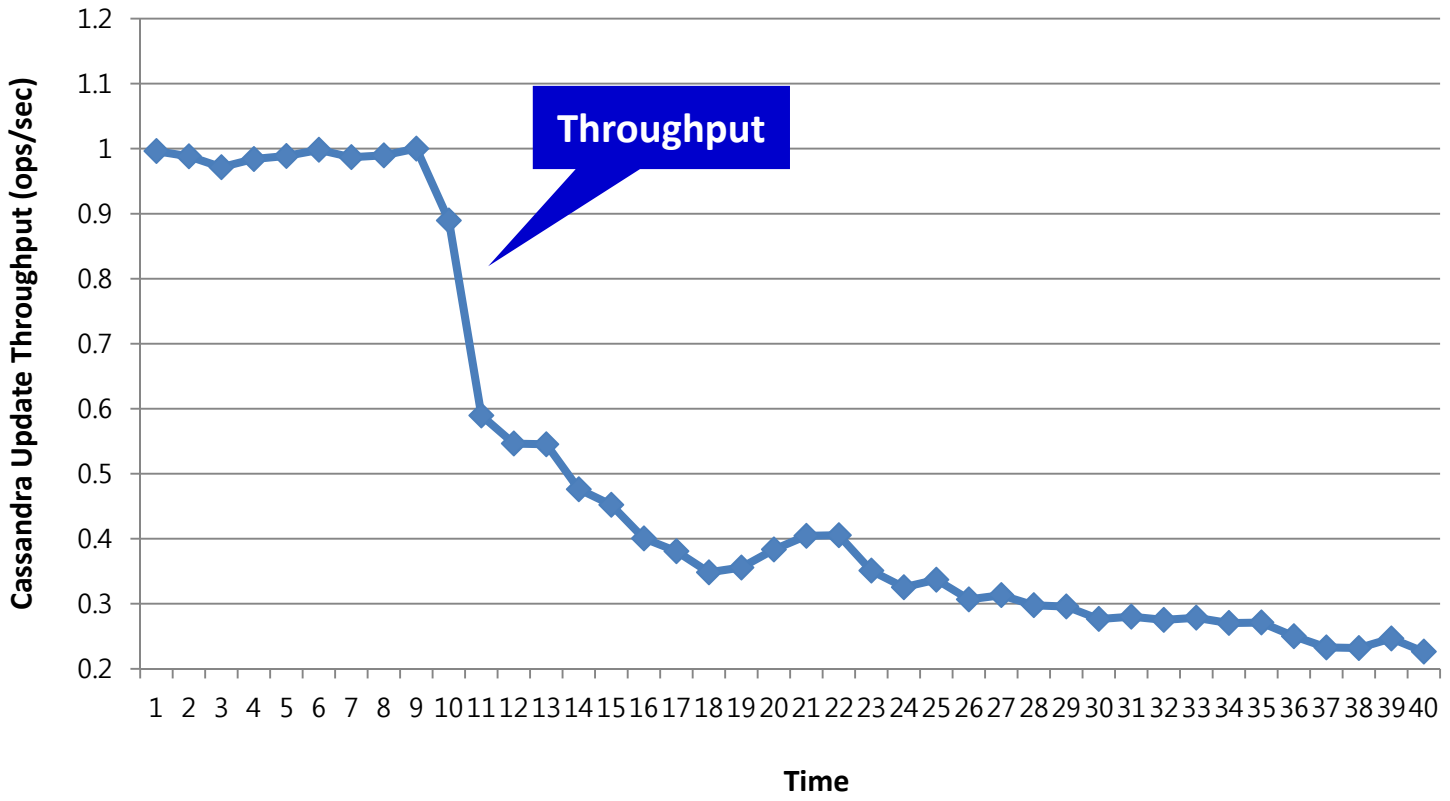
Garbage Collection (GC)

- **GC reclaims space to prepare new empty blocks**
  - NAND's "erase-before-update" requirement ⇒ Valid page copying followed by an erase operation
  - Has a large impact on SSD lifetime and performance



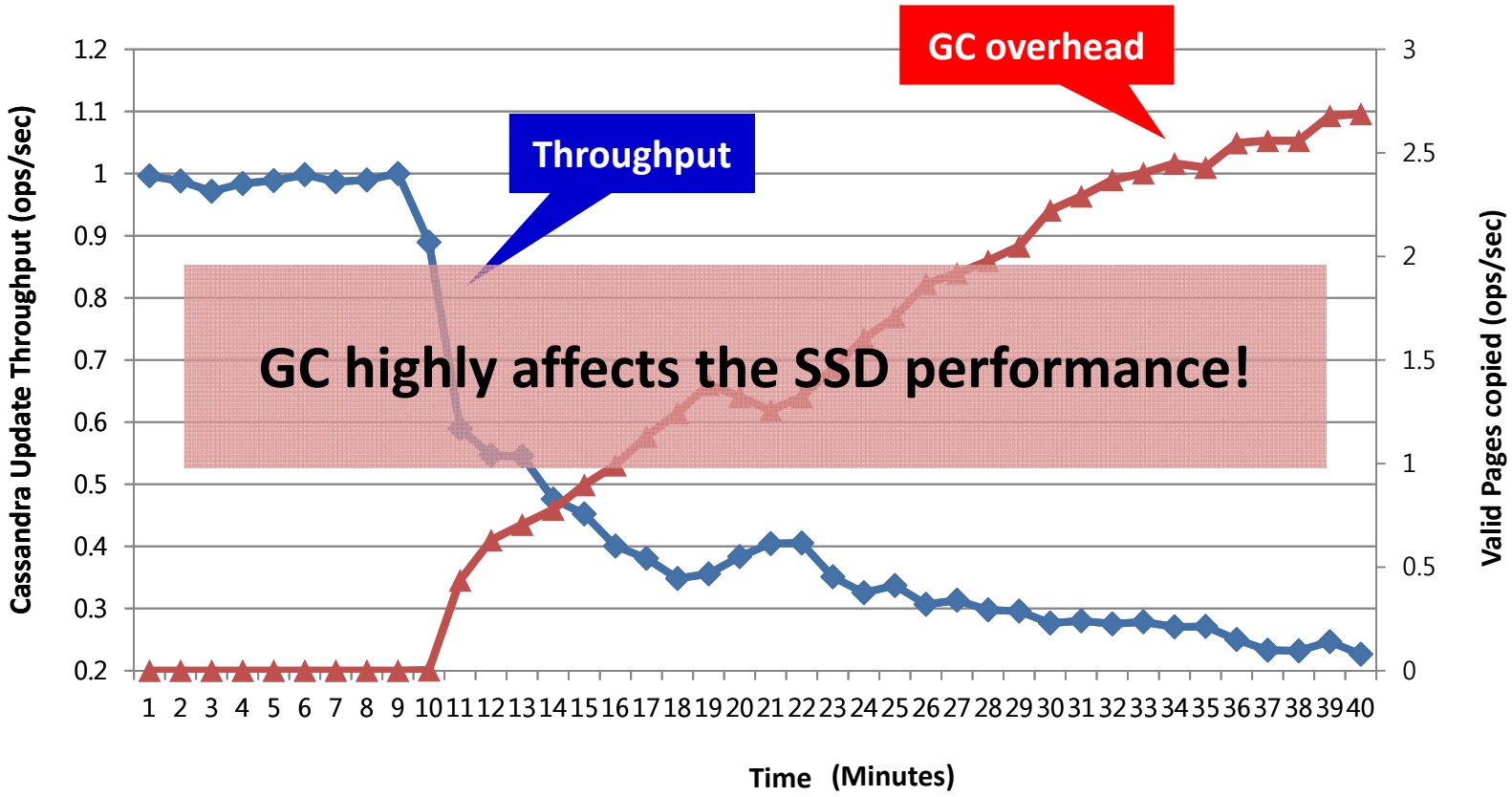
GC is Expensive!

- **Performance of SSD gradually decreases as time goes on**
  - **Example: Cassandra update throughput**

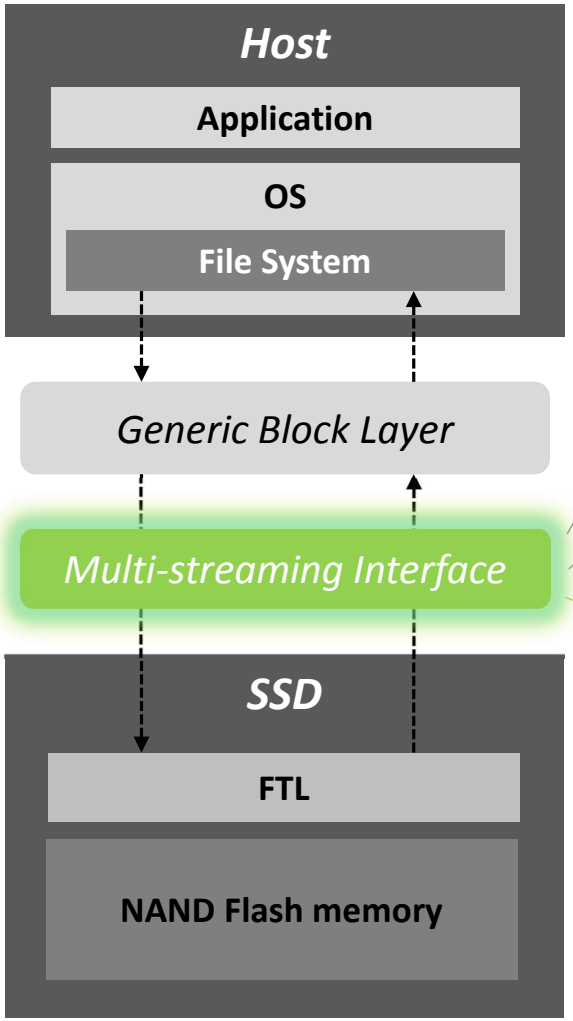


GC is Expensive!

- **Performance of SSD gradually decreases as time goes on**
  - **Example: Cassandra update throughput**



Our Idea: Multi-streamed SSD



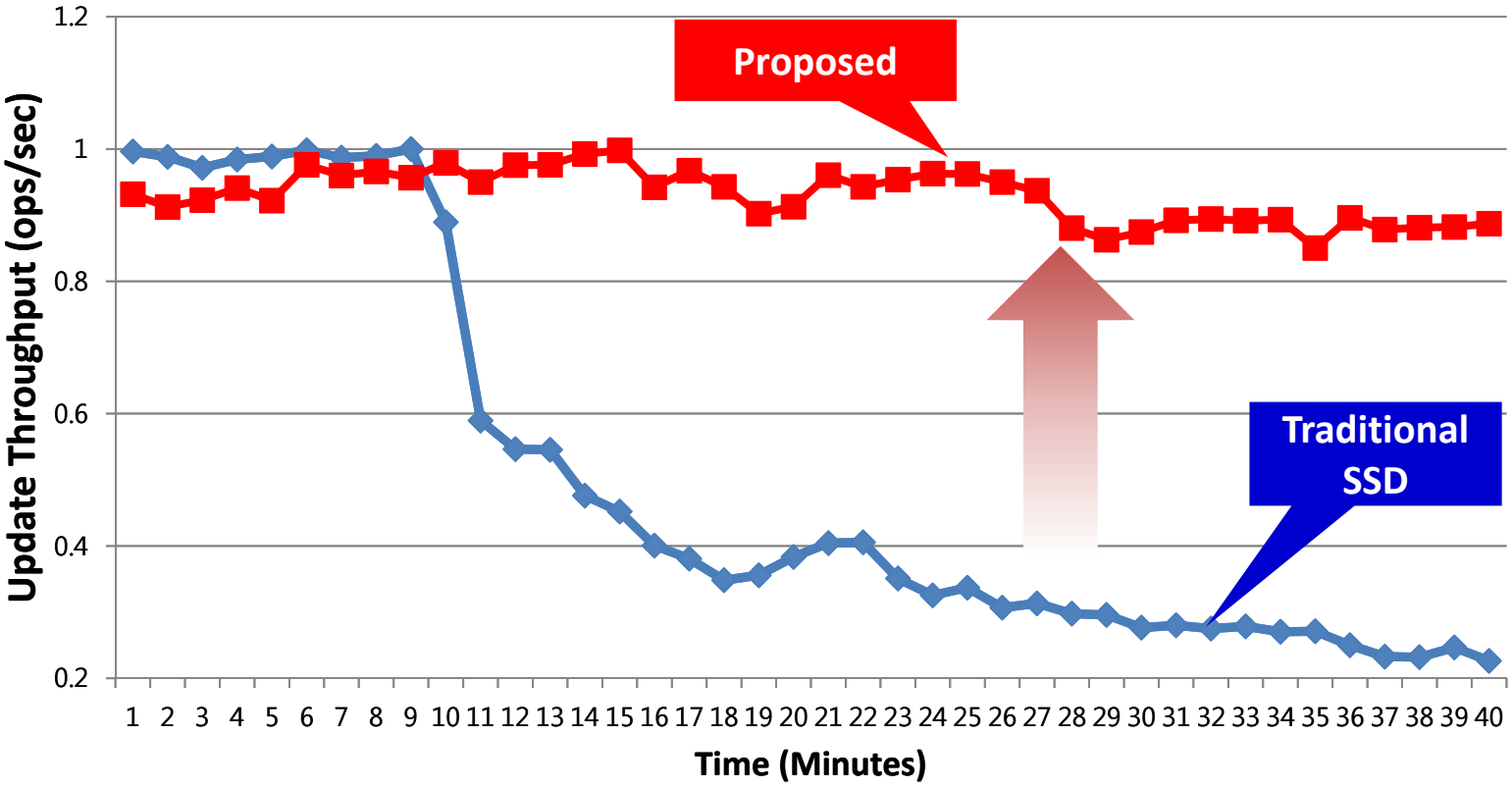
Co-exists with the existing block layer

General & concrete interface

Host-provided stream information guides desirable data placement within SSD!

End Result

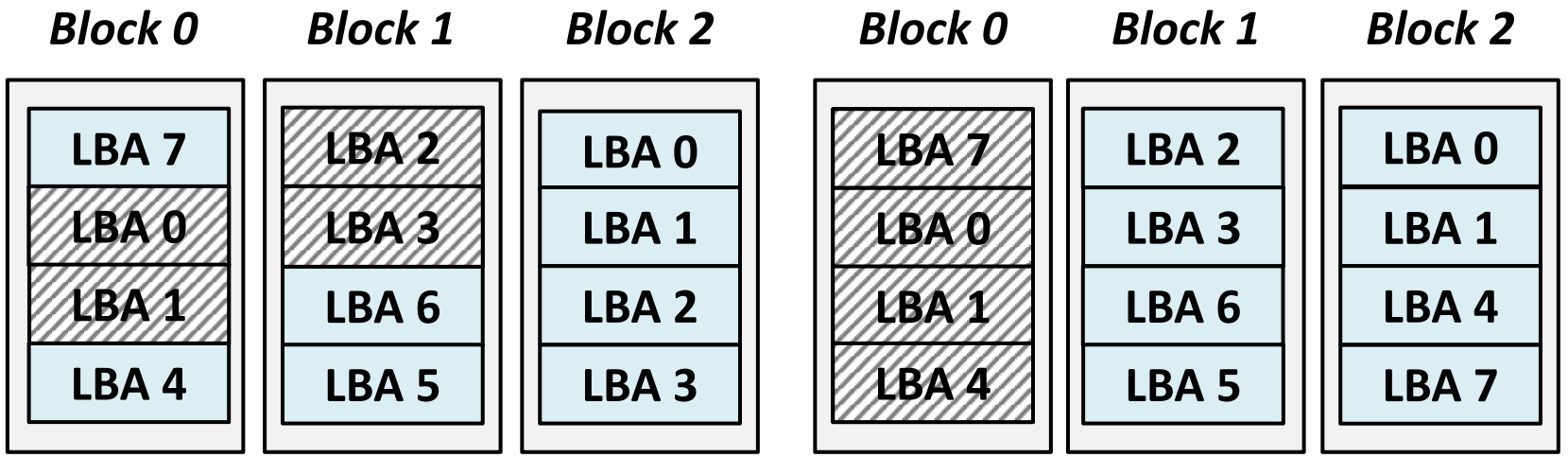
- **The multi-streamed SSD can sustain Cassandra update throughput**



Contents

Effects of Write Patterns

- **Previous write patterns (=current state) matter**



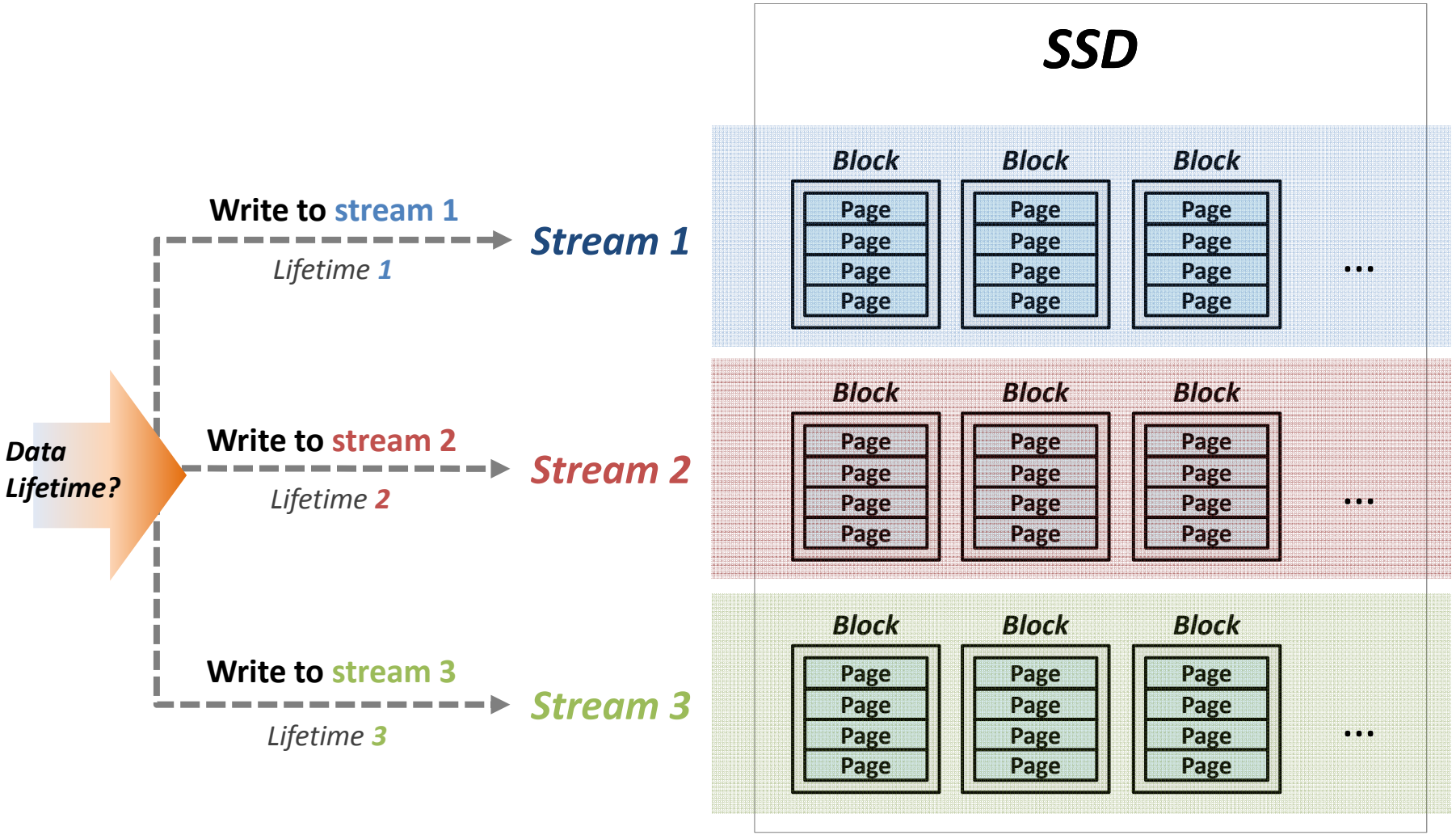
Sequential LBA updates into Block 2

Need valid page copying from Block 0 & Block 1

Random LBA updates into Block 2

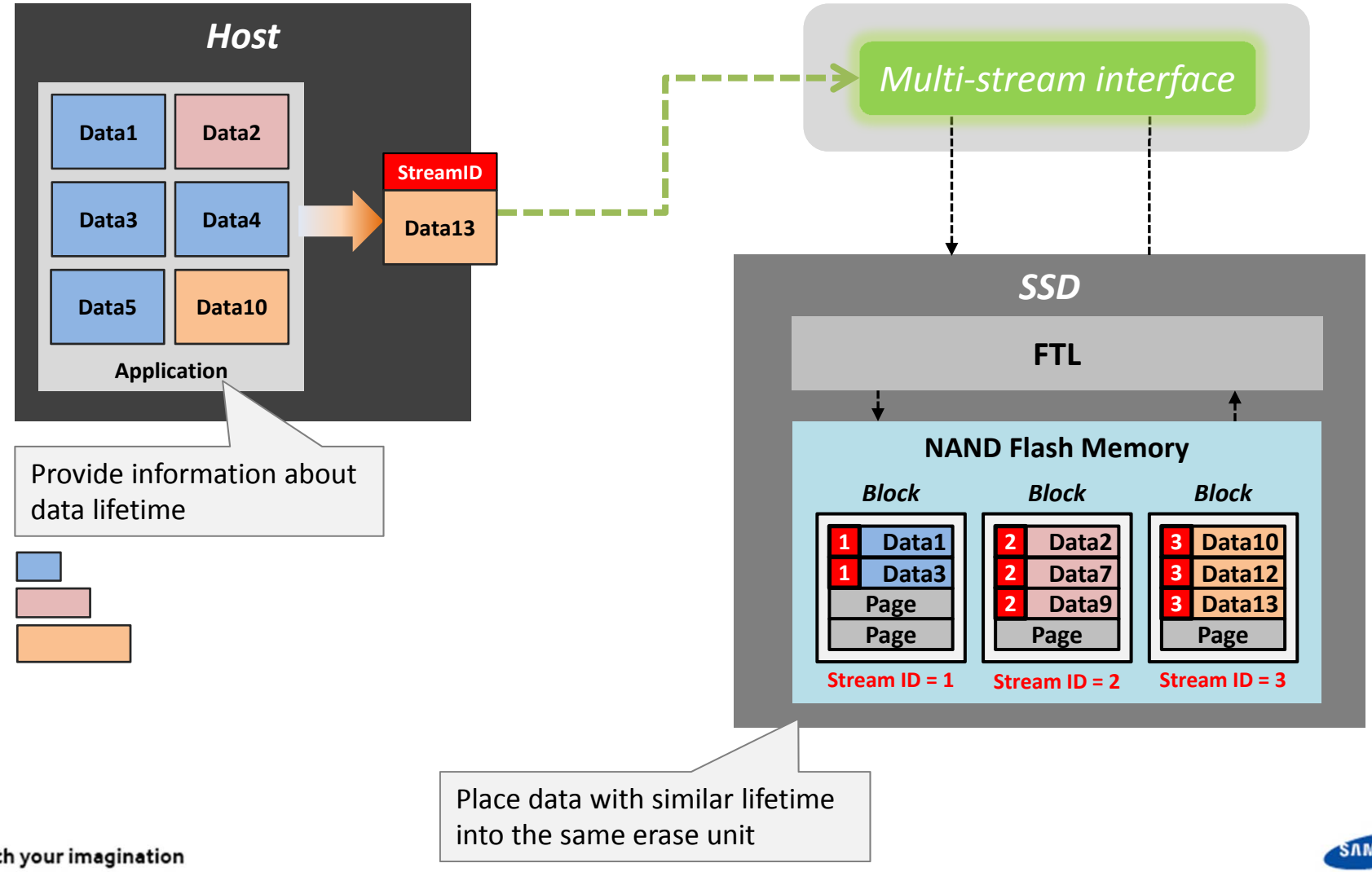
Just erase Block 0

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
...

Multi-streamed SSD

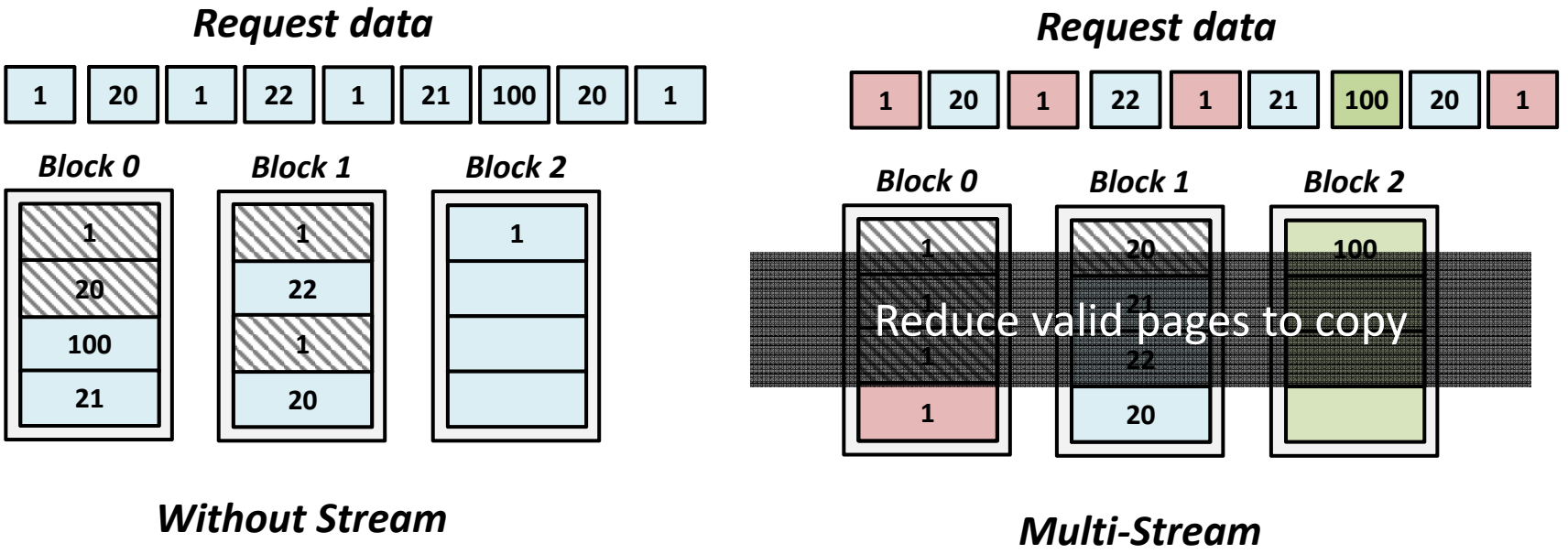
- Mapping data with different lifetime to different streams



Working Example

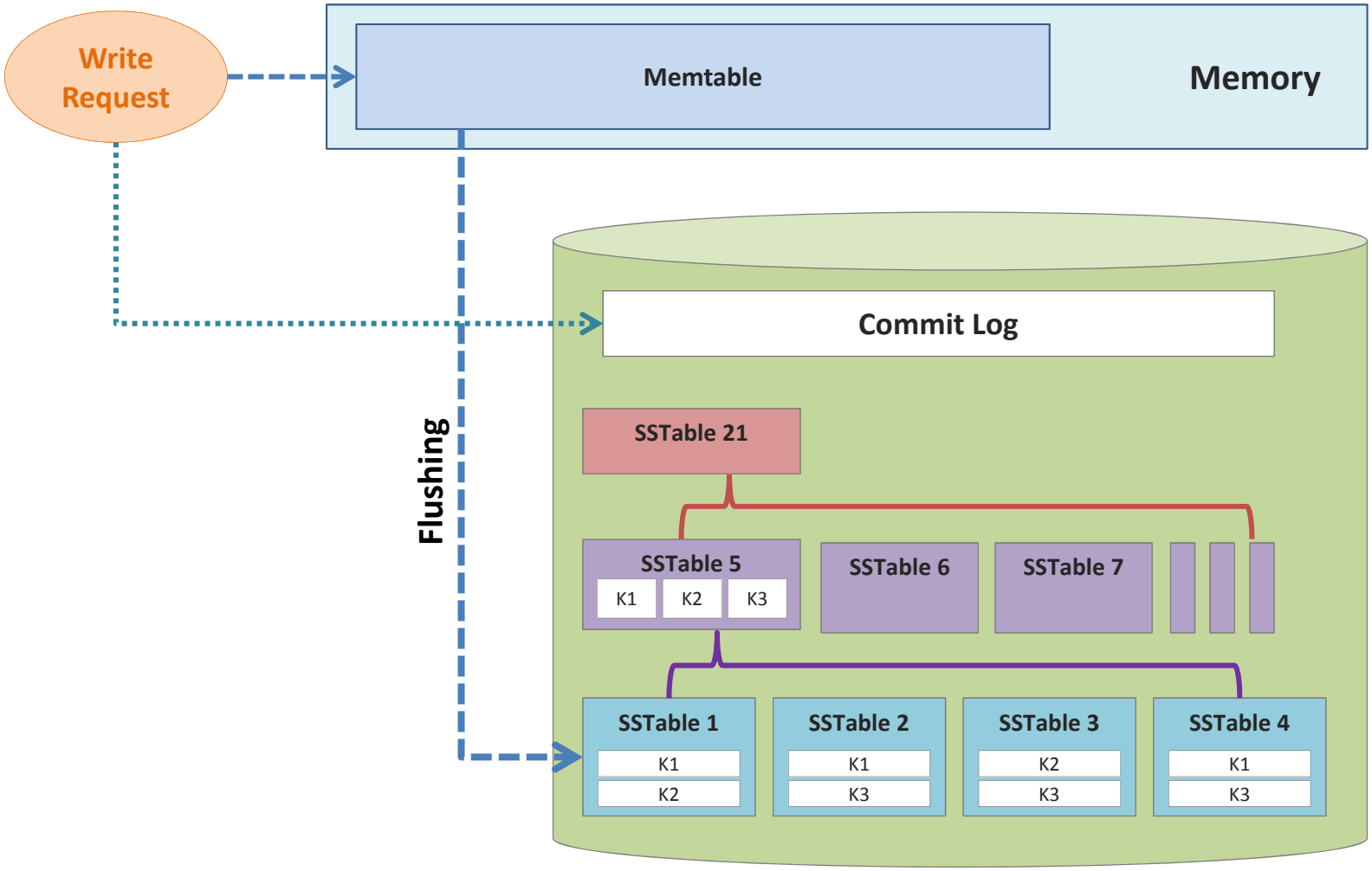
Multi-streamed SSD

- High GC efficiency (Reduce GC overheads) → effects on Performance!



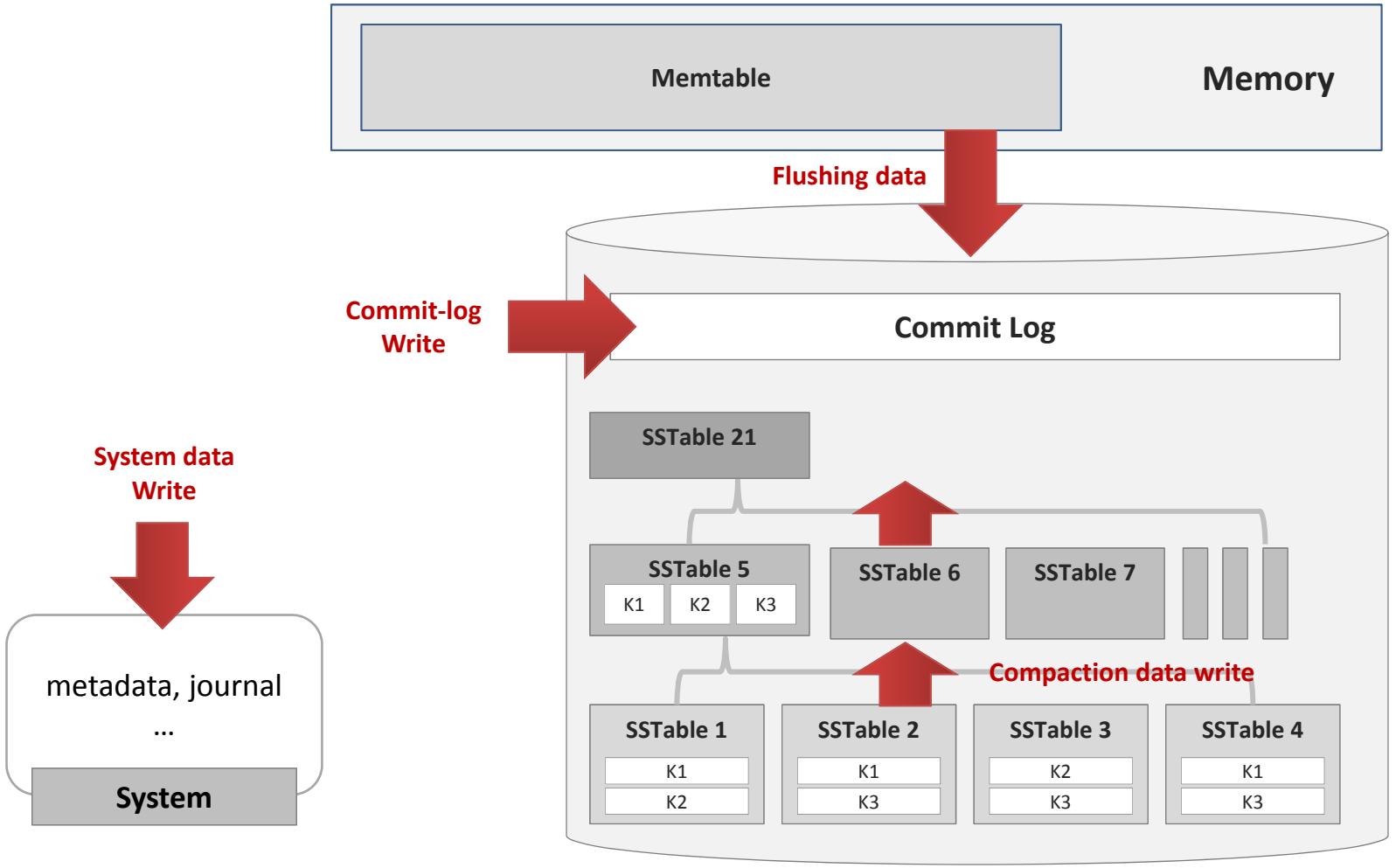
For effective multi-streaming,
proper mapping of data to streams is essential!

- **Cassandra employs a size-tiered compaction strategy**



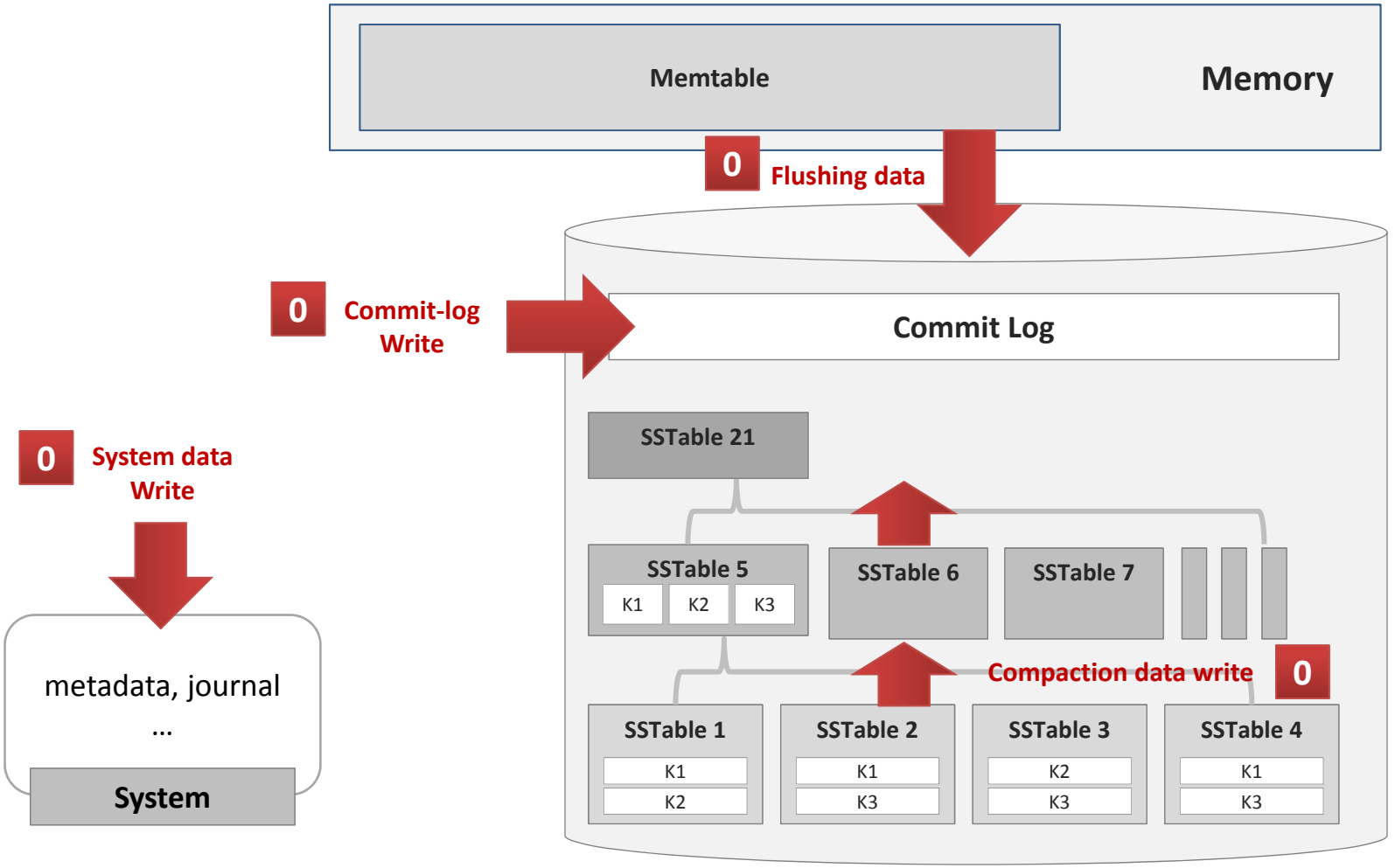
Summary of Cassandra's Write Patterns

Write operations when Cassandra runs



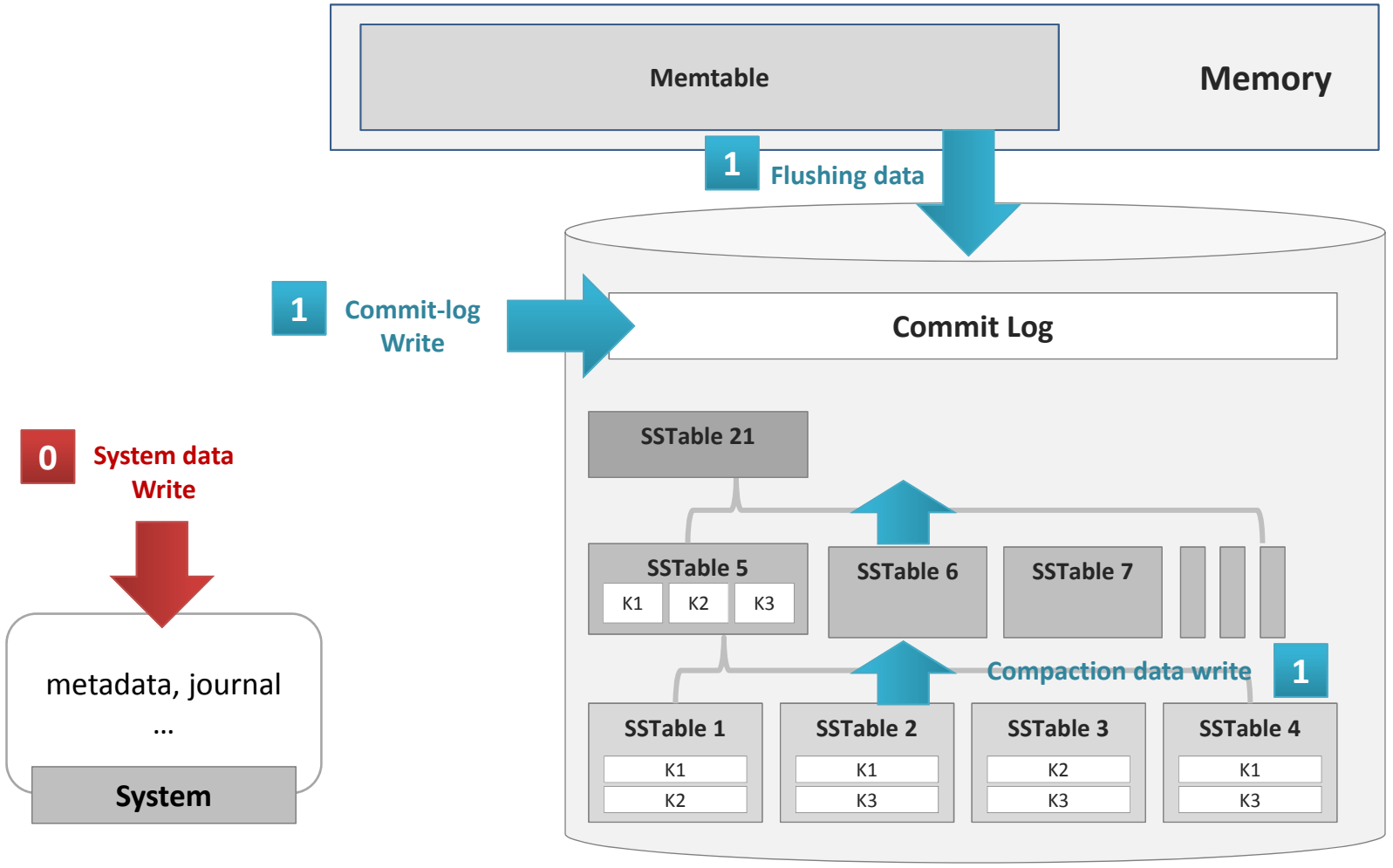
Mapping #1: “Conventional”

- **Just one stream ID (= conventional SSD)**



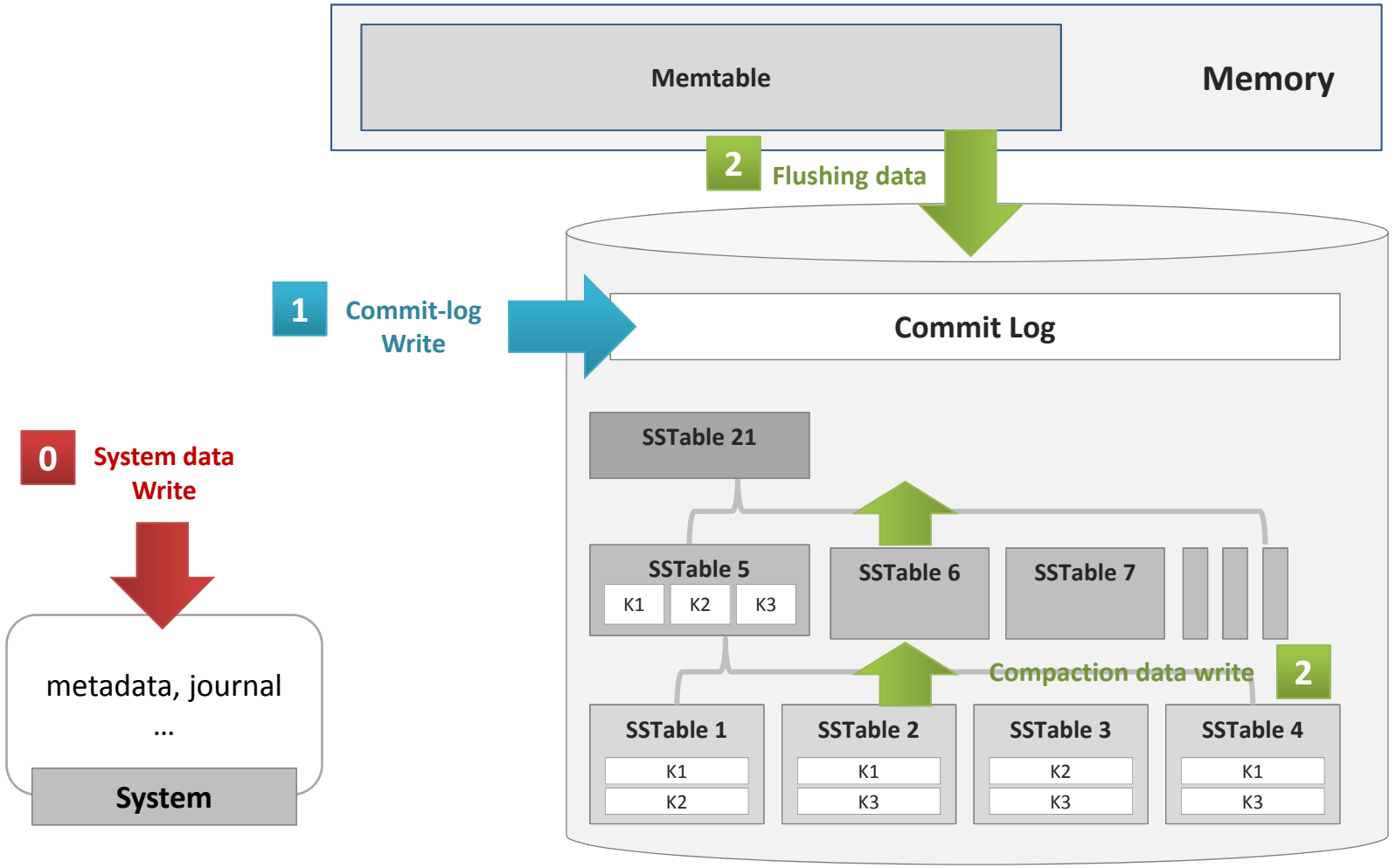
Mapping #2: “Multi-App”

- **Add a new stream to separately handle application writes (stream ID 1) from system traffic (stream ID 0)**



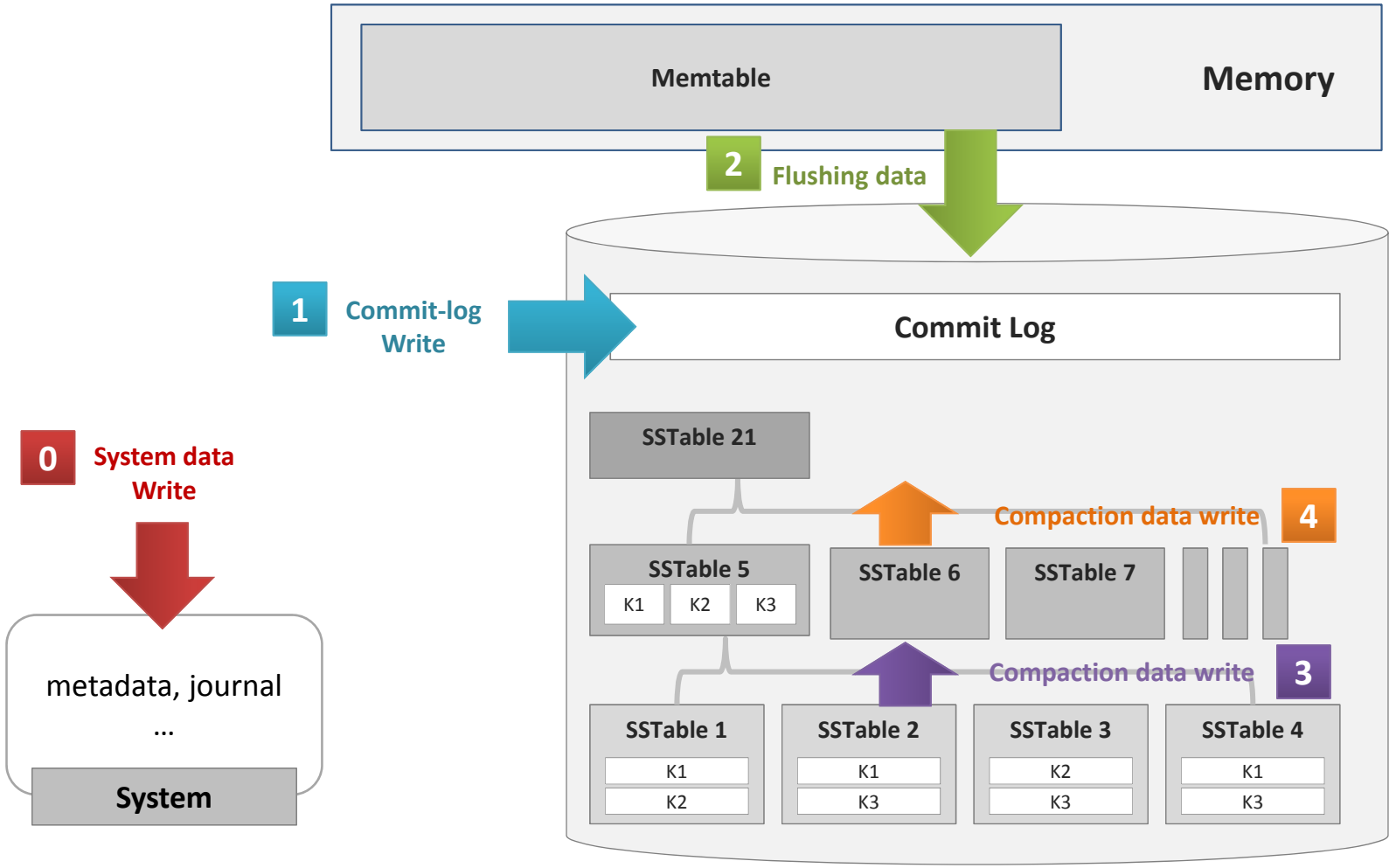
Mapping #3: “Multi-Log”

- **Use three streams; further separate Commit Log**



Mapping #4: “Multi-Data”

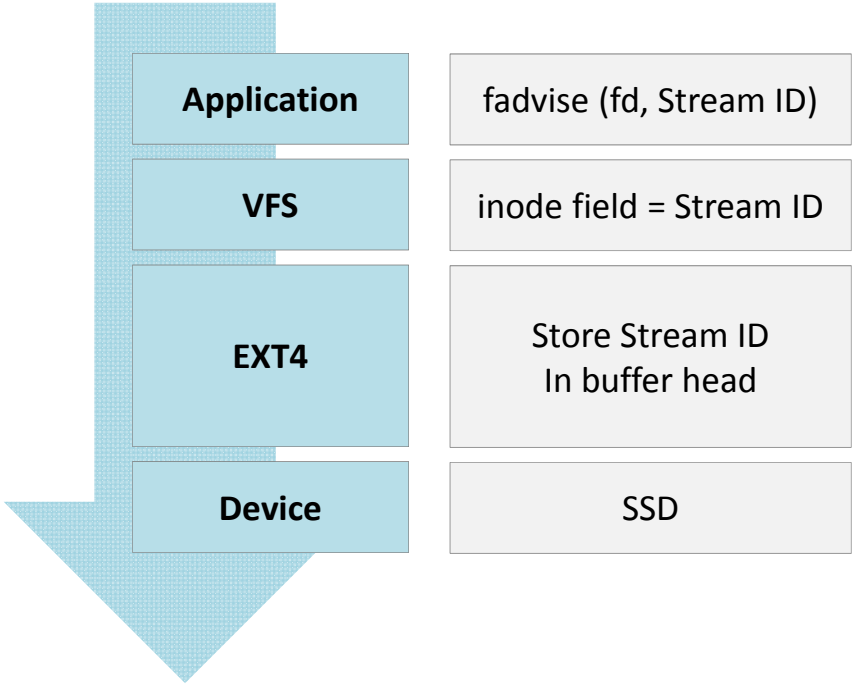
- **Give distinct streams to different tiers of SSTables**



Experimental Setup

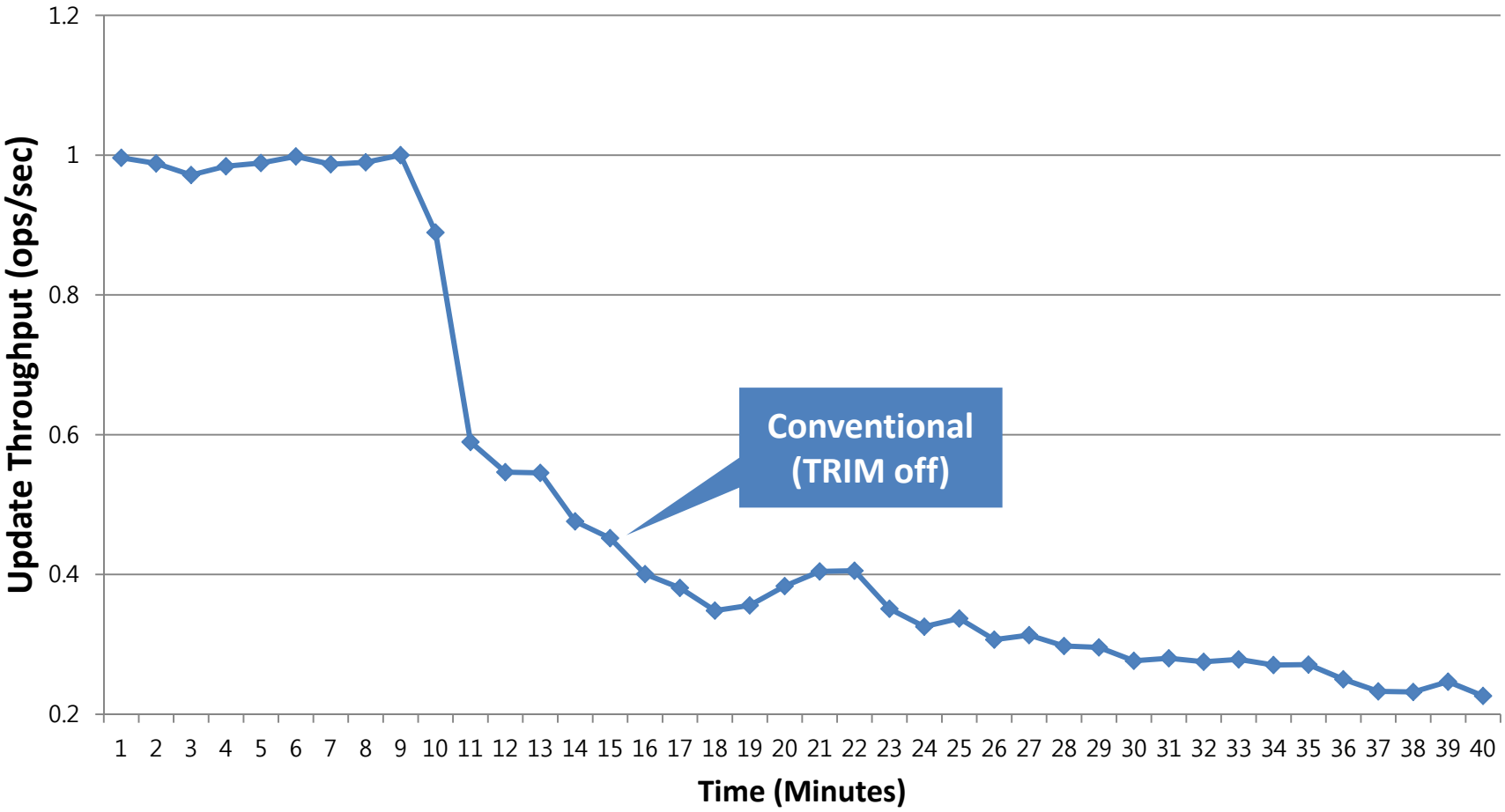
- **Multi-stream SSD Prototype**
  - **Samsung 840 Pro SSD**
    - 60 GB device capacity
- **YCSB benchmark on Cassandra**
  - **Write intensive workload**
    - 1 K data x 1,000,000 record counts
    - 100,000,000 operation counts
- **Intel i7-3770 3.4 GHz processor**
- **2 GB Memory**
  - **Accelerates SSD aging by increasing Cassandra's flush frequency**

- **Linux kernel 3.13 (modified)**
  - **Passes the stream ID through fadvise() system call**
  - **Stores in the inode of VFS**



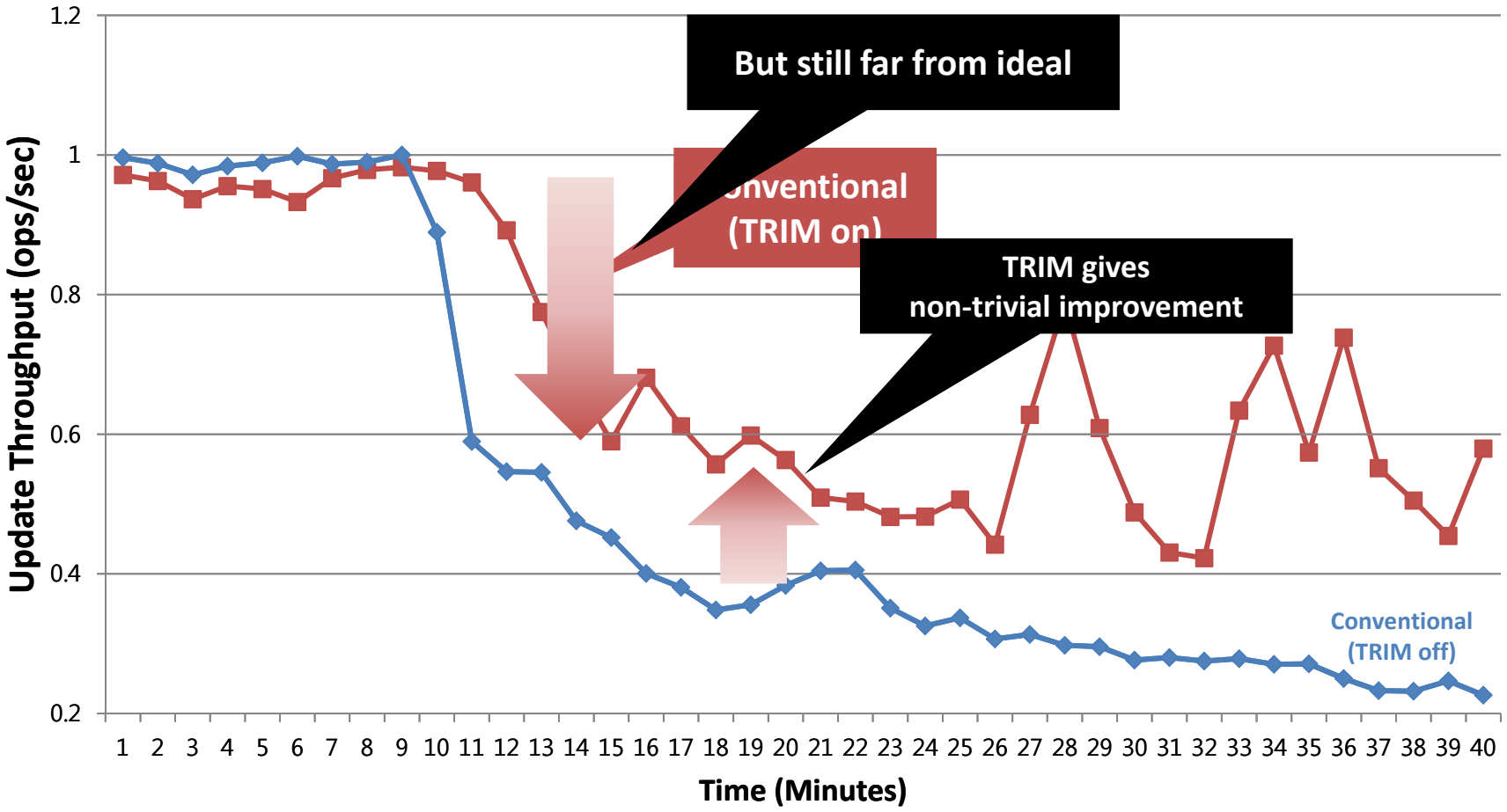
Cassandra's normalized update throughput

- Conventional "TRIM off"



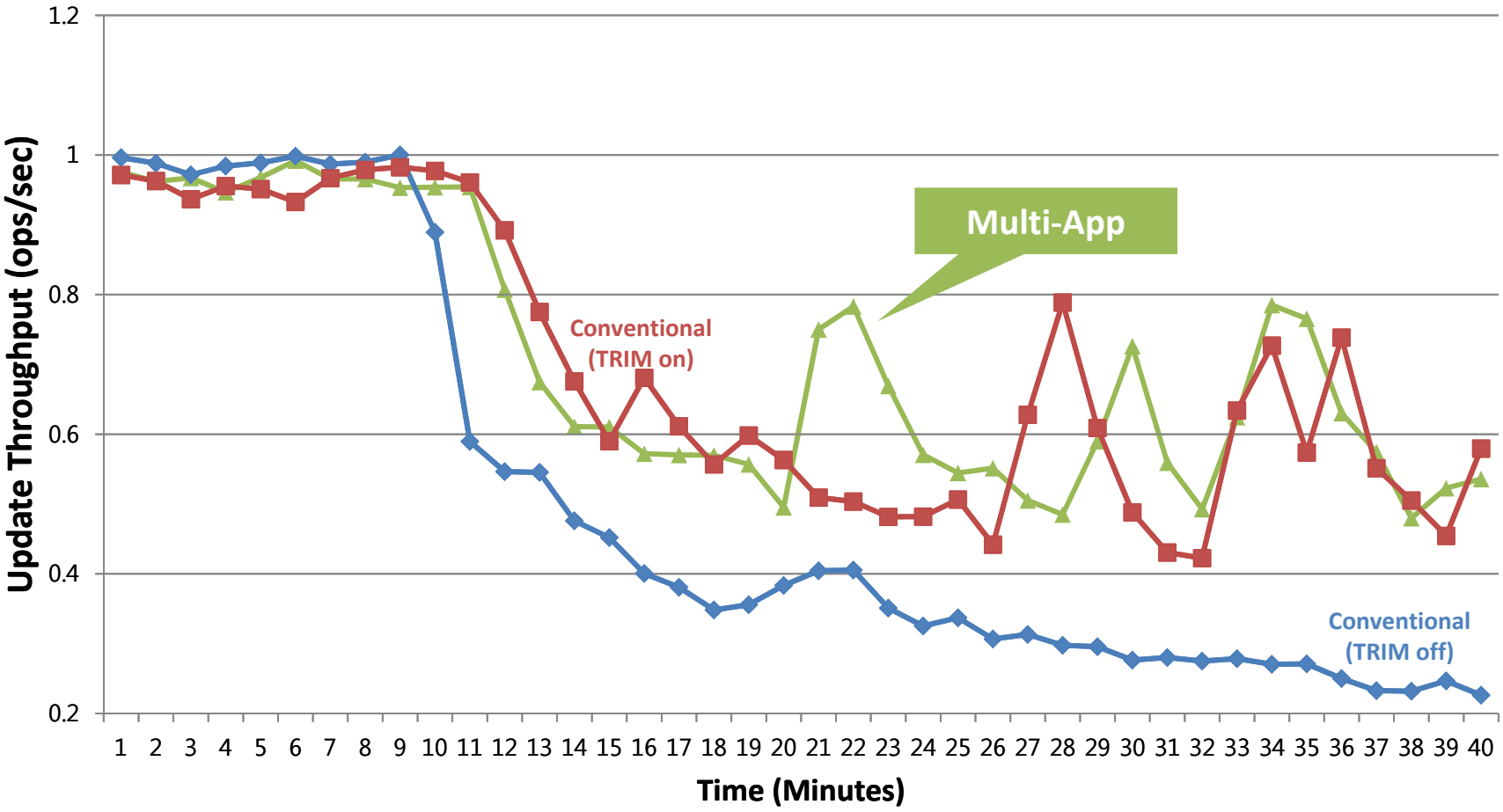
Cassandra's normalized update throughput

- Conventional "TRIM on"



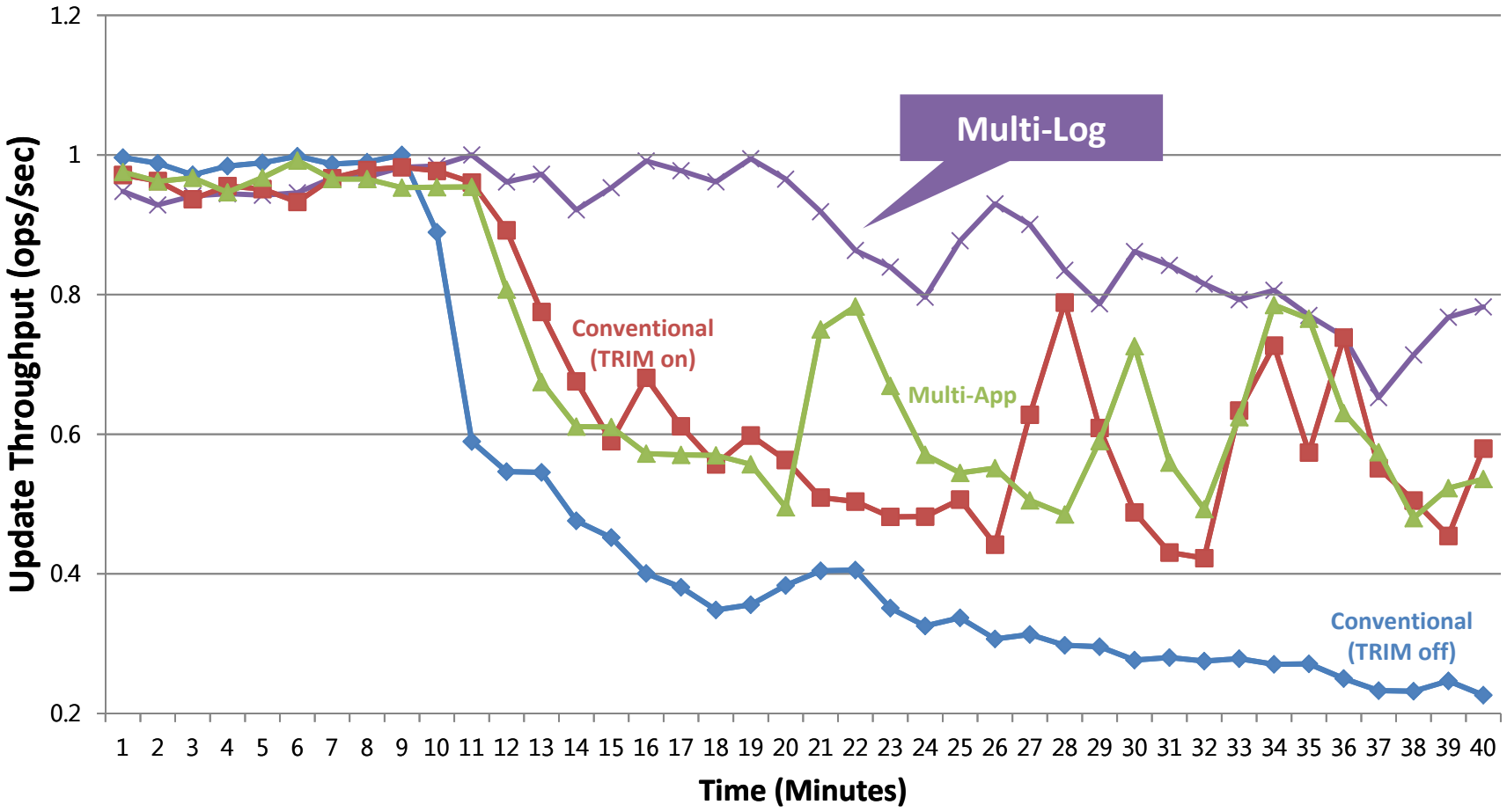
Results

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



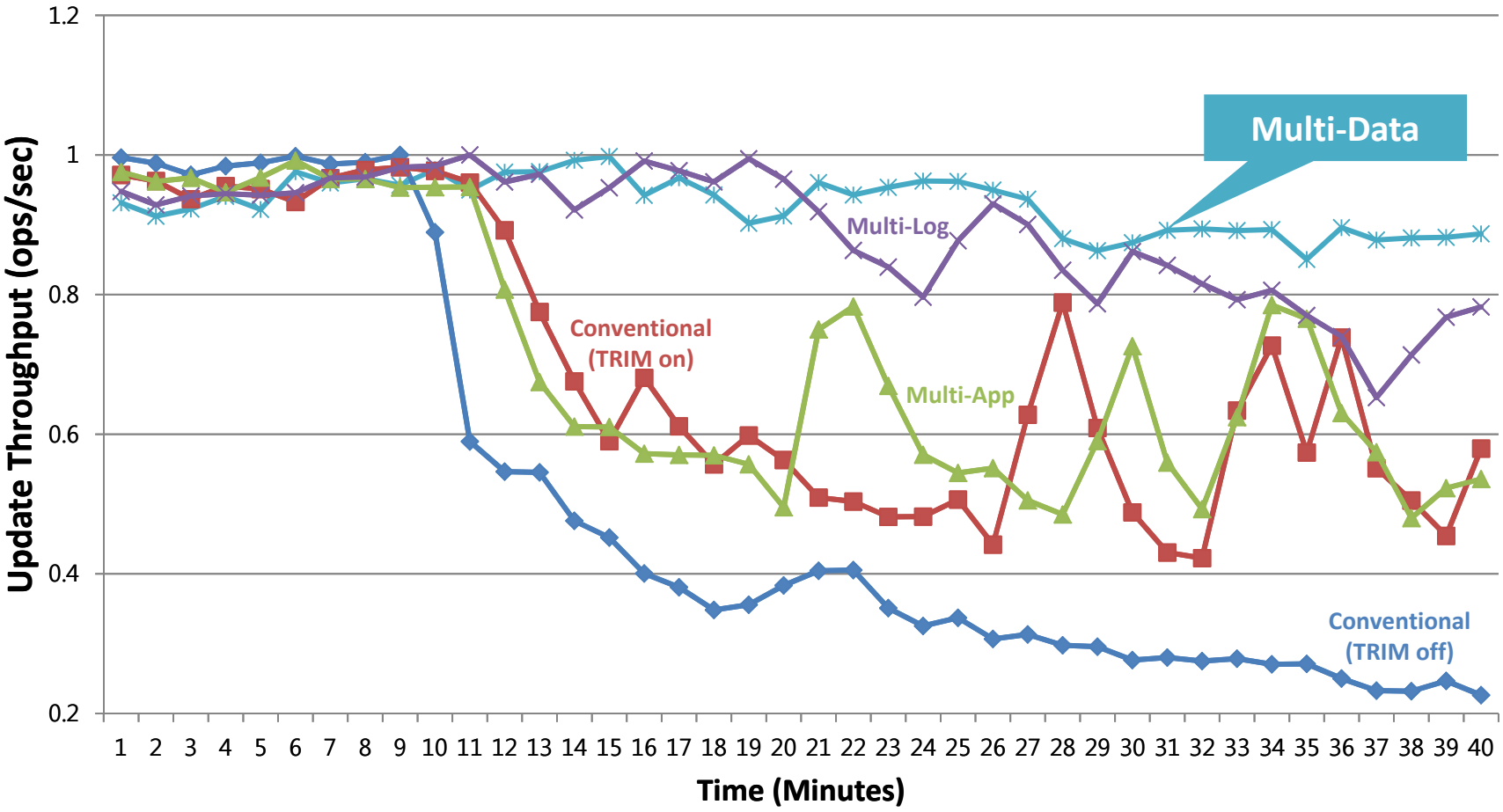
Results

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



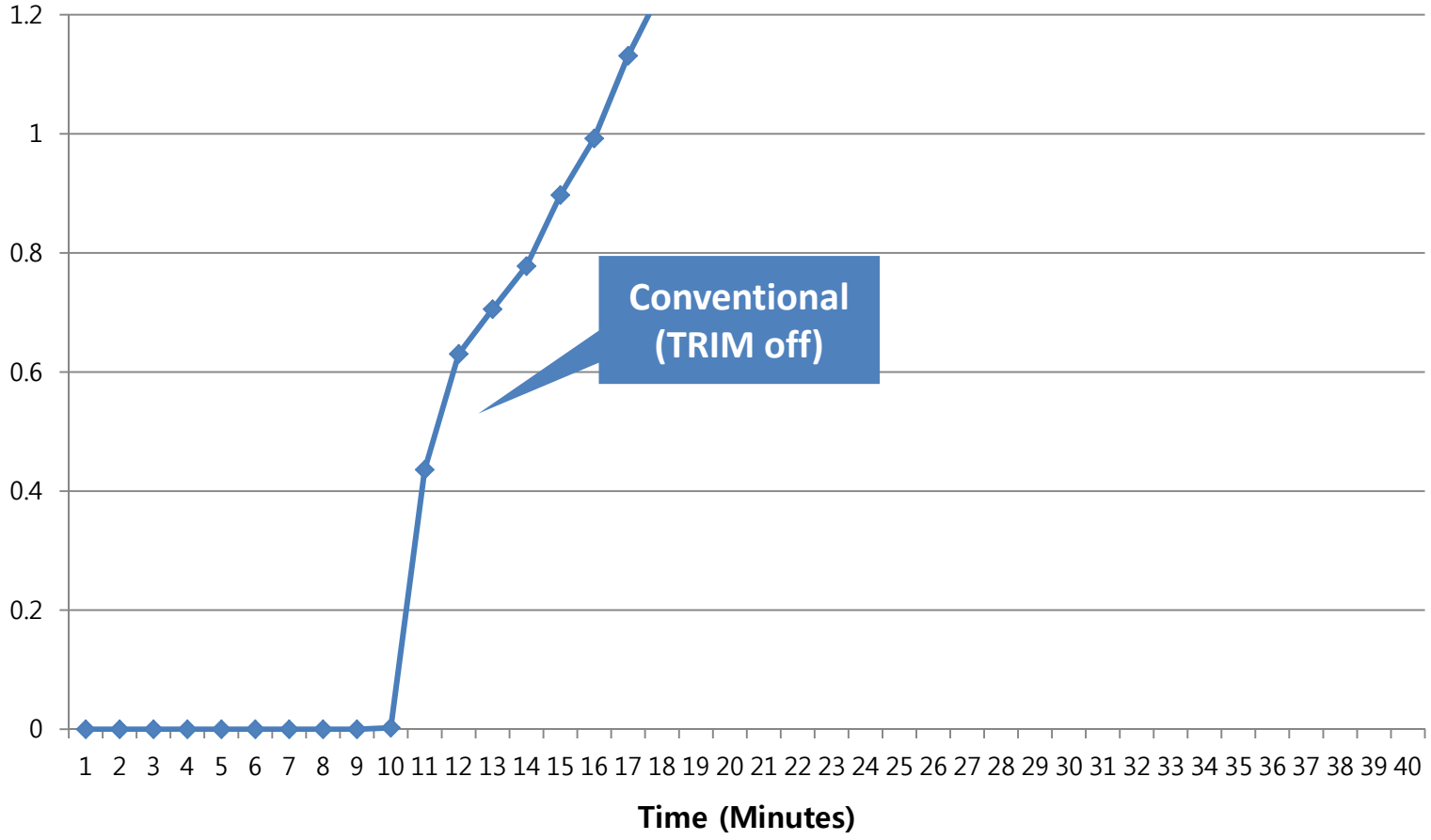
Cassandra's normalized update throughput

- "Multi-Data" (System data vs. Commit-Log vs. Flushed data vs. Compaction data)



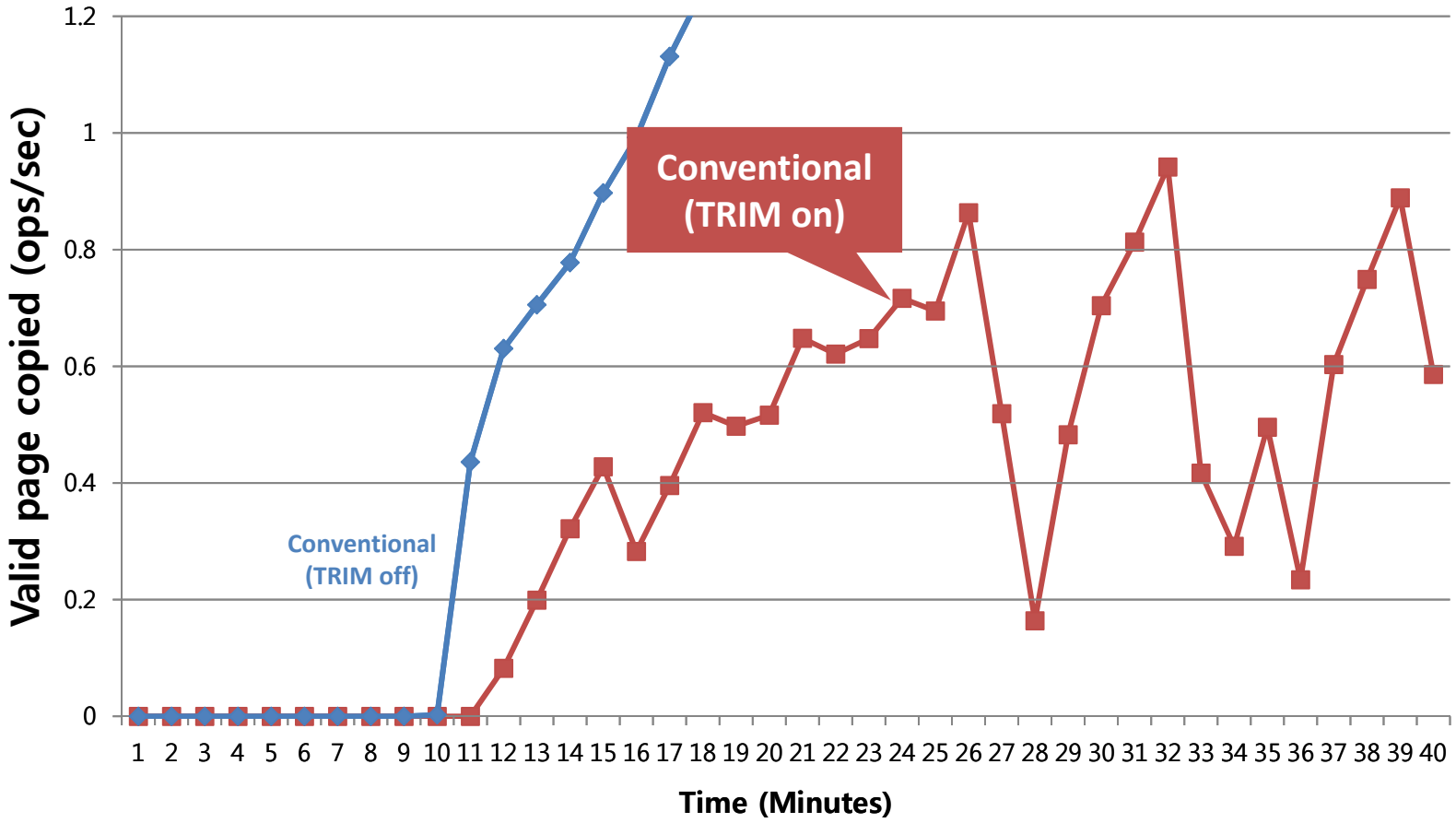
Result #2

- **Cassandra's GC overheads**
  - **Conventional "TRIM off"**



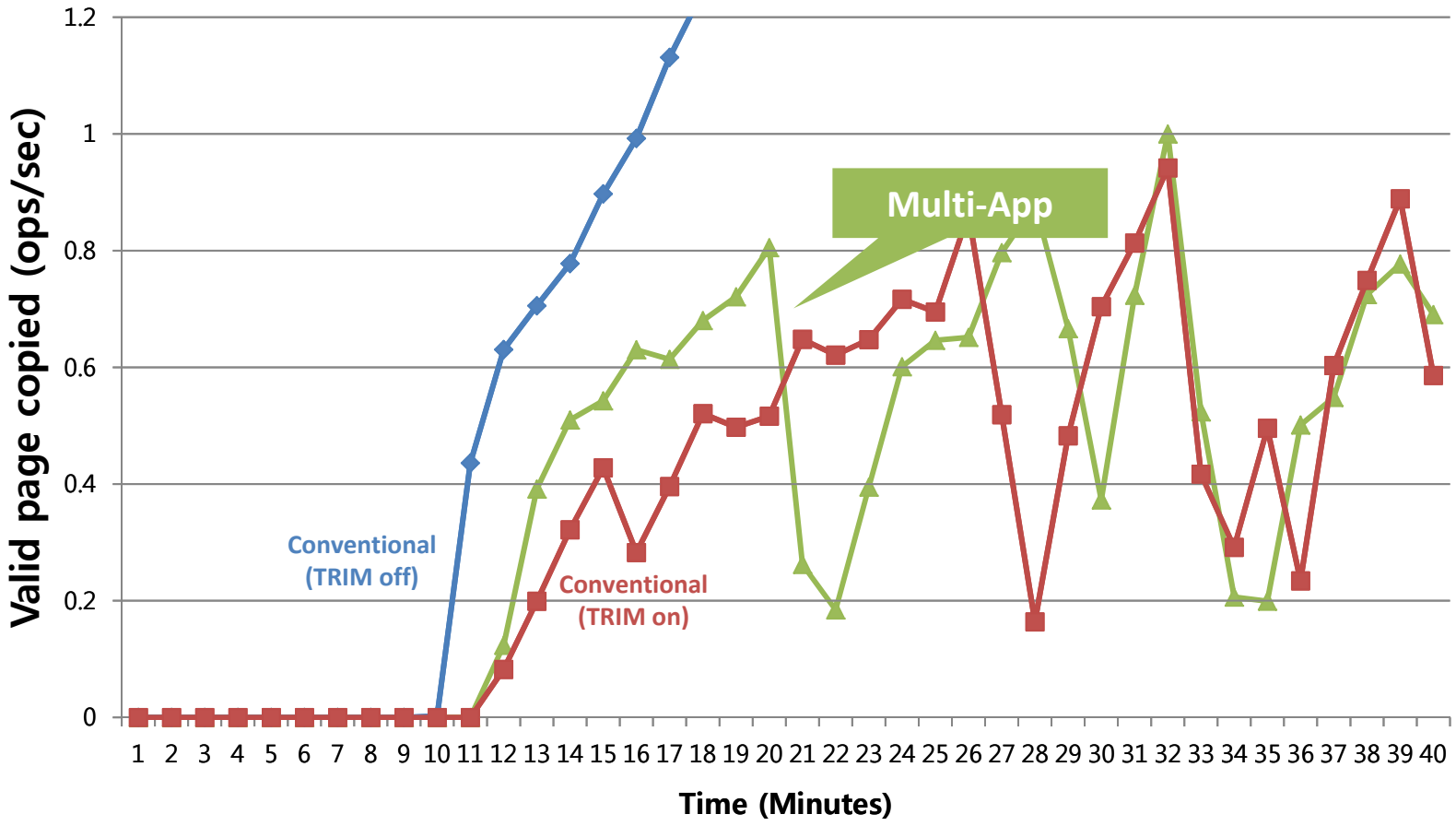
Cassandra's GC overheads

- Conventional "TRIM on"



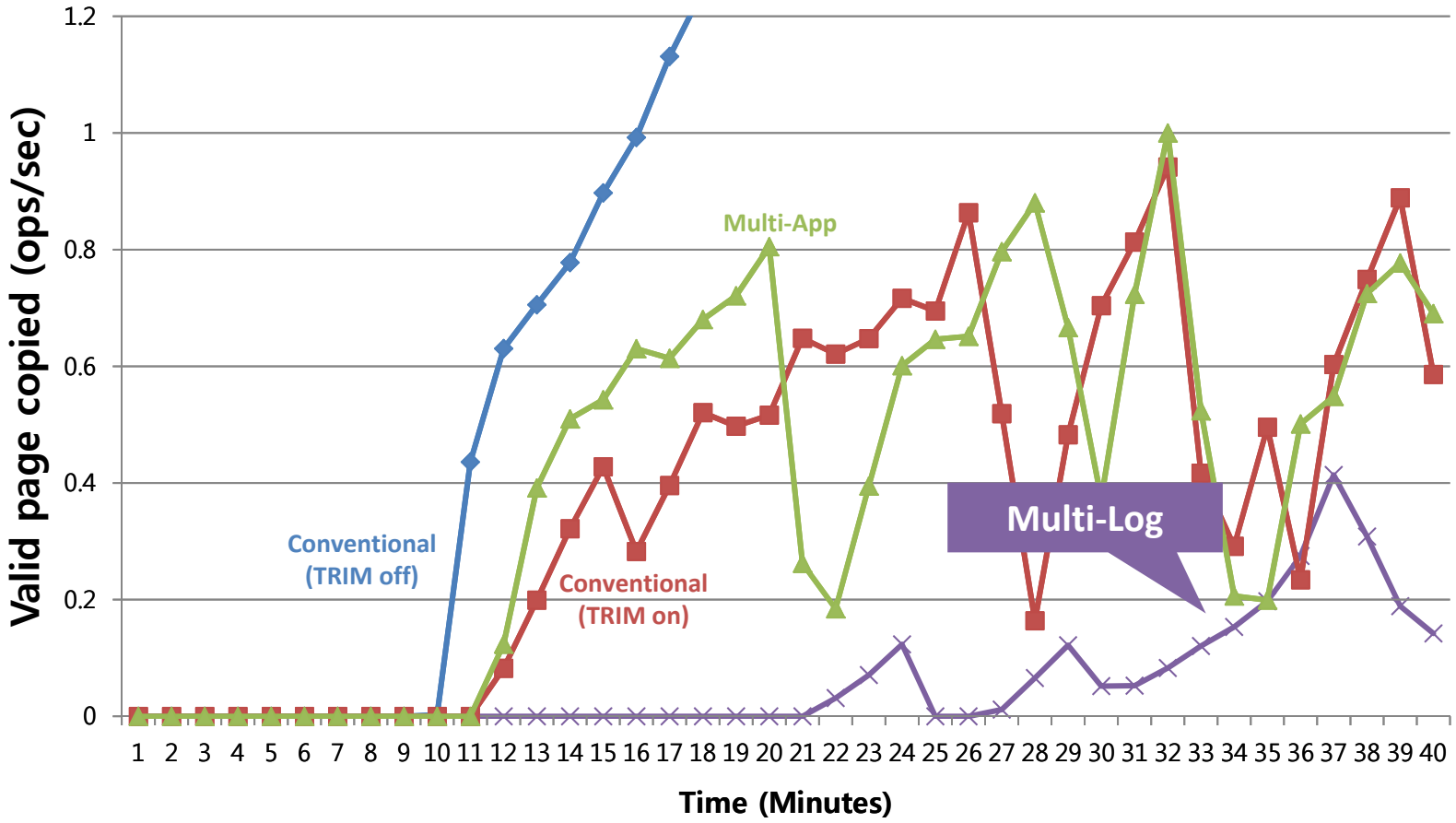
Cassandra's GC overheads

- "Multi-App" (System data vs. Cassandra data)



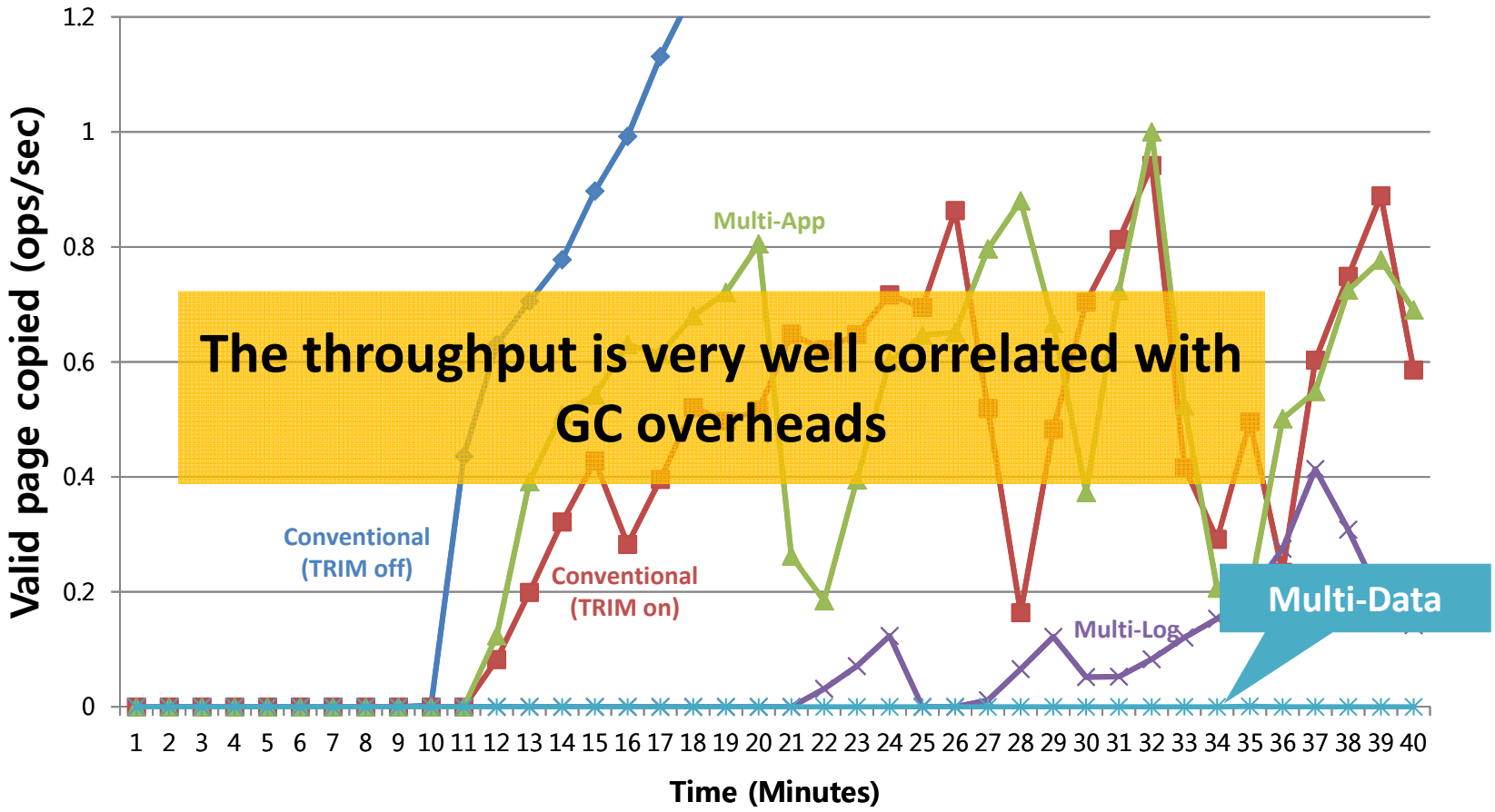
Cassandra's GC overheads

- "Multi-Log" (System data vs. Commit-Log vs. Flushed data)



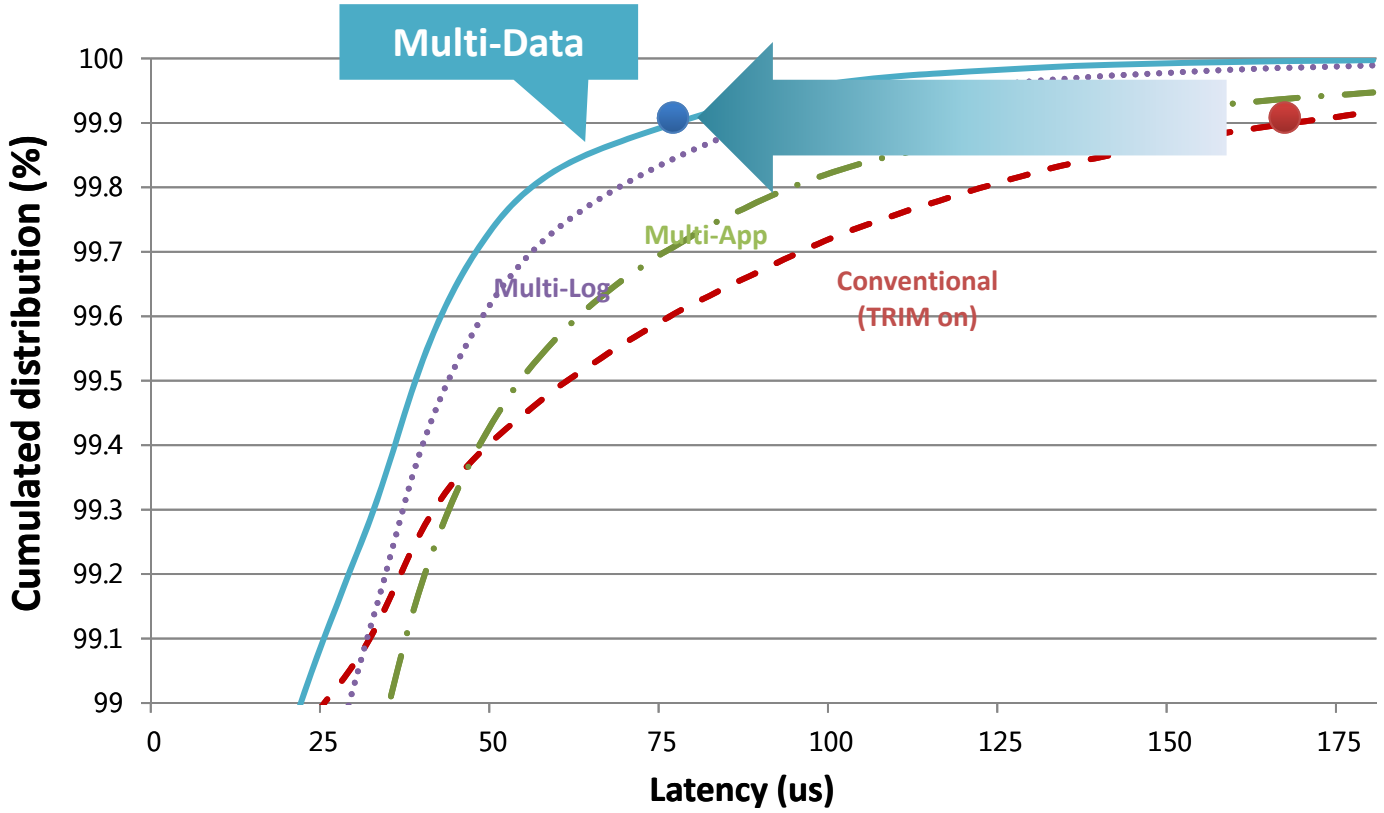
Cassandra's GC overheads

- "Multi-Data" (System data vs. Commit-Log vs. Flushed data vs. Compaction data)



Result #3

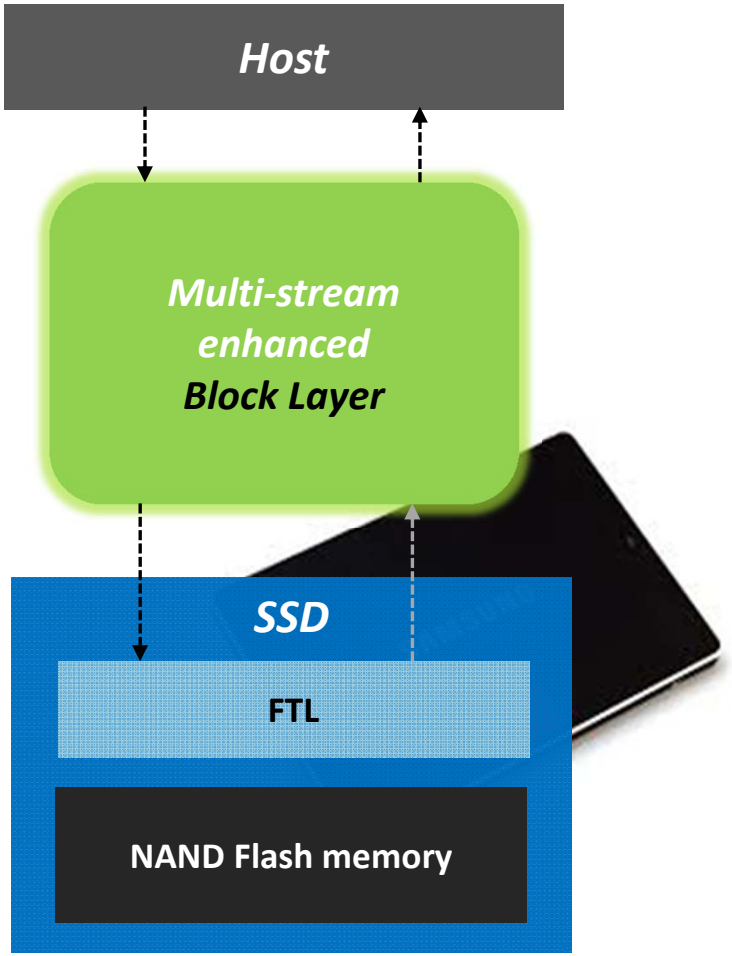
- **Cassandra's cumulated latency distribution**
  - Multi-streaming improves write latency
  - At 99.9%, Multi-Data lowers the latency by 54 % compared to Normal



Conclusion

Multi-streamed SSD

- 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
- Multi-stream interface can be standard for using SSD more efficiently




Be at the Center
of a Revolution
Less Energy. More Speed.
Thank you.
SAMSUNG
PCIe SSD
SAMSUNG