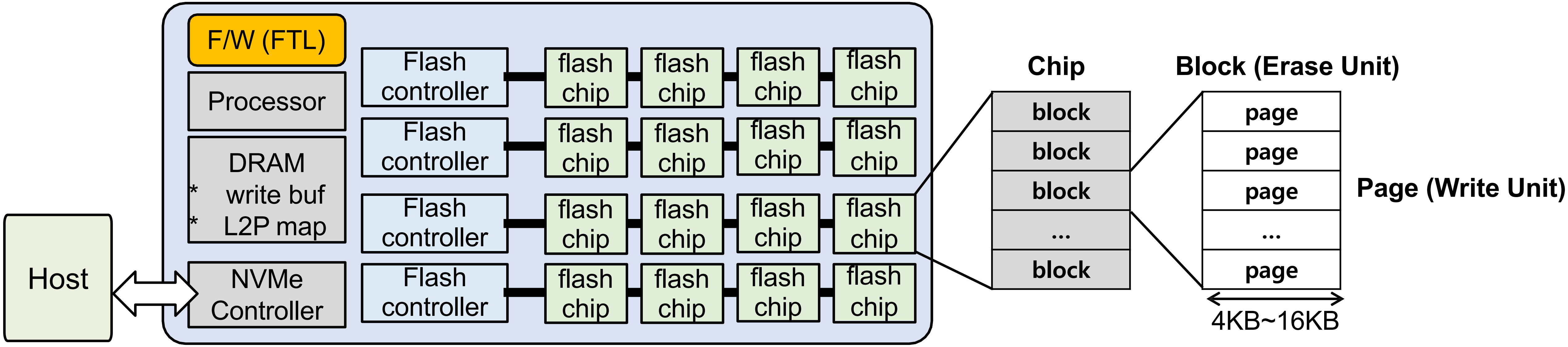
ZNS: Avoiding the Block Interface Tax for Flash-based SSDs

Bjørling Matias, Aghayev Abutalib, Holmberg Hans, Ramesh Aravind, Le Moal Damien, Ganger Greg R, Amvrosiadis George

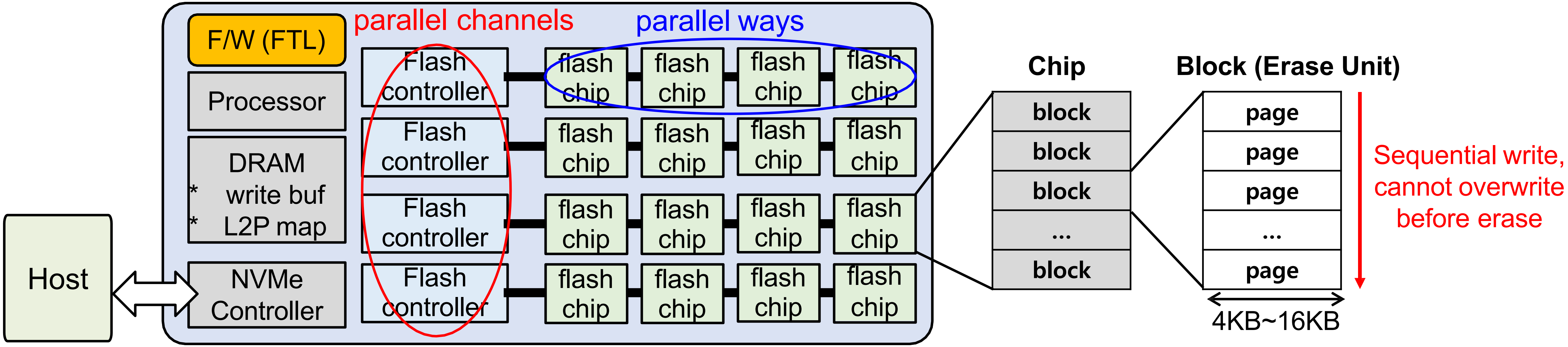
USENIX ATC 21

SSD Architecture



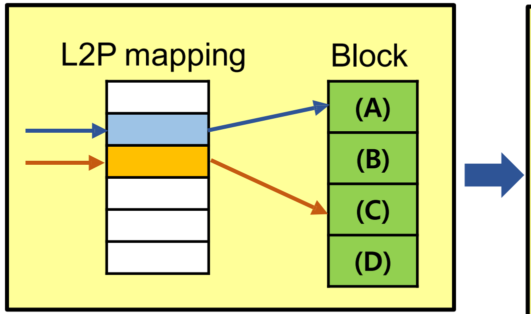
- Existing SSD presents it to the host as a **block device** through the Flash Translation Layer (FTL)
- Provide **block interface**: one-dimensional arrays of fixed-size logical data blocks

SSD Architecture

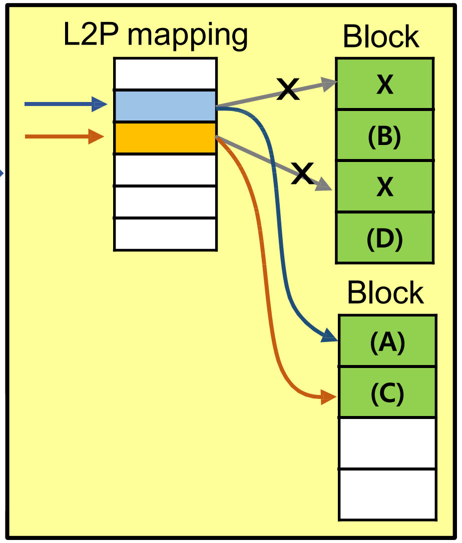


16 (4 channels x 4 ways) flash chips can be accessed in parallel

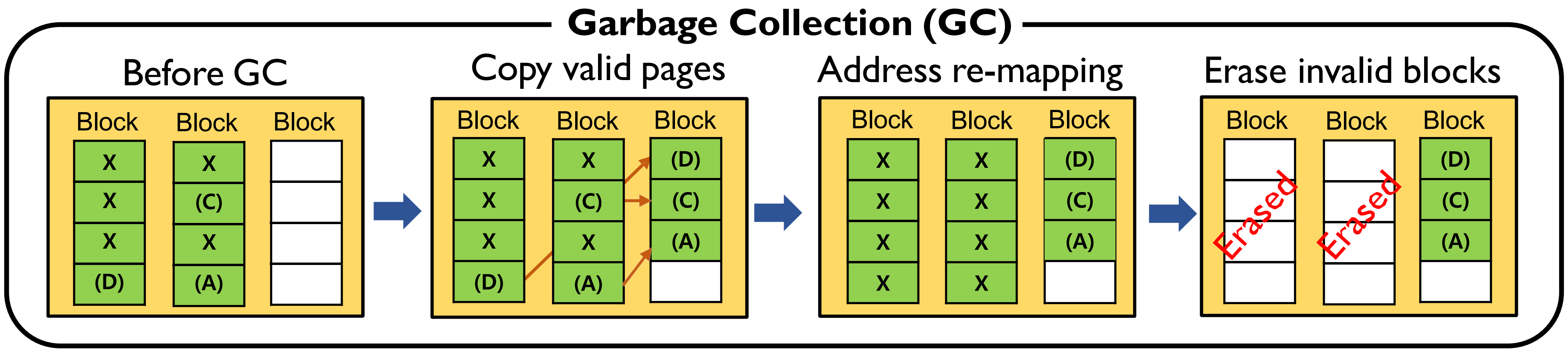
- FTL (L2P map) support:
  - Out-of-place update
  - Address re-mapping



Initial logical-to-physical (L2P) address mapping

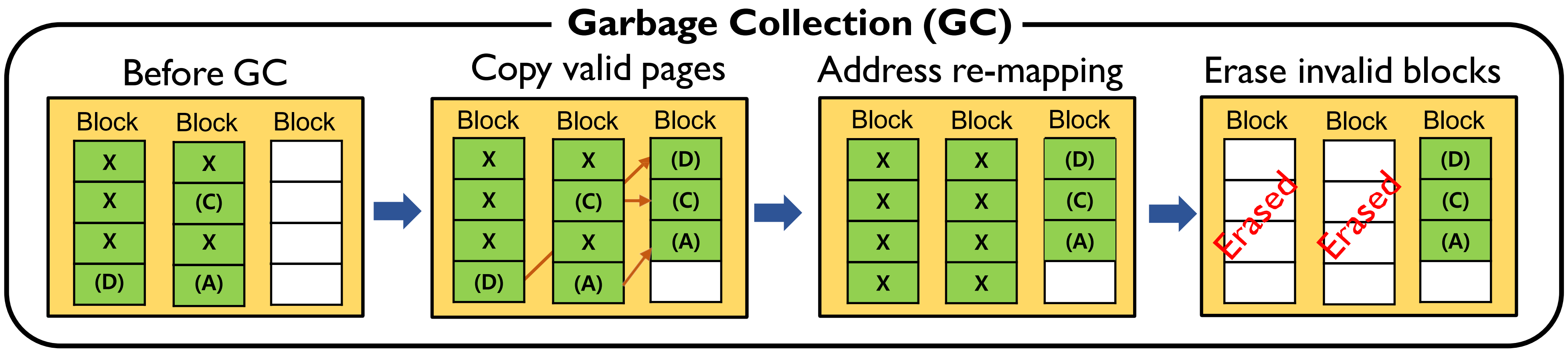


SSD Architecture



- ➢ SSDs "append" pages to erase blocks, need to erase whole block before rewriting
- ➢ Data placement overhead: media over-provisioning (7-28%), higher cost and lower performance

SSD Architecture

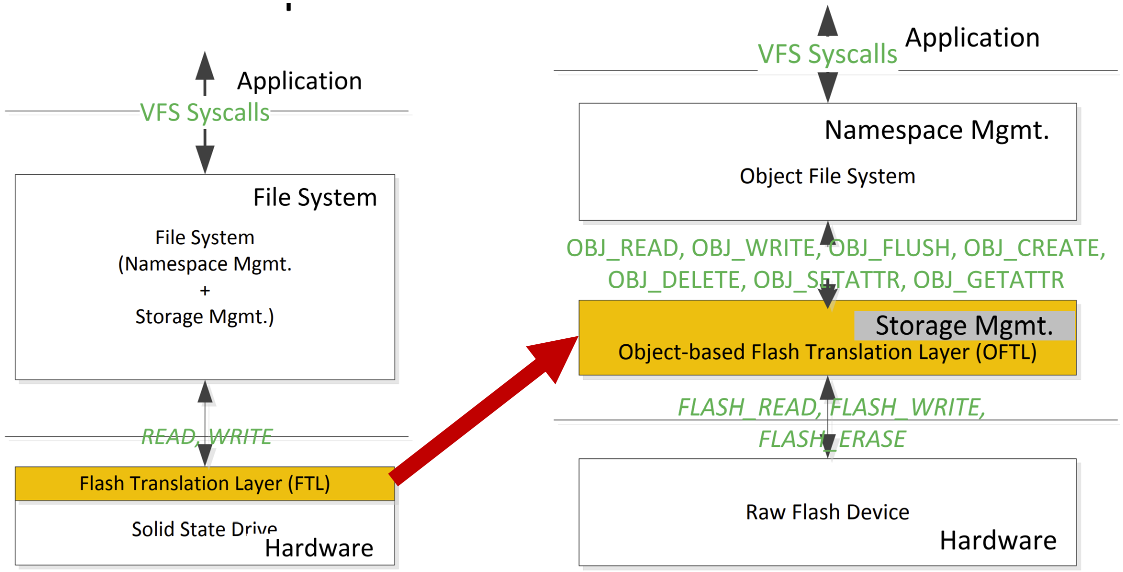


- ➢ SSDs "append" pages to erase blocks, need to erase whole block before rewriting
- ➢ Data placement overhead: media over-provisioning (7-28%), higher cost and lower performance

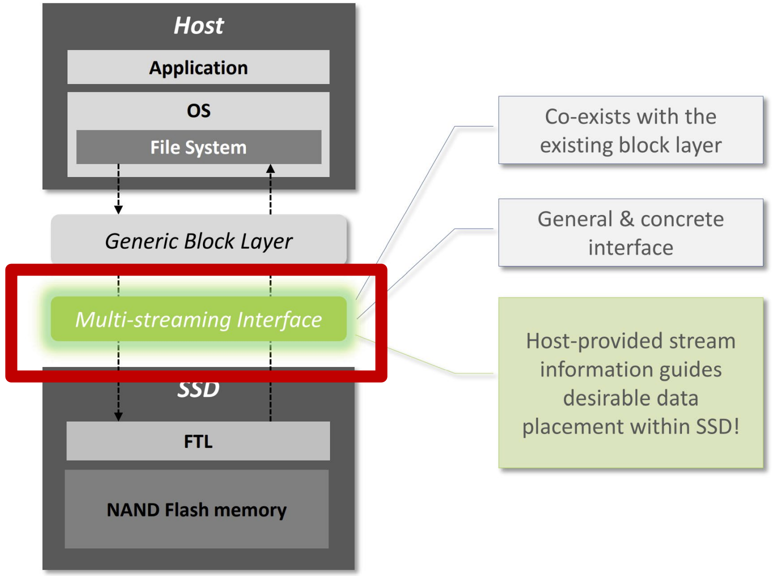
↑ **Problem of Regular SSDs: The Block Interface Overhead**

Existing Overhead-Reduction Strategies

- Open-Channel SSDs
  - Move FTL to host
  - Difficult to adapt to different SSDs

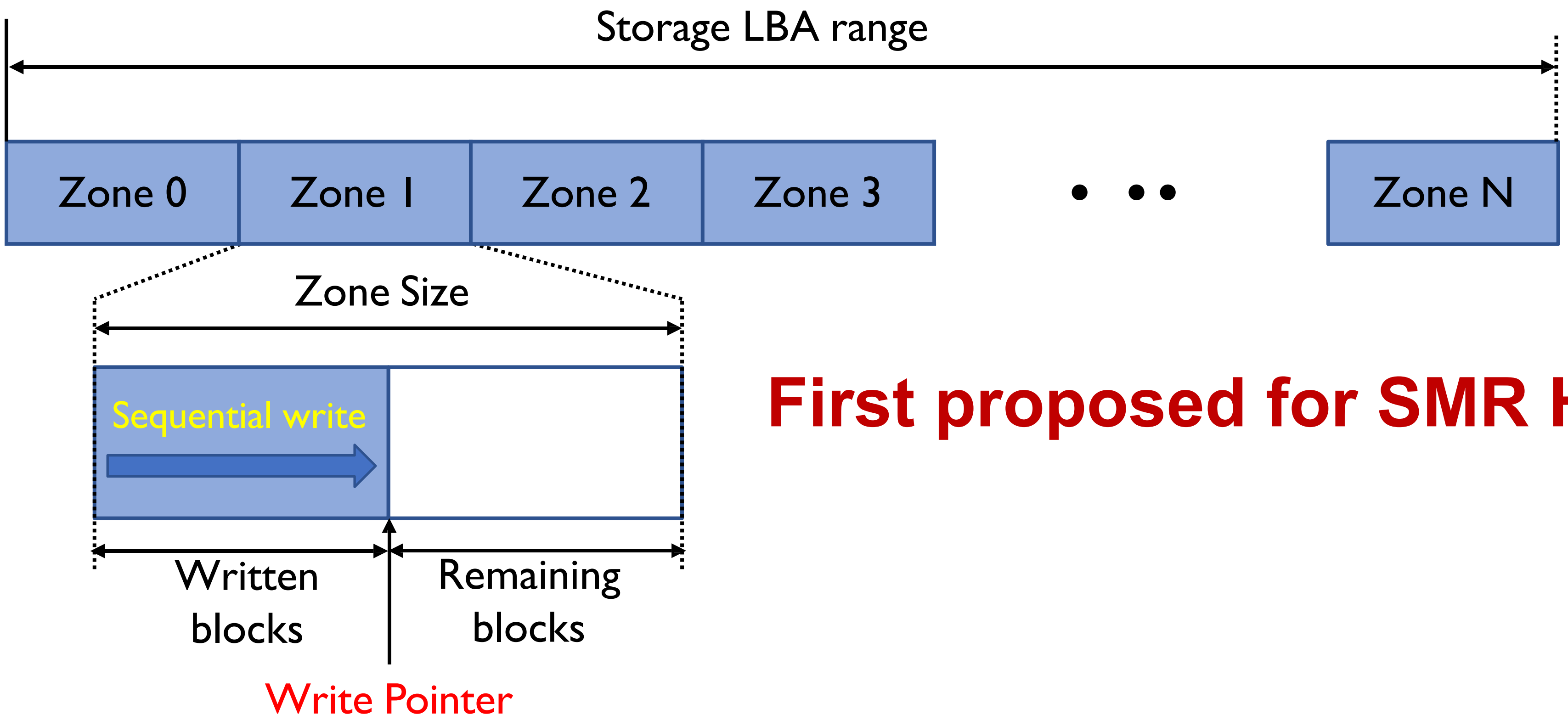


- Stream SSDs
  - Host guide data placement
  - Low GC overhead
  - Unable to saving DRAM and OP overhead



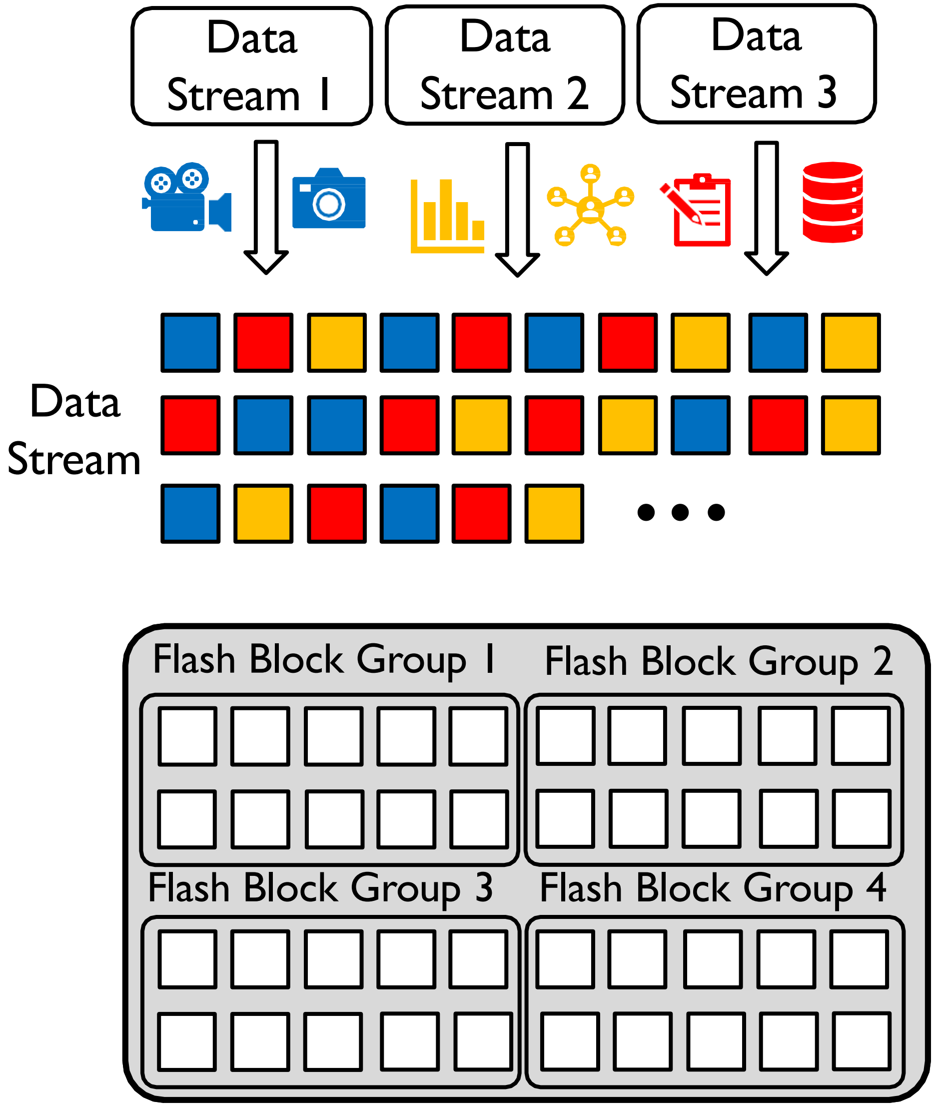
Zoned Name Space (ZNS) Storage

- ➢ The logical address space is divided into fixed-sized zones
- ➢ Each zone must be written sequentially and reset explicitly for reuse

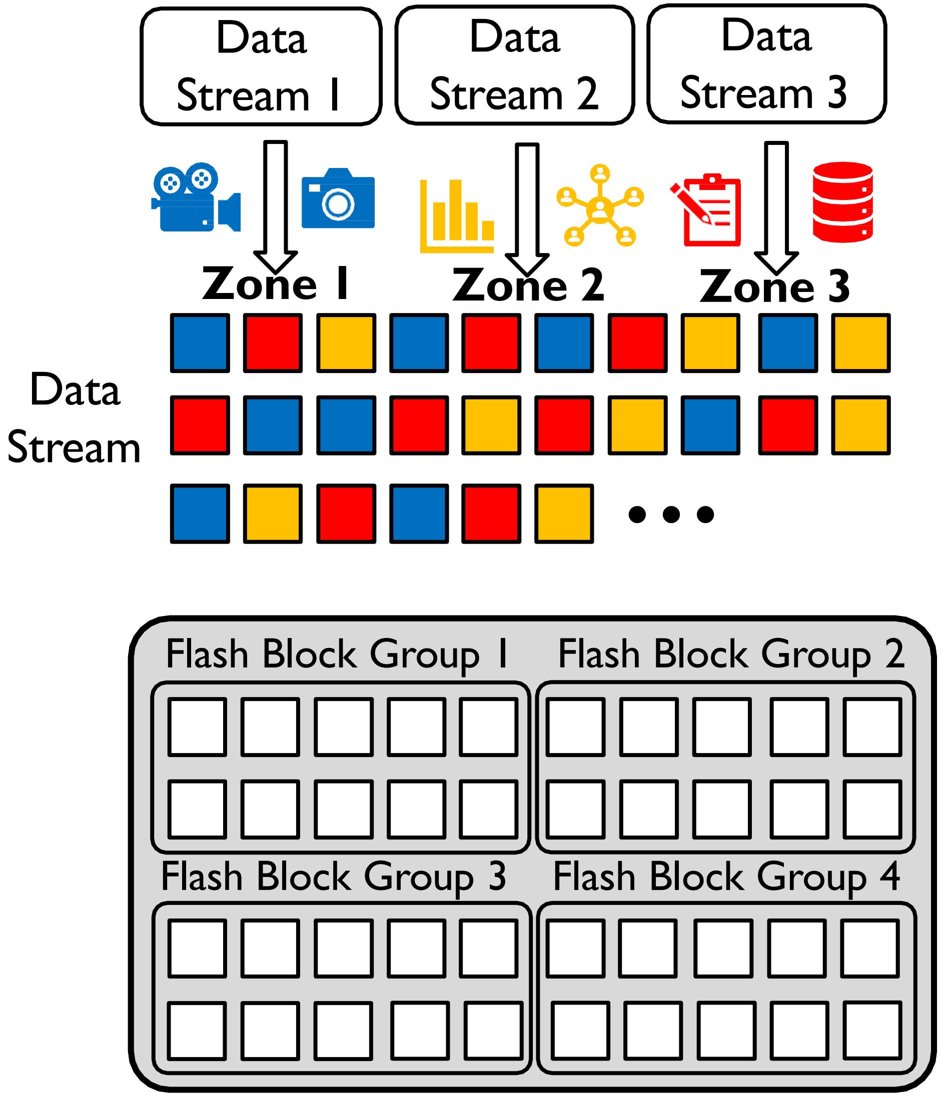


First proposed for SMR HDDs

Regular vs. ZNS SSD

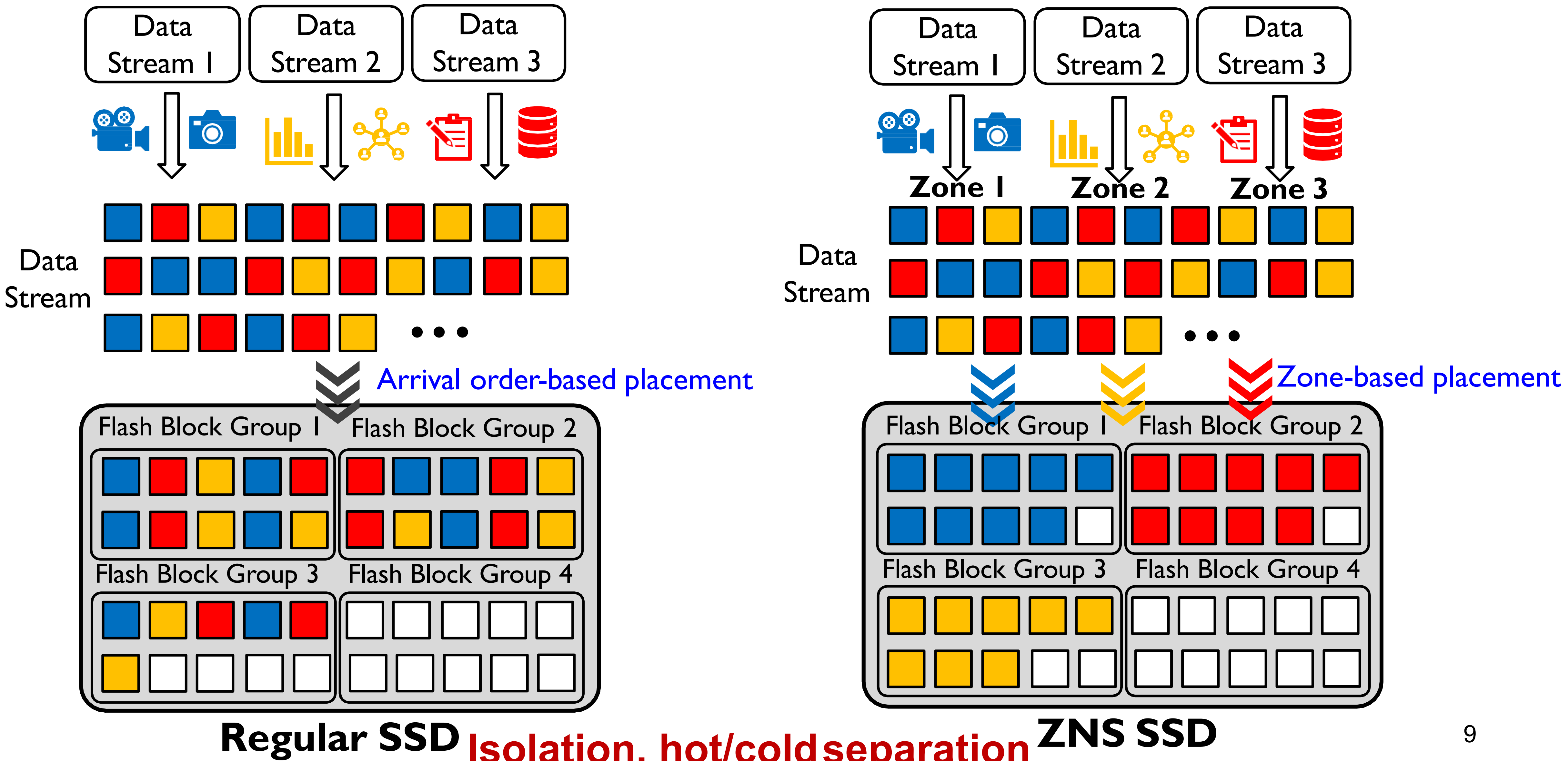


Regular SSD



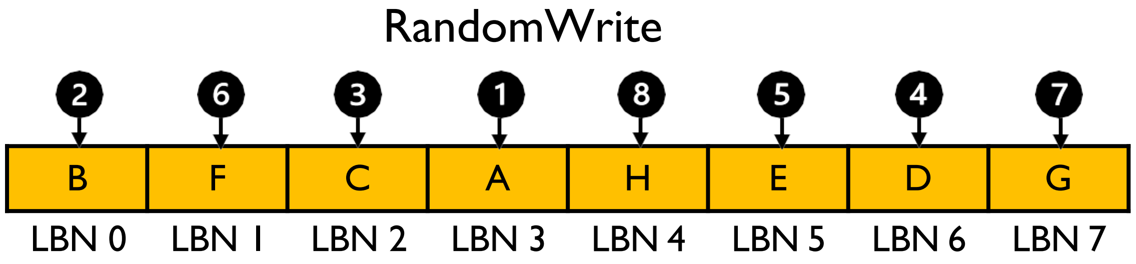
ZNS SSD

Regular vs. ZNS SSD



Isolation, hot/cold separation

Regular vs. ZNS SSD

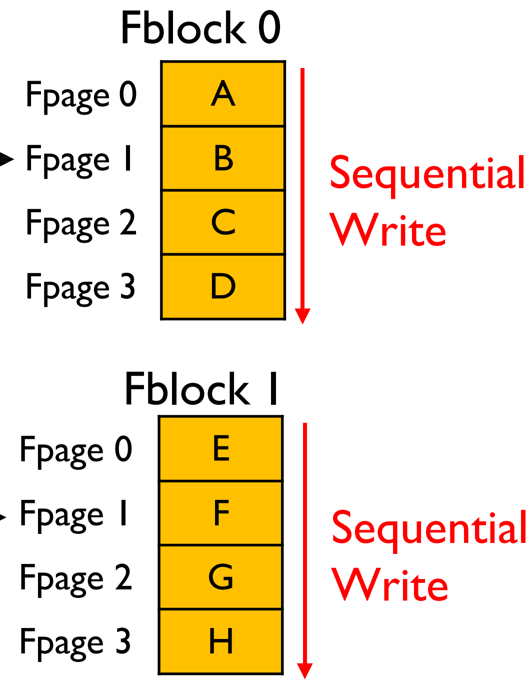


LBN: Logical Block Number

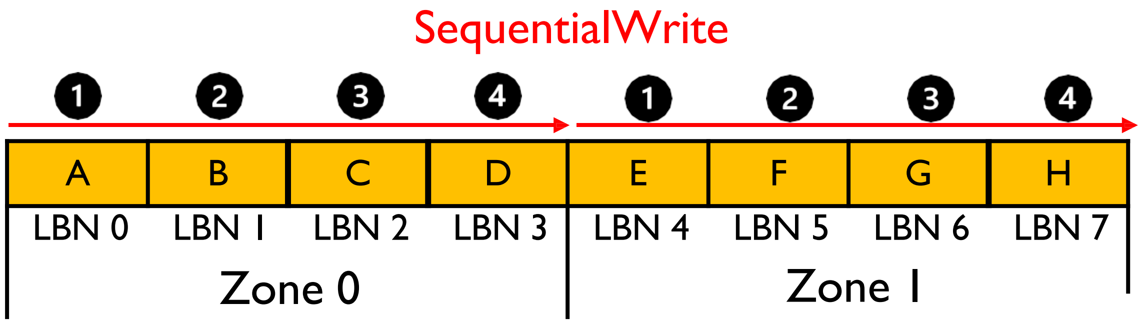
LBN-level L2P translation table

LBN	Fblock/Fpage
0	0/1
1	1/1
2	0/2
3	0/0
4	1/3
5	1/0
6	0/3
7	1/2

Logical Block Size = 4KB



Regular SSD

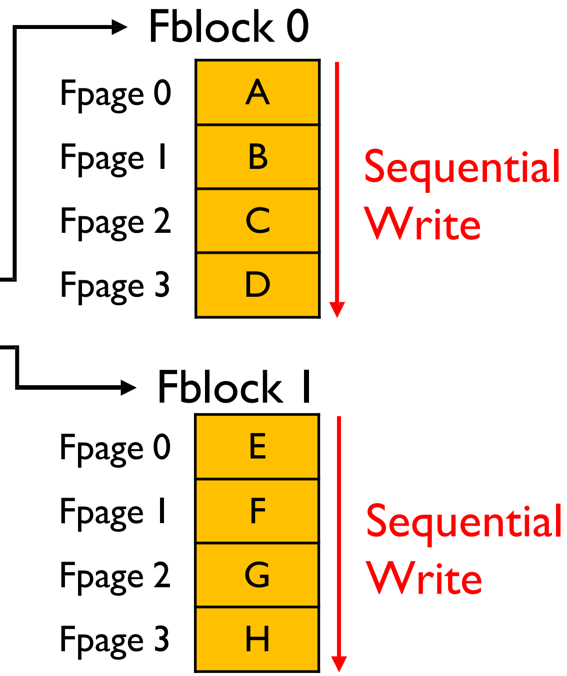


Zone-level L2P translation table

Zone	Fblock
0	0
1	1

Logical Zone Size = xxMB

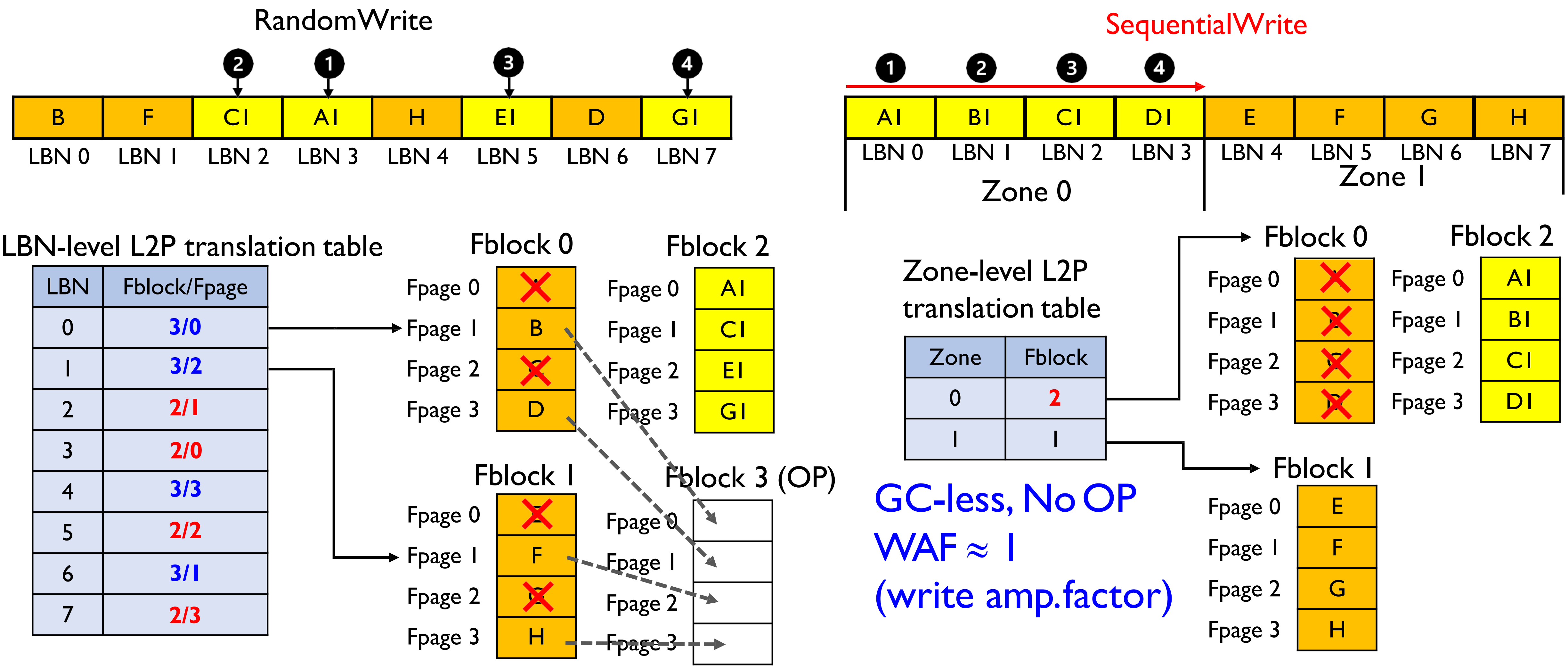
Reduced L2P Size



ZNS SSD

Small L2P Translation Table

Regular vs. ZNS SSD



LBN	Fblock/Fpage
0	3/0
1	3/2
2	2/1
3	2/0
4	3/3
5	2/2
6	3/1
7	2/3

Zone	Fblock
0	2
1	1

GC-less, No OP
WAF ≈ 1
(write amp.factor)

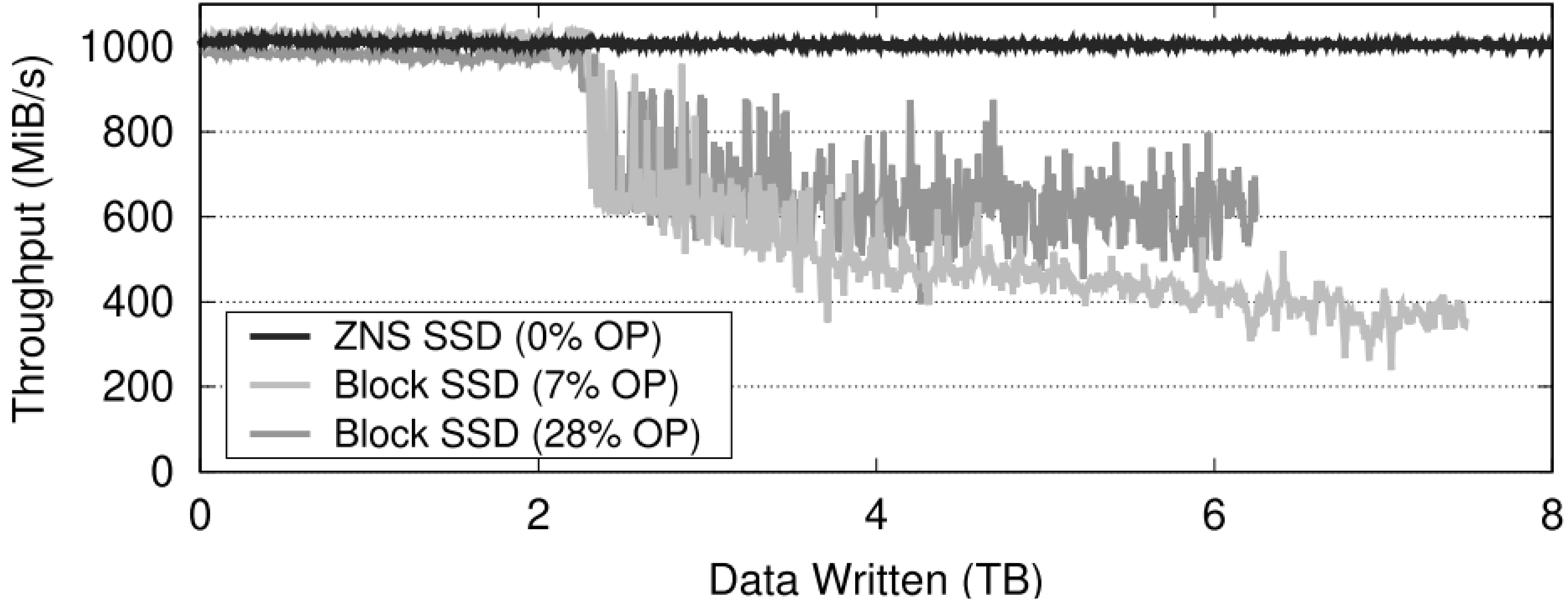
Regular SSD

GC: valid page copy
➔ write amplified, unexpected delay

ZNS SSD

ZNS SSD

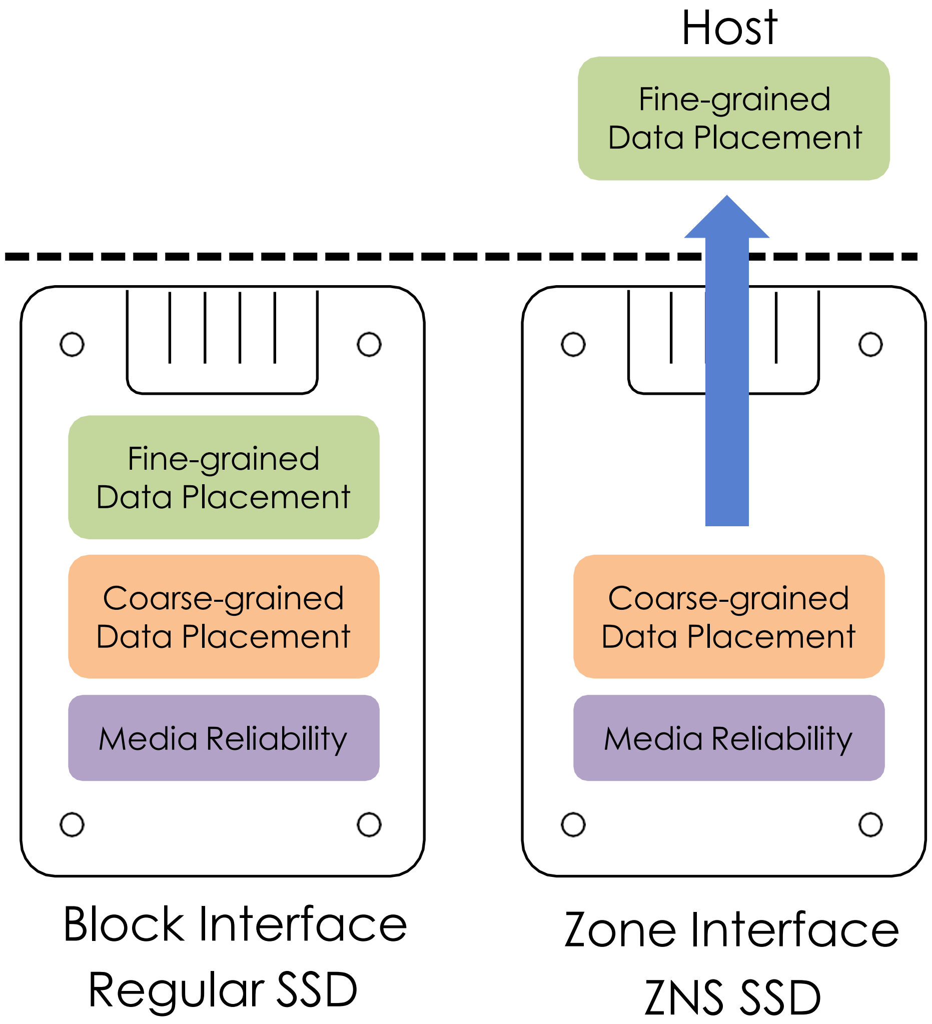
- ➢ No overwrites/out-of-order writes allowed under ZNS.
  - Only works if software layers above are modified to support the limitations



Main Problem: Which applications can evolve to use the ZNS interface? How?

Modeling ZNS SSDs

- Random writes are disallowed by ZNS, and zones must be explicitly reset by the host, the data placement occurs at the coarse-grained level of zones
- Move GC (based on zones) responsibilities from FTL to Host
- ZNS SSD translate sequential zone writes onto distinct erase blocks
- Host manage data arrangement in each zone



Main Idea: shifting responsibilities for managing data placement within erase blocks from FTLs to host software

Three Ways to Adopt ZNS SSDs

- Host FTL
  - Exposes ZNS SSD as block interface SSD
  - Enable workloads that specifically require random write characteristics.
  - High system overhead on DRAM and CPU

  Zoned Block Device (ZBD) subsystem

- In Kernel File Systems
  - Place data onto zones using file system characteristics
  - Efficient use of resources, as file system simply places data more efficiently
  - Layer of indirection away from the application, and therefore some inefficient data placement causes host GC.

  Modify F2FS

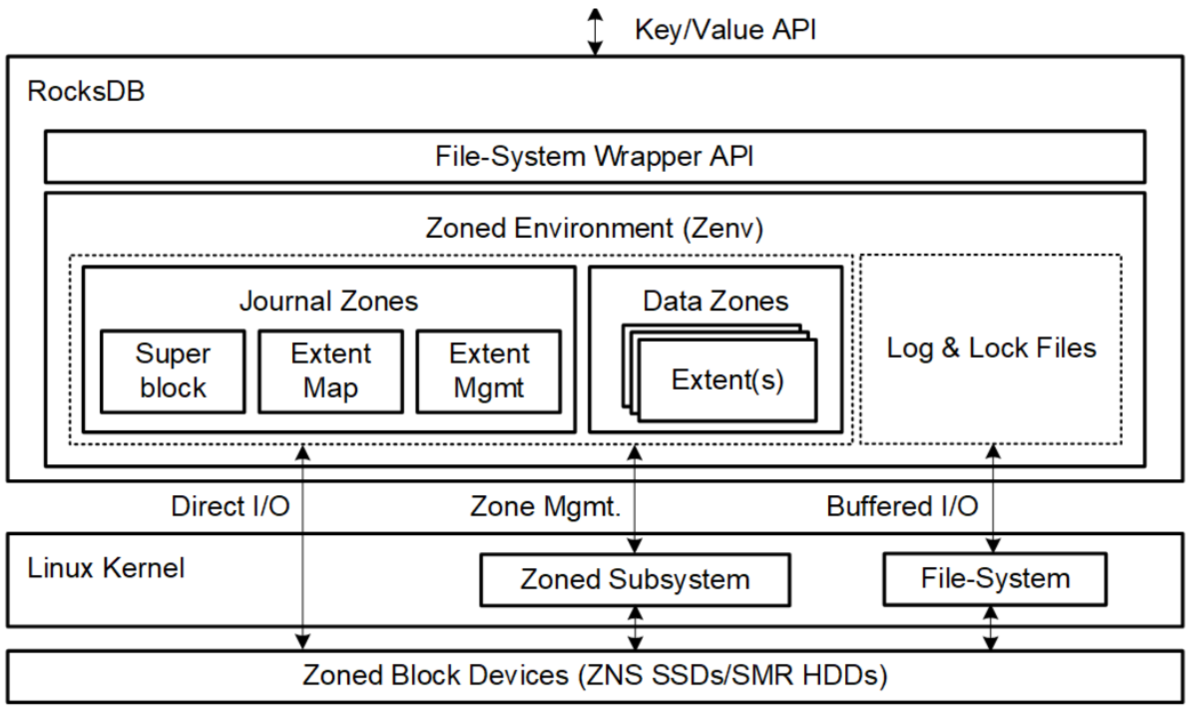
- End-to-End Application
  - Places data onto zones using application characteristics
  - No indirection overhead cause by FTL data placement nor file system.
  - Highest performance and the lowest write amplification

  RocksDB/**ZenFS**

Low — Performance — High

RocksDB use ZenFS as Backend

- Write lifetime hints from RocksDB simplify Garbage Collection
- Journal Zones for:
  - The superblock: initial entry point for ZenFS
  - The write-ahead-log (WAL) and data file to zone translation mapping through extents.
- Data Zones for extents: variable-sized, block-aligned, contiguous region that is written sequentially to a data zone (SSTables)



Main Idea: Arrange data into different zones based on the RocksDB hint

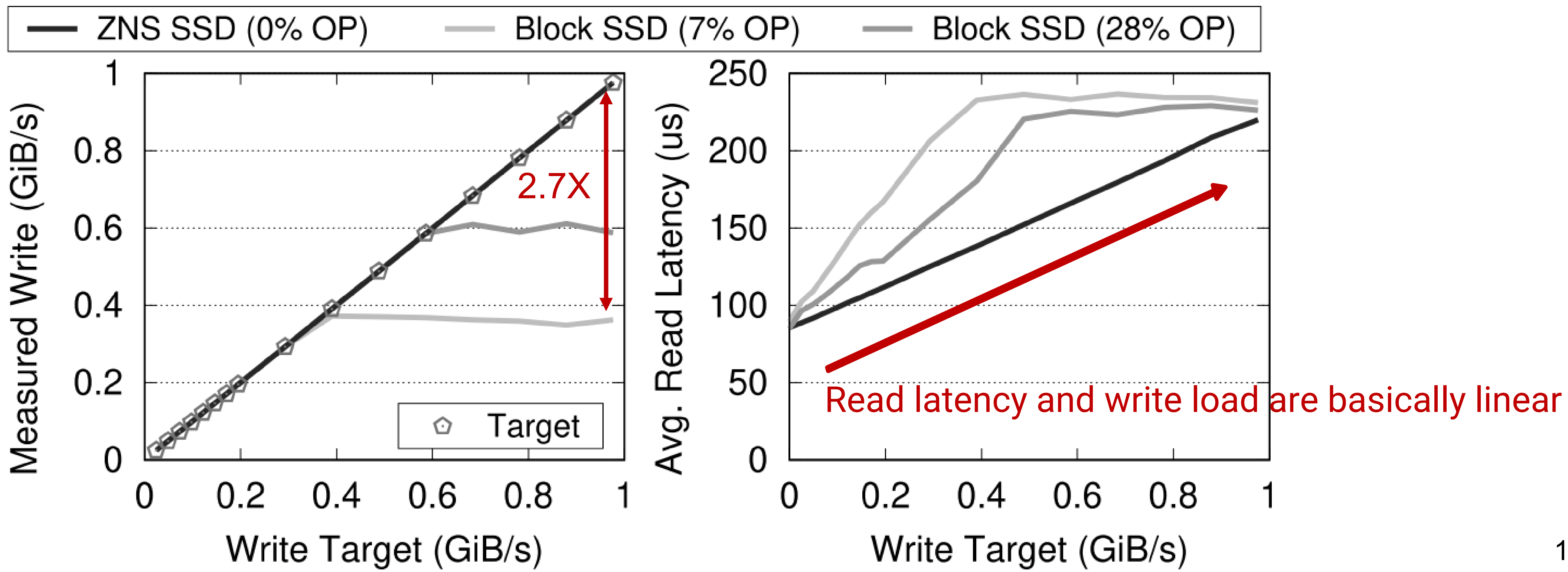
Evaluation

- Platforms
  - Production hardware platform that can expose itself as either a block-interface SSD or a ZNS SSD.
- Methodology
  - Raw I/O performance
  - RocksDB Performance
    - XFS, F2FS (Block)
    - F2FS /w zone support (ZNS)
    - RocksDB /w ZenFS (ZNS)

SSD Interface	Conv.	Conv.	ZNS
Media Capacity	2 TiB	2 TiB	2 TiB
Host Capacity	1.92 TB	1.60 TB	2 TB
Over-provisioning	7%	28%	0%
Placement Type	None	None	Zones
Max Active Zones	N/A	N/A	14
Zone Size	N/A	N/A	2048 MiB
Zone Capacity	N/A	N/A	1077 MiB

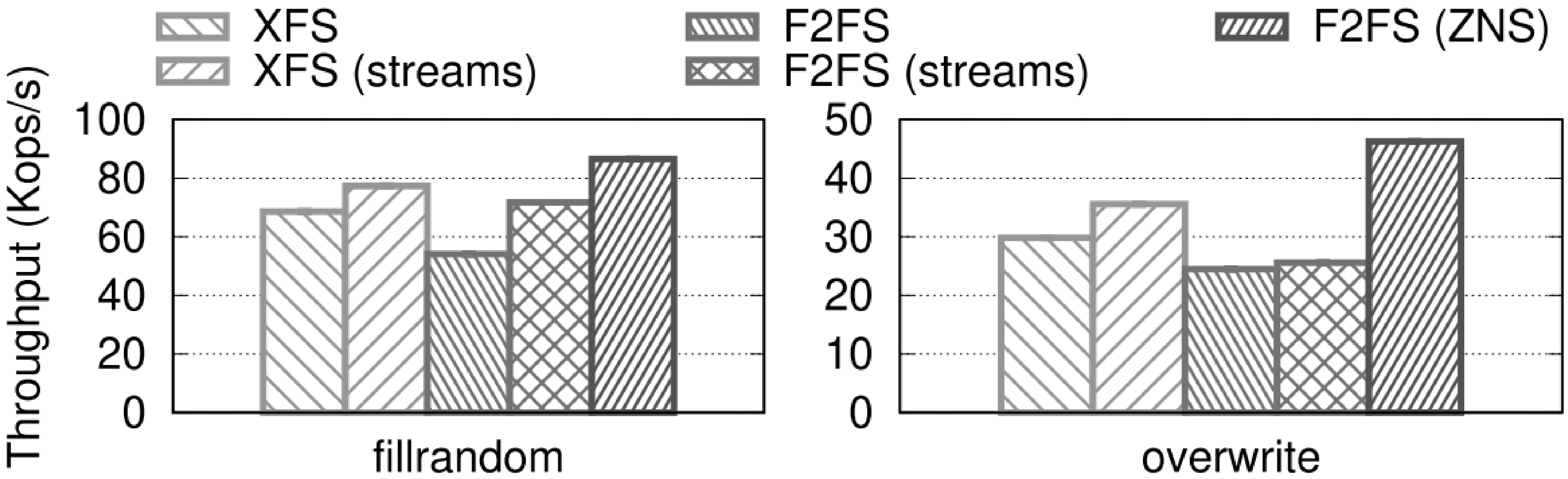
Raw I/O Write Throughput & Read Latency

- ➢ Measure the drive's ability to reach specific host write throughput target ranging from 0 to 1GiB/s
  - ZNS SSD Issues 64KiB write I/Os to 4 active zones



RocksDB Writes

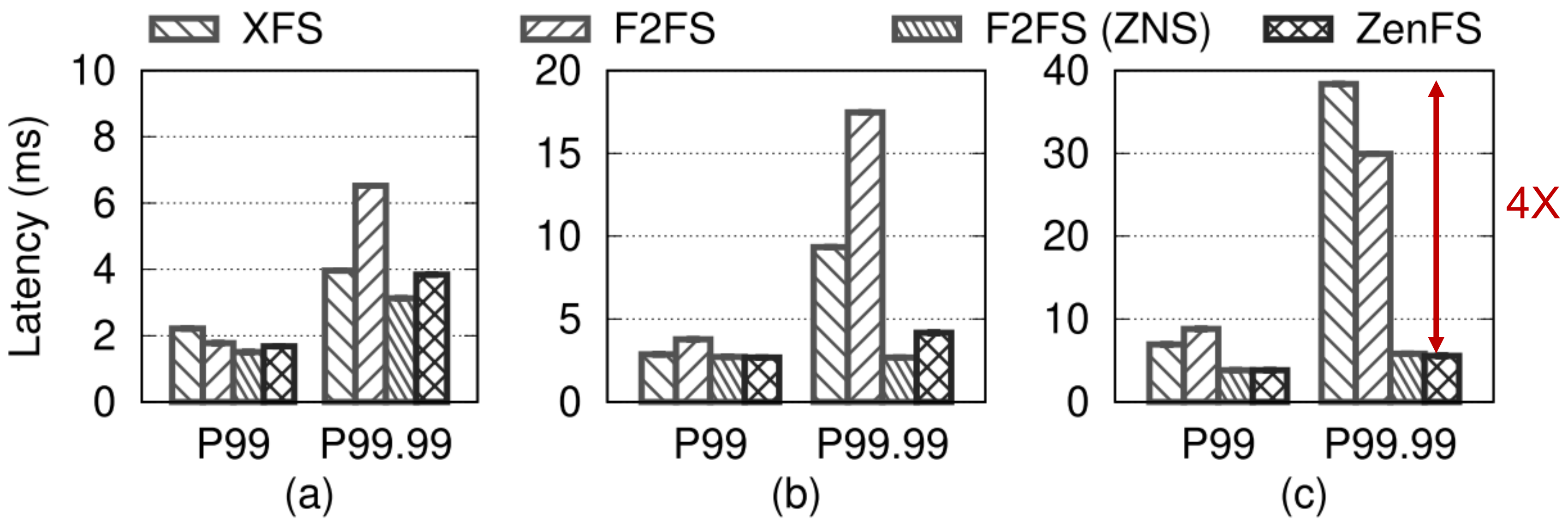
- Throughput of RocksDB during the fill-random and overwrite benchmarks when executing on an block interface SSD with 7% OP and stream support.



(Repeatedly writes device capacity) (Overwrites random key-value pairs)

RocksDB Reads

- Latency of RocksDB reads during:
  - (a) Random-read benchmark
  - (b) The read-while-writing benchmark with writes rate-limited to 20 MiB/s
  - (c) Read-while-writing benchmark with no write limits using the block-interface SSD with 28% OP and the ZNS SSD.



Conclusion

- ZNS SSD is higher performance and lower cost.
- By shifting responsibilities for managing data placement within erase blocks from FTLs to host software, ZNS eliminates the need for fine-grained indirection table, garbage collection, and media over-provisioning.
- 99.9th-percentile random-read latency for RocksDB/ZenFS is at least 2-4x lower on a ZNS SSD compared to a block-interface SSD, and the write throughput is 2x higher.