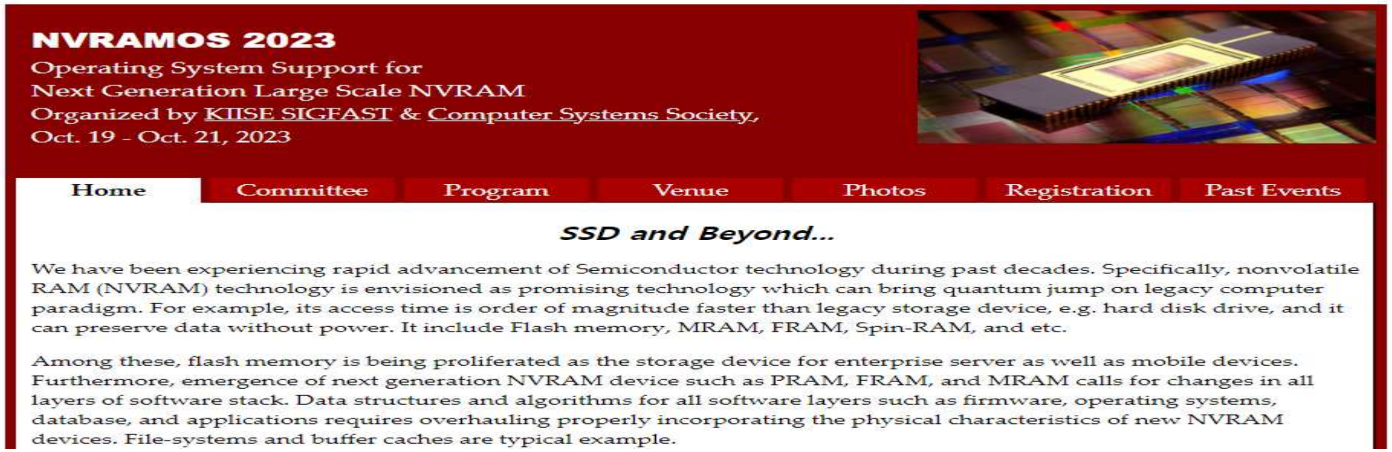
ZNS SSDs (Zoned NameSpace SSDs): Characteristics and Implications


NVRAMOS 2023
Operating System Support for
Next Generation Large Scale NVRAM
Organized by KIISE SIGFAST & Computer Systems Society,
Oct. 19 - Oct. 21, 2023

Home | Committee | Program | Venue | Photos | Registration | Past Events

SSD and Beyond...

We have been experiencing rapid advancement of Semiconductor technology during past decades. Specifically, nonvolatile RAM (NVRAM) technology is envisioned as promising technology which can bring quantum jump on legacy computer paradigm. For example, its access time is order of magnitude faster than legacy storage device, e.g. hard disk drive, and it can preserve data without power. It include Flash memory, MRAM, FRAM, Spin-RAM, and etc.

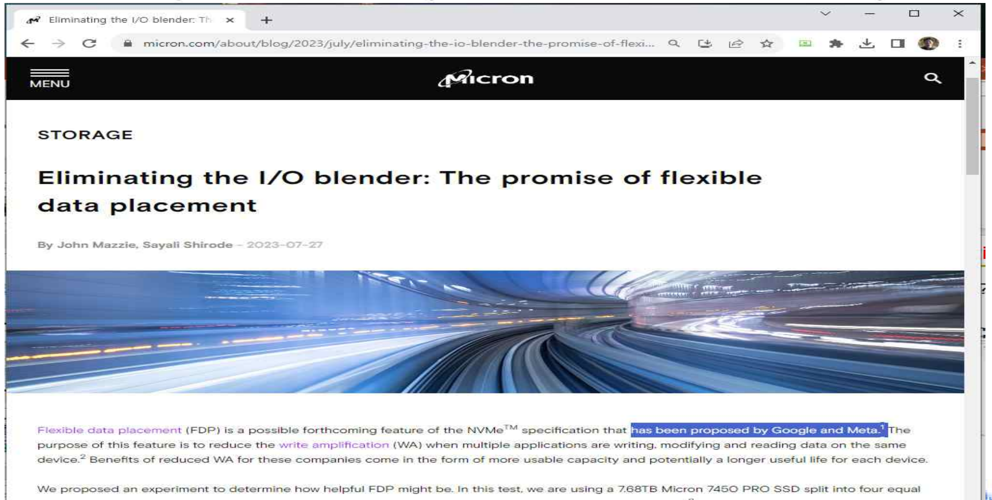
Among these, flash memory is being proliferated as the storage device for enterprise server as well as mobile devices. Furthermore, emergence of next generation NVRAM device such as PRAM, FRAM, and MRAM calls for changes in all layers of software stack. Data structures and algorithms for all software layers such as firmware, operating systems, database, and applications requires overhauling properly incorporating the physical characteristics of new NVRAM devices. File-systems and buffer caches are typical example.

October 19, 2023
Jongmoo Choi
http://embedded.dankook.ac.kr/~choijm

Contents

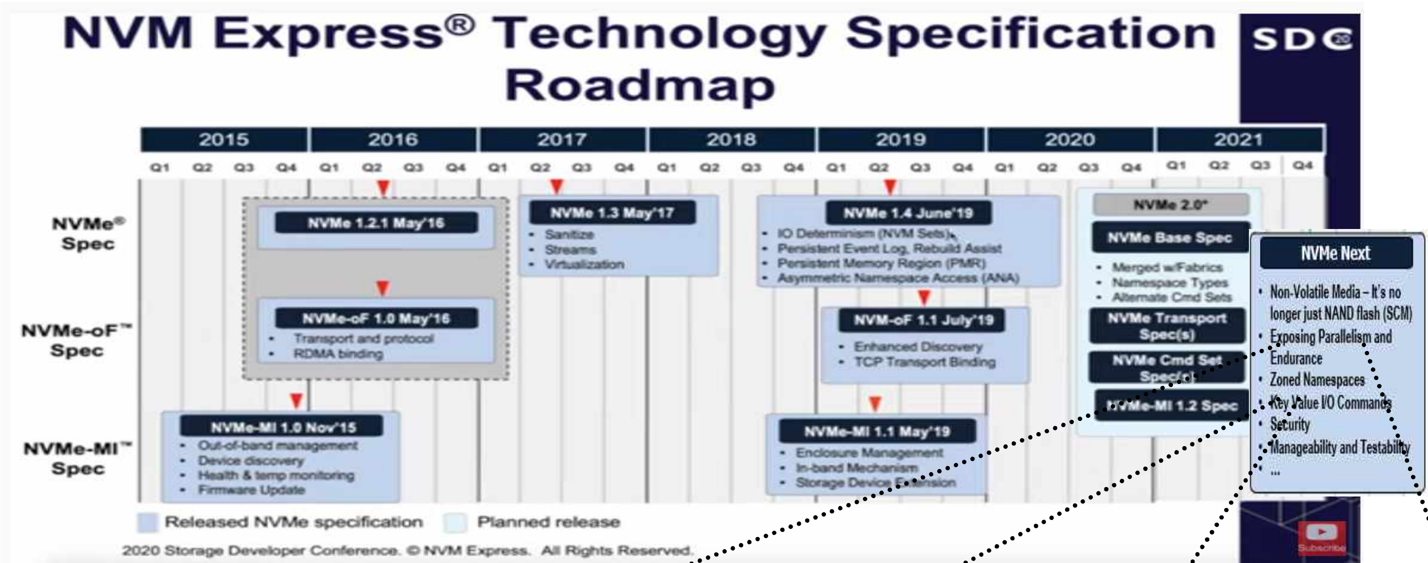
Introduction (1/5)

- What are Next-generation SSDs?
  - ✓ ZNS SSDs: one of Next-Generation SSDs
  - ✓ Some news related to Next-Generation SSDs



Introduction (2/5)

- What are Next-generation SSDs?
  - ✓ From NVMe Specification



(Source: Revised from NVMe 2.0 Spec Preview, SDC, Sep. 2020)

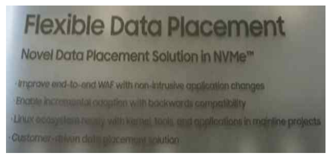
Block IF (Flash Translation Layer)	Stream IF	Zone IF	KV IF	FDP
NAND (channel, way, ...)	NAND (channel, way, ...)	NAND (channel, way, ...)	NAND (channel, way, ...)	NAND (channel, way, ...)

(Traditional SSD)

(Streamed SSD)

(ZNS SSD)

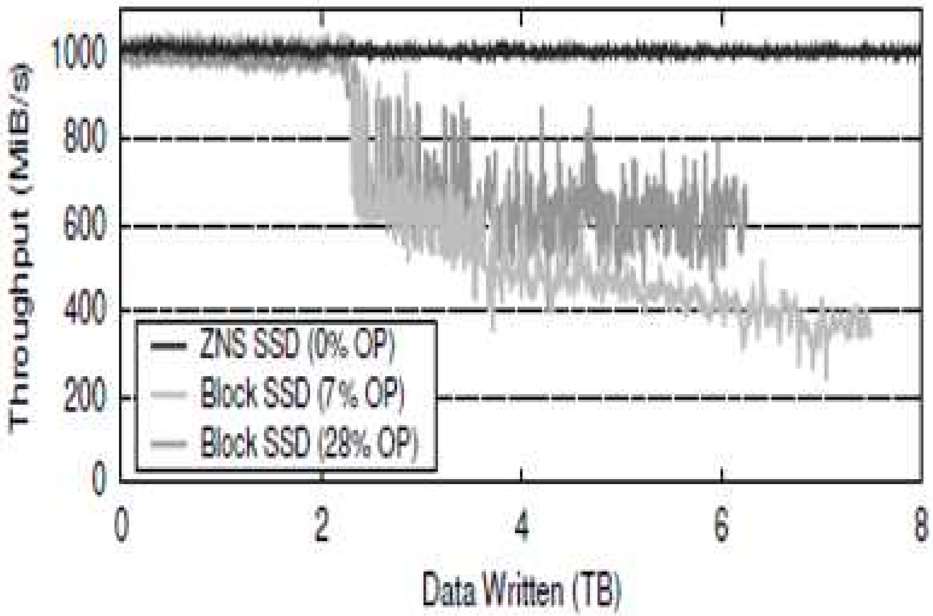
(KV SSD)



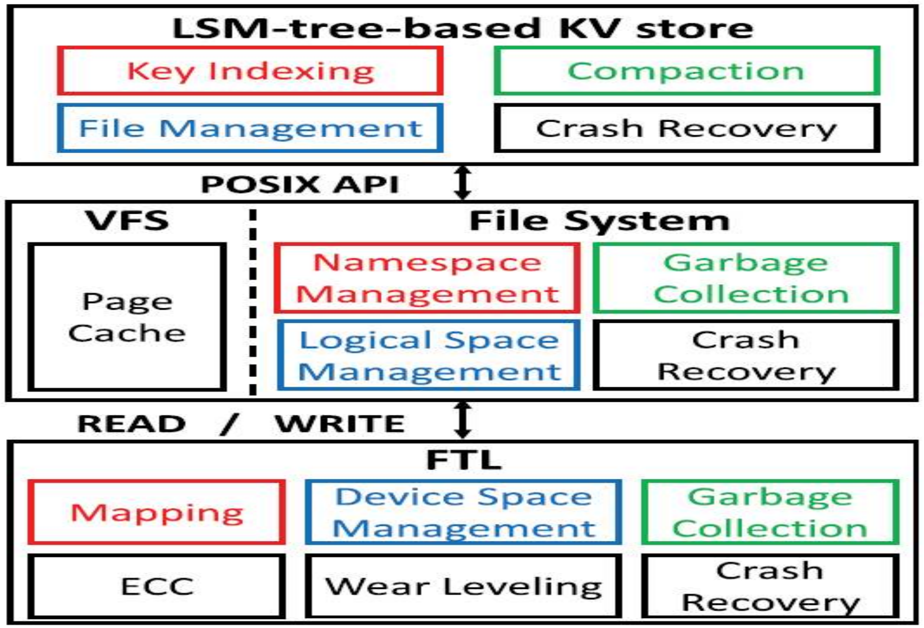
(FDP SSD)

Introduction (3/5)

- Why Next-Generation SSDs?
  - ✓ Block Interface Tax
    - Unawareness, Semantic Gap
    - Unexpected performance drop, High cost (due to OP/DRAM), …
  - ✓ Redundant Functionalities
    - FTL, FS, KV DB (or other applications)
    - Journal of Journal, Increased WAF, Lose optimization opportunities, …



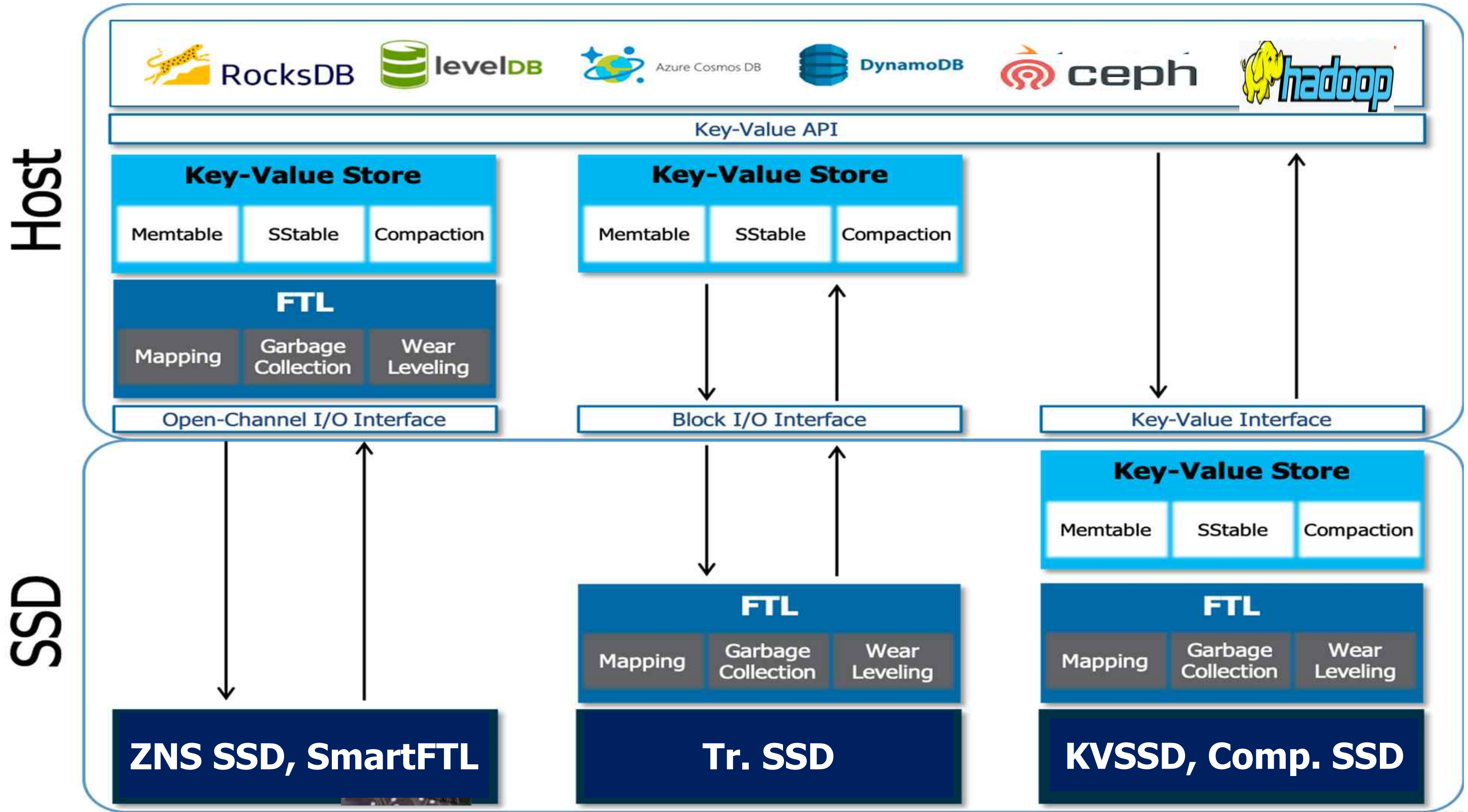
(Source: Avoiding BI Tax, ATC'21)



(Source: FlashKV, ACM TECS'17)

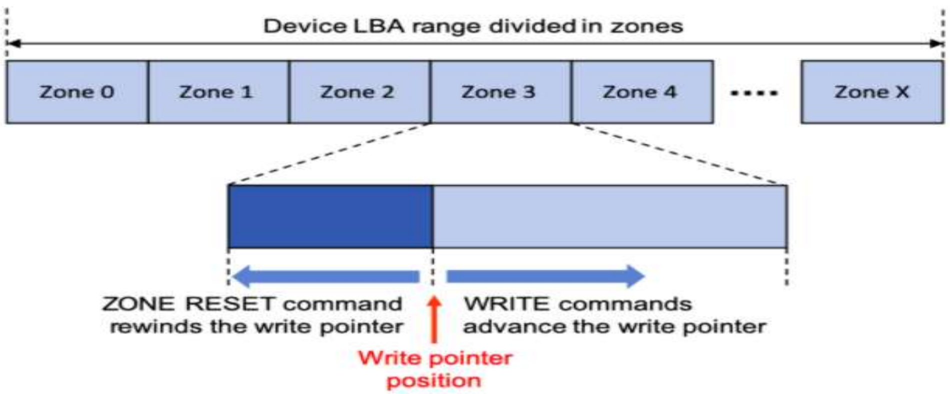
Introduction (4/5)

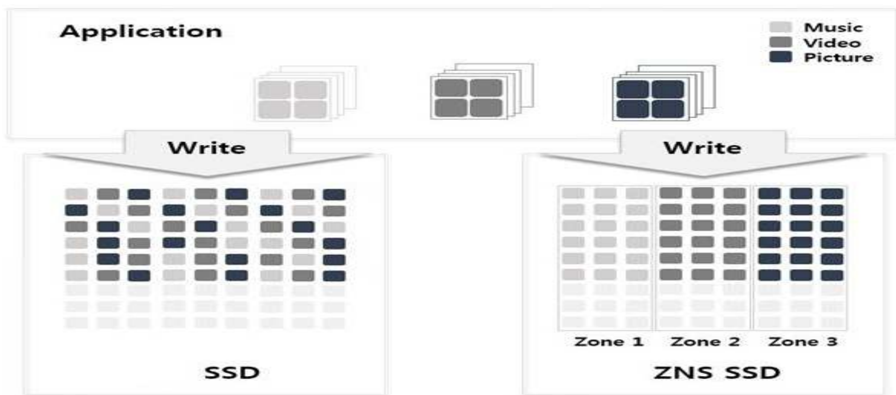
- Common Goal of Next-Generation SSDs
  - ✓ Reconsidering Storage SW Stack
    - How to manage flash memory at the host level?
    - How to realize ISP (such Key-Value Store) at the device level?



Introduction (5/5)

- ZNS SSD 101
  - ✓ Storage is divided into zones and each zone is written sequentially
    - Ratified technical proposal (TP 4053) for the NVMe™1.4a
  - ✓ Potentials
    - Workload separation ➔ reduce WAF and be predictable
    - Resource reduction in SSD (DRAM, OP) ➔ decrease TCO
  - ✓ Challenges
    - Sequential write constraint and Host-level management (e.g. zone reset, limited active zone, …)
    - How to use? (in terms of parallelism and isolation)



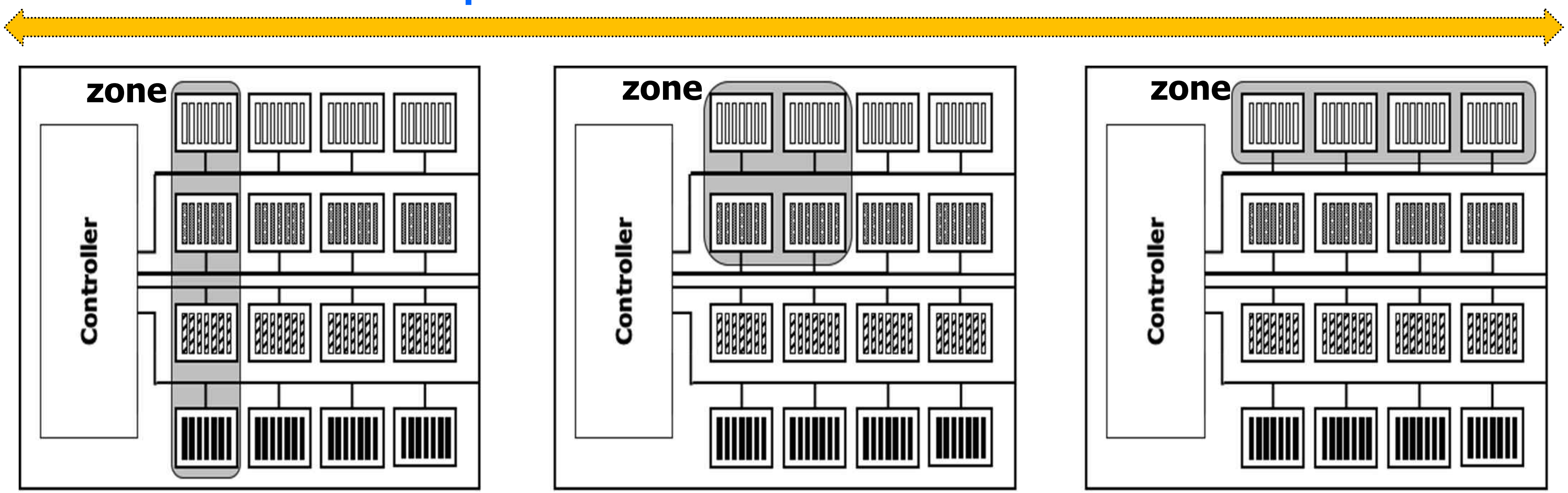


(Source: www.cdrinfo.com/d7/content/sk-hynix-demonstrated-zoned-namespaces-ssd-solution-datacenters)

Observations (1/9)

- Internal architecture of ZNS SSDs
  - ✓ Parallel Unit (PU) in SSDs
    - Channel, Way, Die, Plane, Multicores, Dual registers, …
  - ✓ How to map a zone to parallel units?
    - A spectrum from 1-to-1 relation to 1-to-all relation
    - This slide considers channels only (can be easily extended)

A Spectrum of ZNS SSDs: Zone-to-Channel



(a) 1-to-All relation (b) 1-to-N relation (c) 1-to-1 relation

Observations (2/9)

- SSDs used for experiments
  - ✓ 3 SSD prototypes

(ZSSD1: U.3)

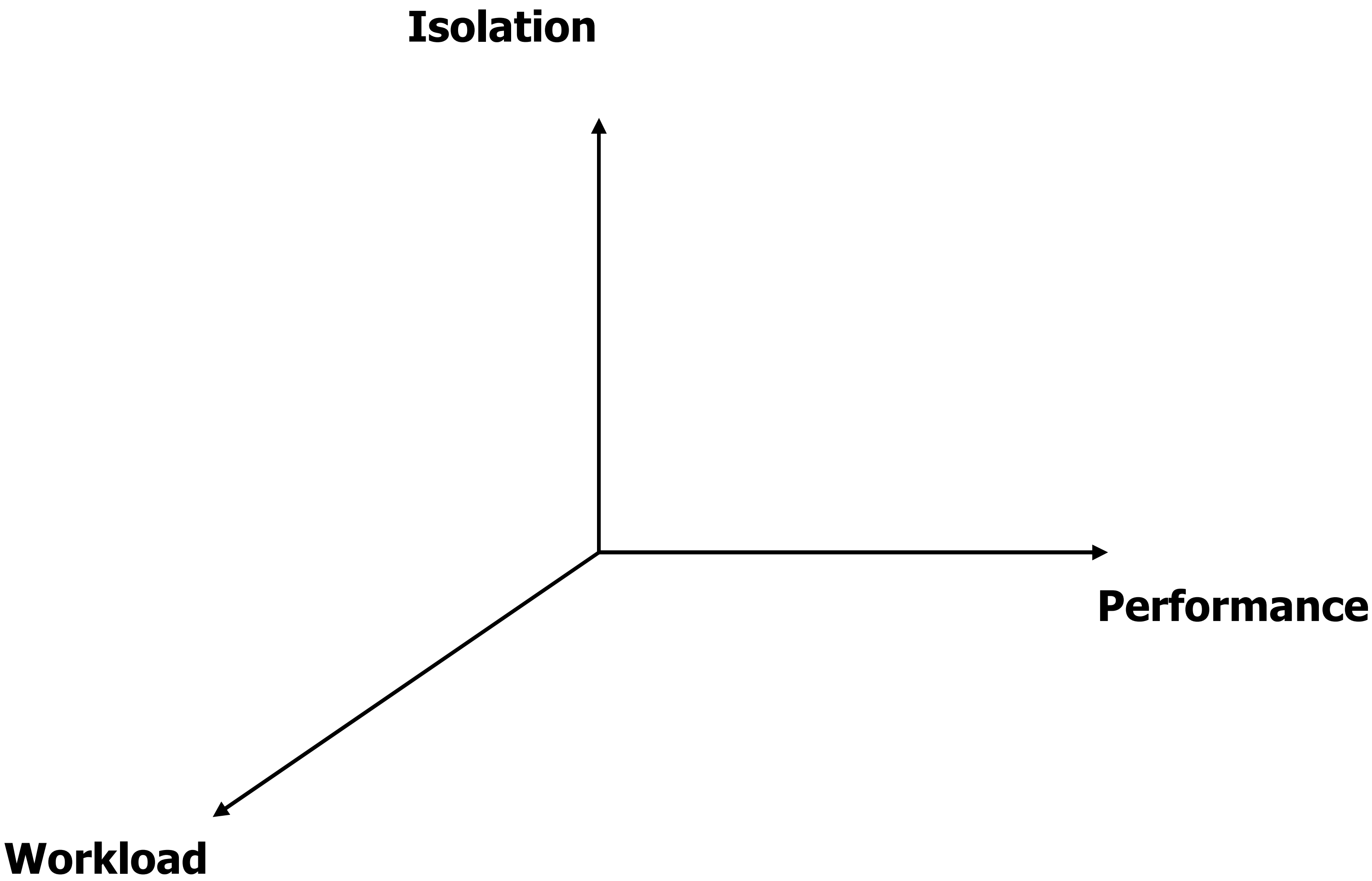
(TrSSD)

(ZSSD2: M.2)

  - ✓ Specification
    - ZSSD1: 1-to-1 relation (also called as small-zone or SU-zone)
    - ZSSD2: 1-to-all relation (also called as large-zone or FU-zone)
    - TrSSD: same hardware of ZSSD1 but different firmware

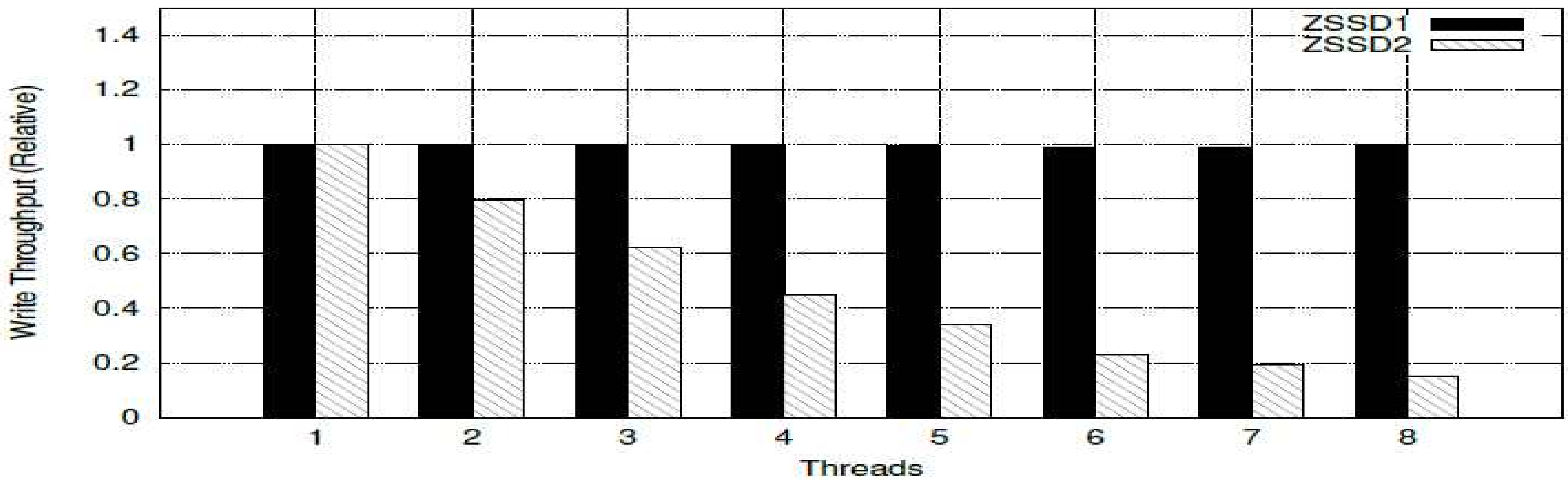
Observations (3/9)

- From what viewpoints?



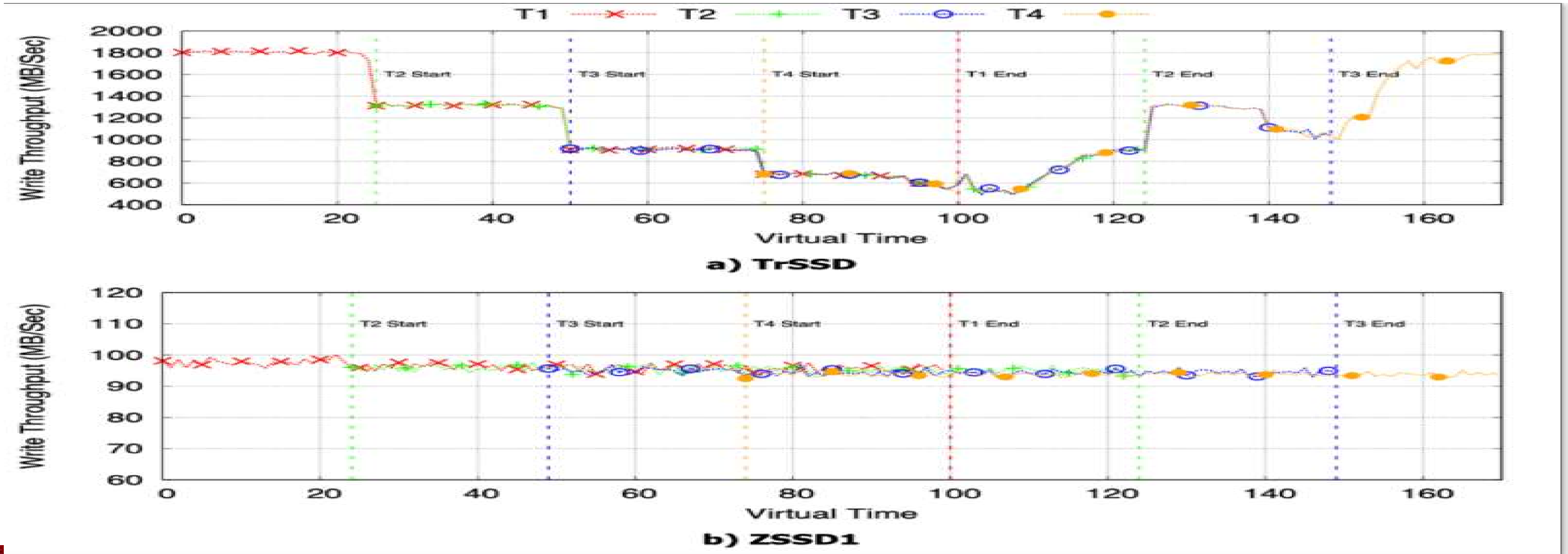
Observations (4/9)

- Isolation
  - ✓ Definition
    - How much performance is degraded when multiple zones are accessed concurrently, compared to performance of a single zone
  - ✓ Workload
    - Each thread runs on a different zone (write)
  - ✓ Observation 1
    - ZSSD1 (small-zone): Good isolation
    - ZSSD2 (large-zone): Bad isolation (also TrSSD)



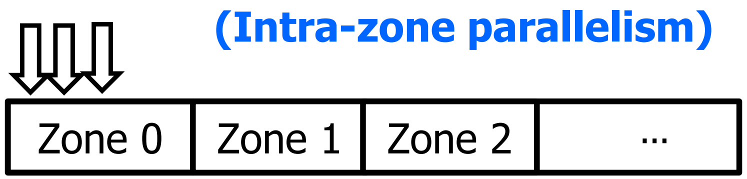
Observations (5/9)

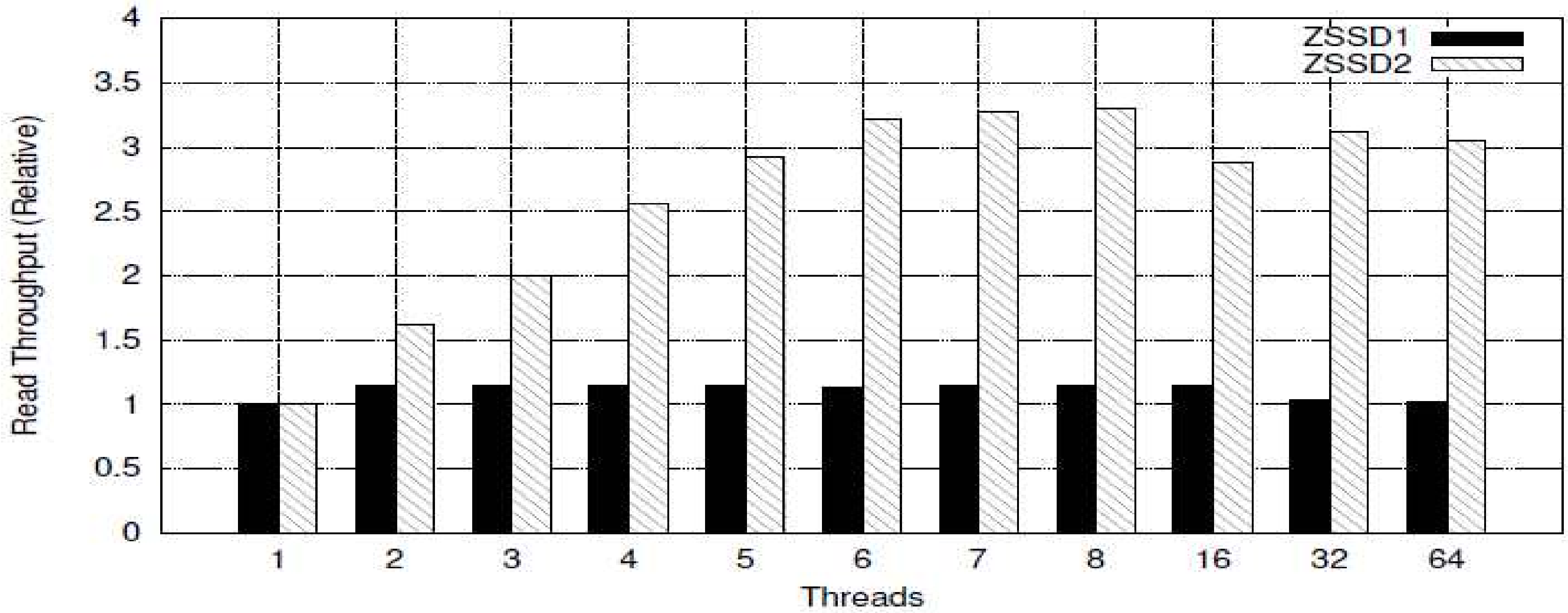
- Both Isolation and Performance
  - ✓ Workload
    - Four threads that start at different times
  - ✓ Observation 2
    - Tradeoff: Isolation vs. Performance
      - TrSSD: Bad isolation, but high performance
      - ZSSD1 (small zone): Good isolation at the cost of low performance
      - ZSSD2 (large zone): shows similar trends to TrSSD



Observations (6/9)

- Performance with SW parallelism
  - ✓ Workload
    - Intra-zone parallelism
    - Write a file and read the file using multiple threads (sync mode)
  - ✓ Observation 3
    - ZSSD1 (small-zone): Not scalable
    - ZSSD2 (large-zone): Somewhat scalable (3X)

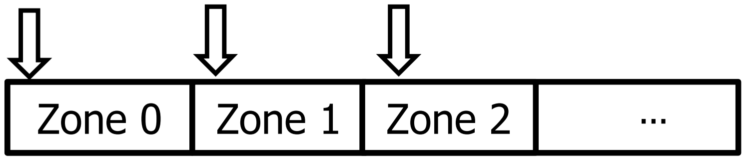


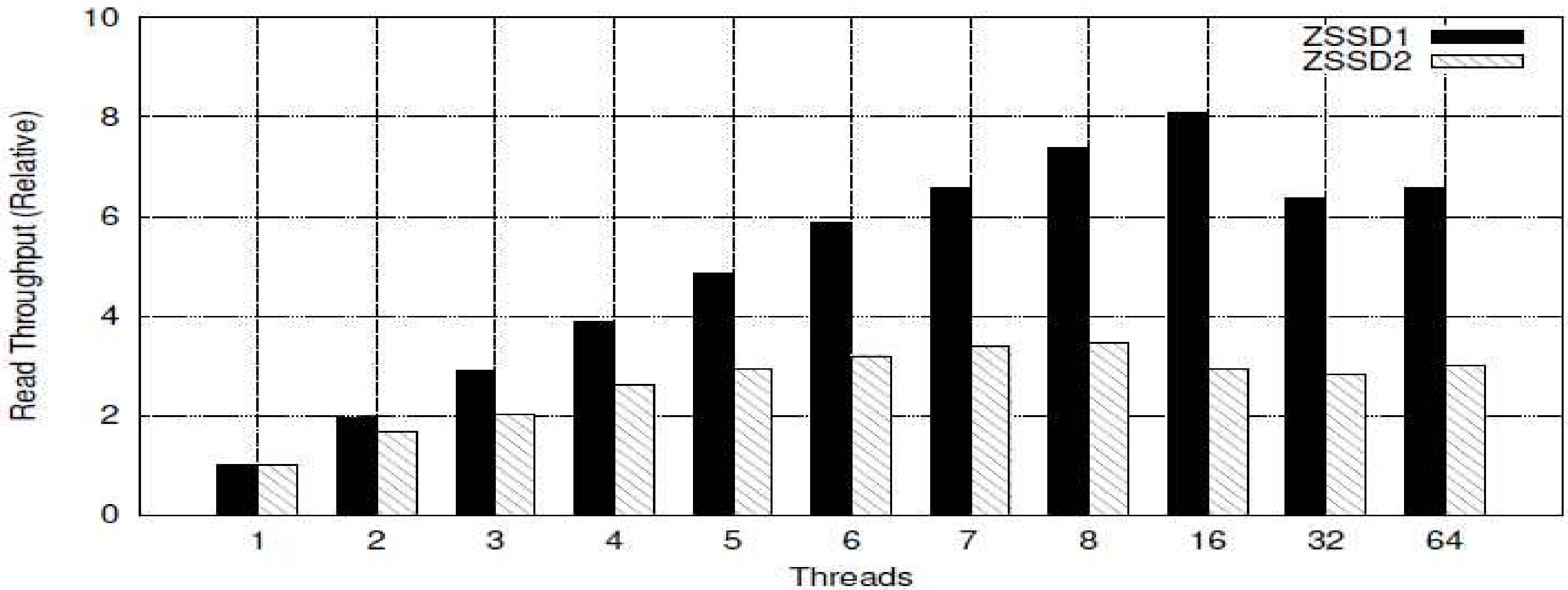


Observations (7/9)

- Performance with SW parallelism
  - ✓ Workload
    - Inter-zone parallelism
    - Distribute a file into multiple zones and read the file using multiple threads.
  - ✓ Observation 4
    - ZSSD1 (small-zone): Good scalable (8X)
    - ZSSD2 (large-zone): Somewhat scalable (3X)

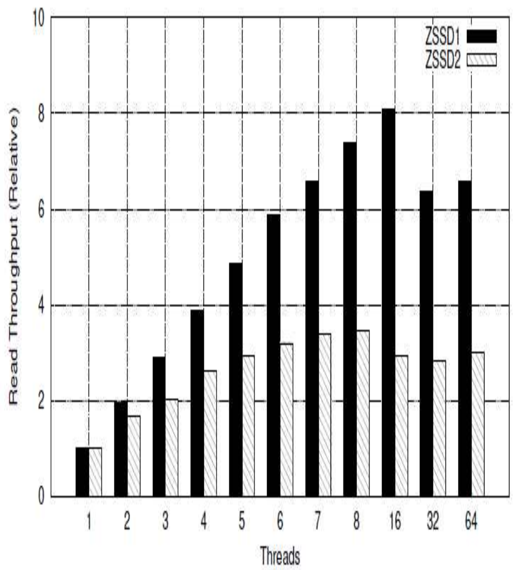
(Inter-zone parallelism)



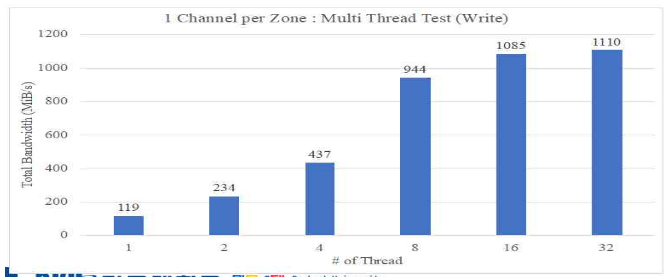


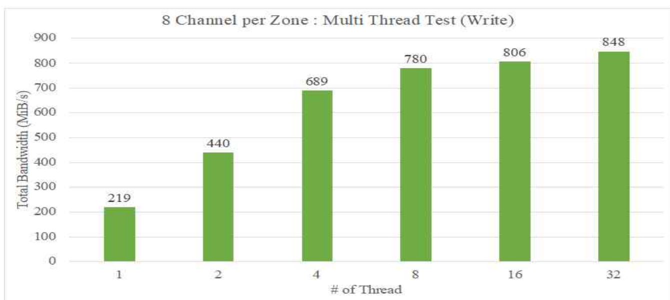
Observations (8/9)

- Workload sensitivity
  - ✓ Sync vs. Async or IO Depth, Request size, …
  - ✓ 1) Sync vs. Async with inter-zone parallelism (128KB request size)
    - 1-to-1 relation: 8X as threads increase, same under sync. & async.
    - 1-to-all relation: 3X as threads increase under sync. vs. max at the 1 thread under async.
  - ✓ 2) Request size: quite important factor
    - "Request size > page size x PU" : max even at 1 thread on 1-to-all (sync)

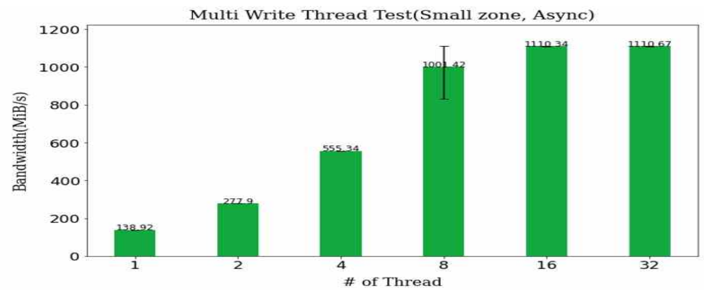


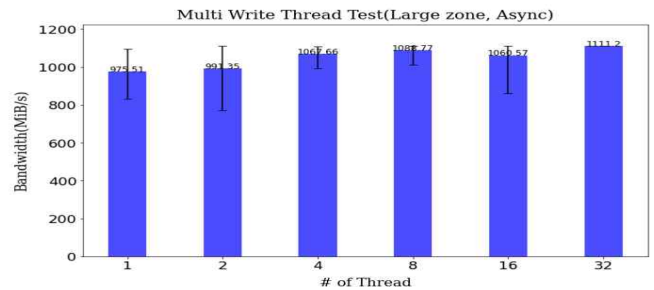
(Observation 4)





(Synchronous)



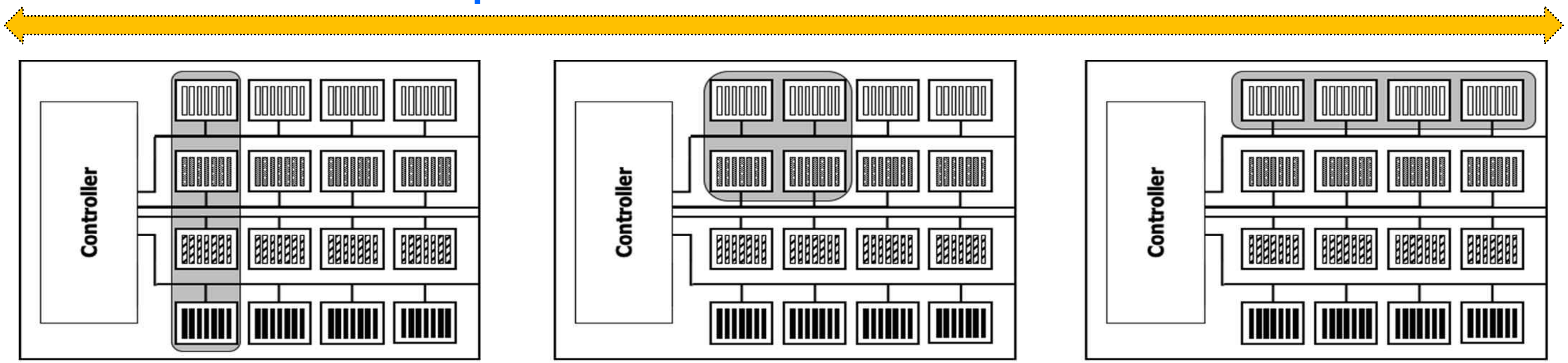


(Asynchronous)

Observations (9/9)

- Summary
  - ✓ 1-to-all relation
    - Good performance (even for a single thread), but bad isolation
    - Similar to TrSSD ➔ not heavy traditional SW modification
  - ✓ 1-to-1 relation
    - Good isolation, but bad performance
    - Need inter-zone parallelism (zone-aware data placement) for enhanced performance

A Spectrum of ZNS SSDs: Zone-to-Channel

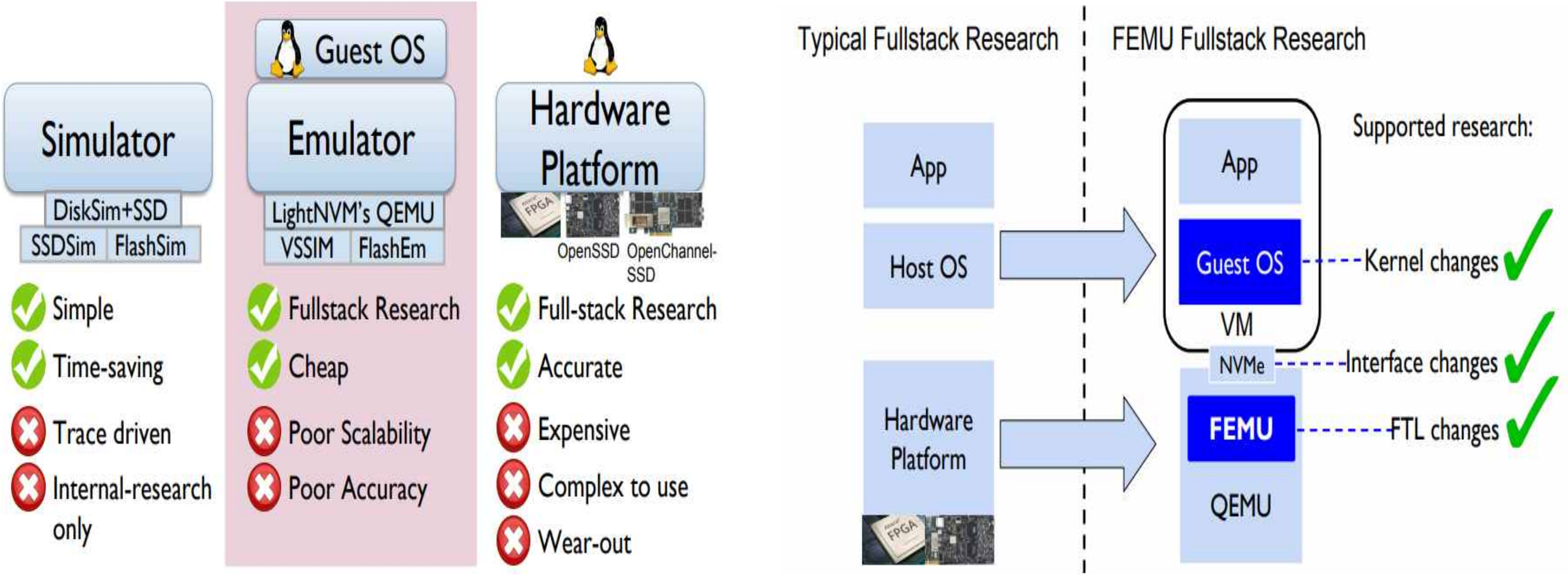


(a) 1-to-All relation (b) 1-to-N relation (c) 1-to-1 relation

☞ **How about 1-to-N (or hybrid)? How about real applications (e.g. RocksDB)? Best use case? ...**

ZNS SSD Emulator (1/8)

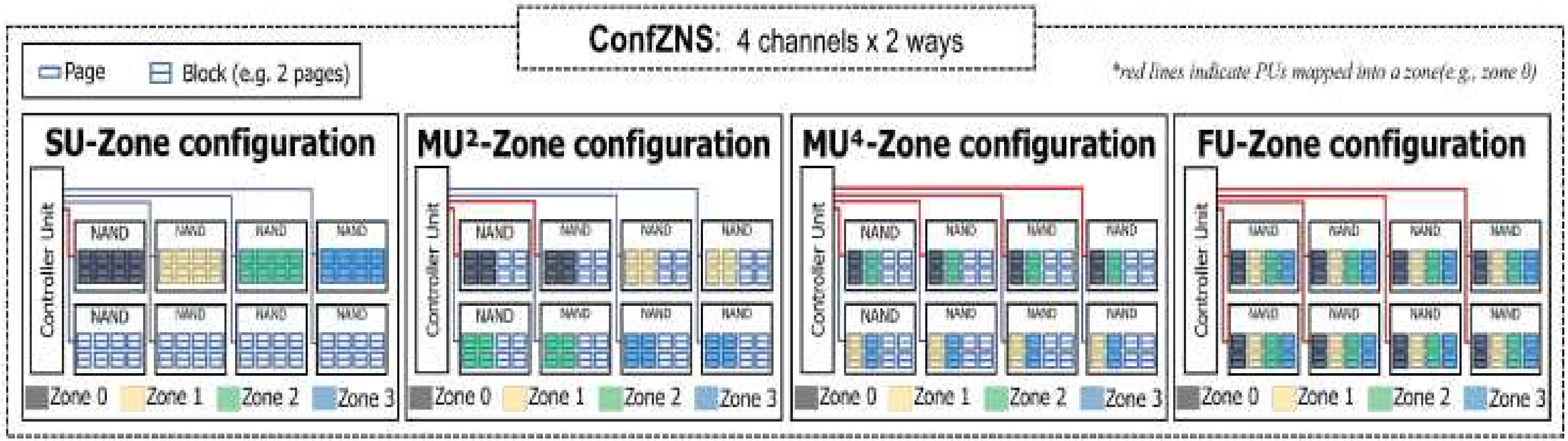
- Requirement of ZNS SSD emulation
  - ✓ Explore various design space
    - How a zone can be mapped into PUs?
    - How ZNS SSD internals affect host SW?
  - ✓ Based on FEMU (Flash Emulator using Qemu)
    - Support CASE: Cheap, Accurate, Scalable, Extensible (Full stack)



(Source: FEMU, FAST, 2018)

ZNS SSD Emulator (2/8)

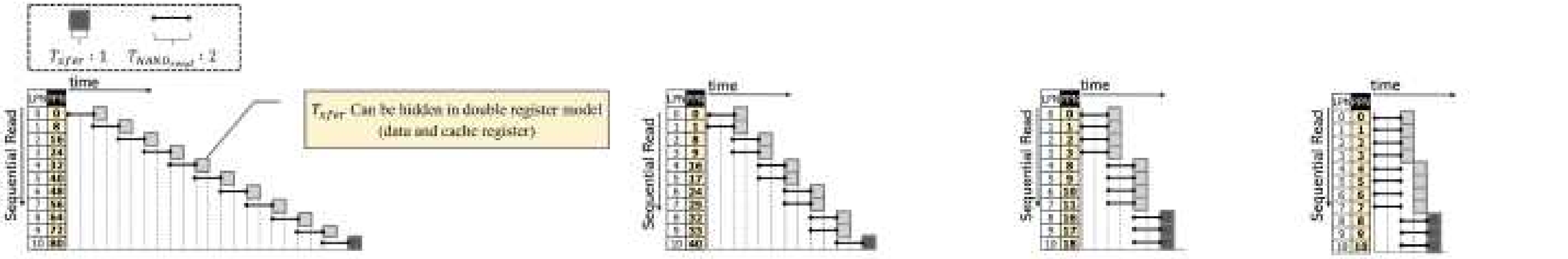
- ConfZNS: FEMU ZNS SSD extension
  - ✓ 1) Support spectrum: diverse Zone-to-PU mappings
    - SU (Single Unit)-zone: 1 zone to 1 unit (stride addressing)
    - MU (Multiple Unit)-zone: 1 zone to multiple units (linear + stride addressing)
    - FU (Full Unit)-zone: 1 zone to full units (linear addressing)
    - E.g.: SSD: 4-channels 2 ways
  - ✓ 2) Support accuracy



(Spectrum of internal architecture supported by ConfZNS)

ZNS SSD Emulator (3/8)

- ConfZNS: FEMU ZNS SSD extension
  - ✓ 1) Support spectrum: diverse Zone-to-PU mappings
  - ✓ 2) Support accuracy
    - Modeling on diverse configurations and parameters
      - Consider contention among PUs

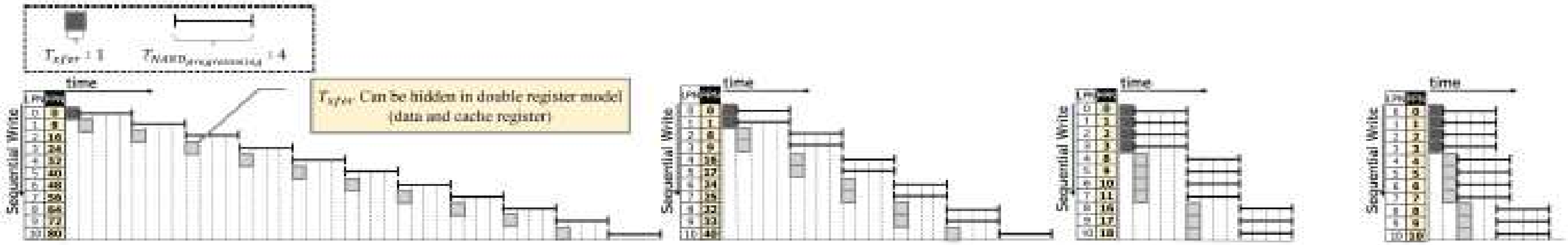


(a)SU-Zone read latency (PU: 1 channel 1way)

(b) MU²-Zone read latency (PU: 2 channels 1way)

(c) MU⁴-Zone read latency (PU: 4 channels 1way)

(d) FU-Zone read latency (PU: 4 channels 2ways)



(a)SU-Zone write latency (PU: 1 channel 1 way)

(b) MU²-Zone write latency (PU: 2channels 1way)

(c) MU⁴-Zone write latency (PU: 4 channels 1way)

(d) FU-Zone write latency (PU: 4 channels 2way)

(Modeling)

ZNS SSD Emulator (4/8)

- ConfZNS: FEMU ZNS SSD extension
  - ✓ 1) Support spectrum: diverse Zone-to-PU mappings
  - ✓ 2) Support accuracy
    - Algorithm: make use of multiple clocks
      - gclock, lclock_ch, lclock_way, …
      - gclock ticks at each time
      - lclock_ch advances when it is requested
        - if (busy)

          lclock_ch = lclock_ch + T_{XFER}

          else /* idle */

          lclock_ch = gclock + T_{XFER}
        - Consider unit dependency

          lclock_ch = max(lclock_all) + T_{XFER}
      - Completion condition
        - lclock_ch == gclock
      - Busy/Idle condition
        - if (lclock_ch > gclock)

          busy

          else

          idle

Algorithm 1: Write Latency Calculation

```
input: req.lpn : logical page number for req request
input: N_CH: number of channels
input: N_WAY: number of ways
input: N_zc: zone-channel association
output: req.ctime : completion time of req request
Data: Lclock_i^CH: local clock for i-th channel
Data: Lclock_ij^WAY: local clock for i-th channel, j-th
      way
Data: Gclock: global clock
/* Initialize global and local clocks */
For all i, j,
Gclock = 0, Lclock_i^CH = 0, Lclock_ij^WAY = 0
while true do
    /* Repeatedly handle request from the submission
       queue */
    ppn = CalculatePPN(req.lpn, N_CH, N_WAY, N_zc)
    i = Mod( ppn, N_CH )
    q = ⌊ppn/N_CH⌋
    j = Mod( q, N_WAY )
    if Lclock_i^CH ≤ Gclock then
        if Lclock_ij^WAY ≤ Gclock then
            /* both channel i and way j are idle */
            Lclock_i^CH = Gclock + T_XFER
            Lclock_ij^WAY = Lclock_i^CH + T_PROGRAMMING
        else
            /* channel i is idle, but way j is busy */
            Lclock_i^CH = Gclock + T_XFER
            Lclock_ij^WAY = max { Lclock_i^CH, Lclock_ij^WAY }
                            + T_PROGRAMMING
        end
    else
        if Lclock_j^WAY ≤ Gclock then
            /* channel i is busy, and way j is idle */
            Lclock_i^CH = Lclock_i^CH + T_XFER
            Lclock_ij^WAY = Lclock_i^CH + T_PROGRAMMING
        else
            /* both channel i and way j are busy */
            Lclock_i^CH = Lclock_i^CH + T_XFER
            Lclock_ij^WAY = max { Lclock_i^CH, Lclock_ij^WAY }
                            + T_PROGRAMMING
        end
    end
    req.ctime = Lclock_ij^WAY
    Gclock++
    if req.ctime ≥ Gclock then
        Notify user of request completion
    end
end
```

(Algorithm)

ZNS SSD Emulator (5/8)

- Validation 1
  - ✓ 1) Two real ZNS SSDs, ConfZNS with two configurations

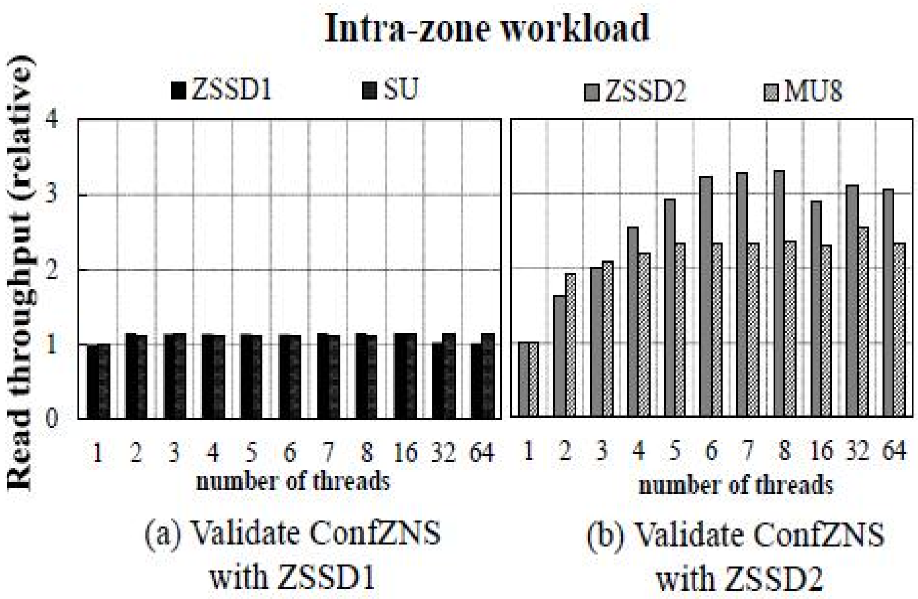
Device	Capacity	Interface	Zone size	# of Zones
ZSSD1	2TB	PCIe Gen3	72MB	29,172
ZSSD2	1TB	PCIe Gen3	1.6GB	530

Table 1: Two real ZNS SSDs used for validation.

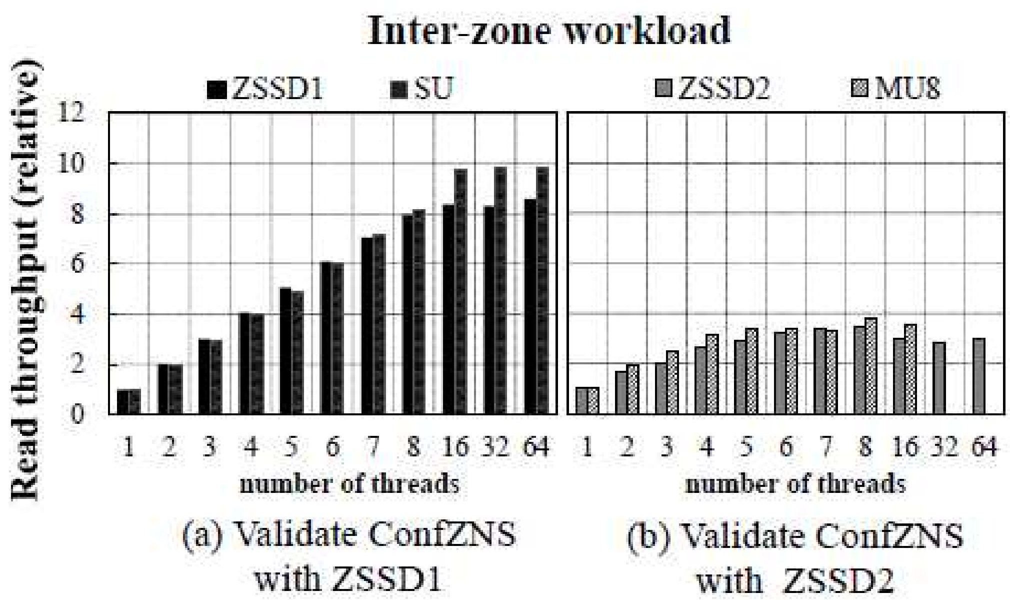
Configuration	Zone size	PU	Page size	T_{pgm}, T_{read}
SU	72MB	16	48KB	450us, 65us
MU[8]	1.6GB	16	48KB	450us, 65us

Table 2: Configurations for ConfZNS (PU stands for the number of Parallel Units. We set T_{xfer} as 1200MT/s.)

  - ✓ 2) Accuracy: 8% error on average



(Validation: intra-zone)



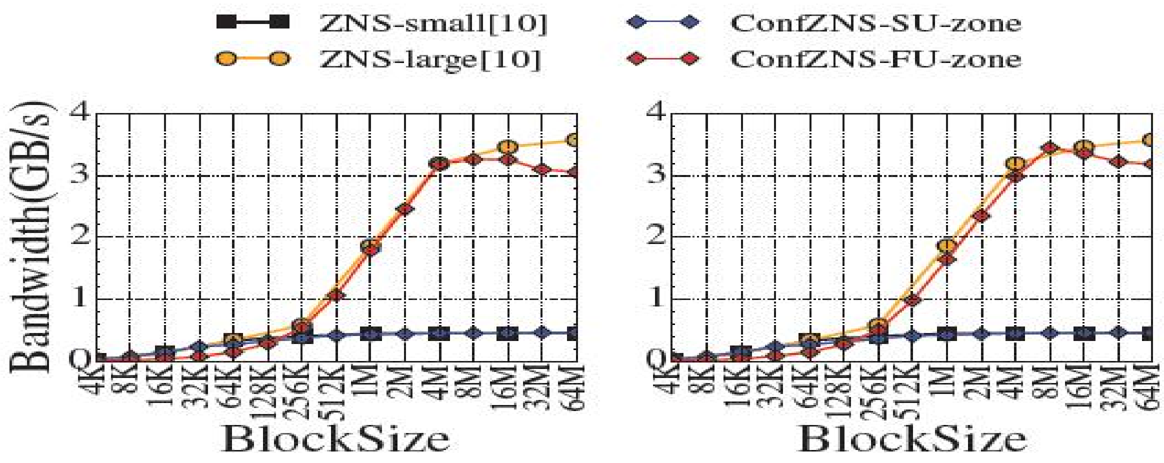
(Validation: inter-zone)

ZNS SSD Emulator (6/8)

- Validation 2
  - ✓ 1) Data from previous study
    - Bae et al., “What You Can't Forget: Exploiting Parallelism for Zoned Namespaces”, HotStorage’22

ConfZNS configuration		
Zone Configuration	*SU*-zone	*FU*-zone
Zone size	96MB	1.5GB
Channels per zone	1	8
Ways per zone	1	2

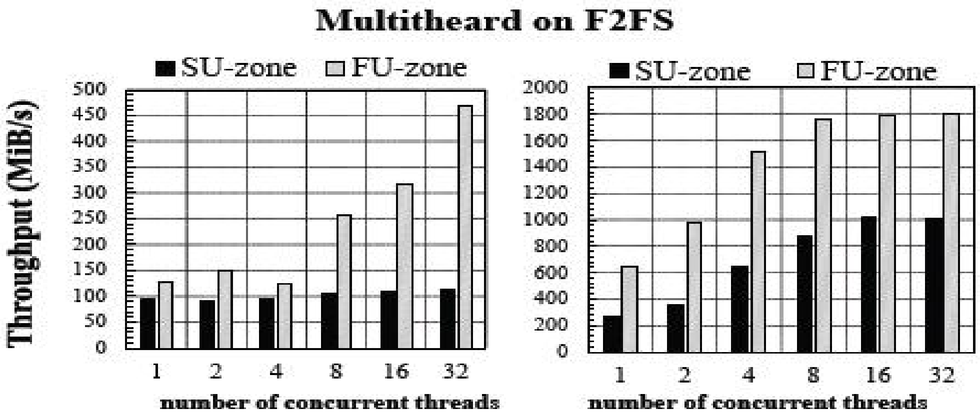
  - ✓ 2) Accuracy: show similar trends (black-box approach)



(a) Sequential read (b) Random read

ZNS SSD Emulator (7/8)

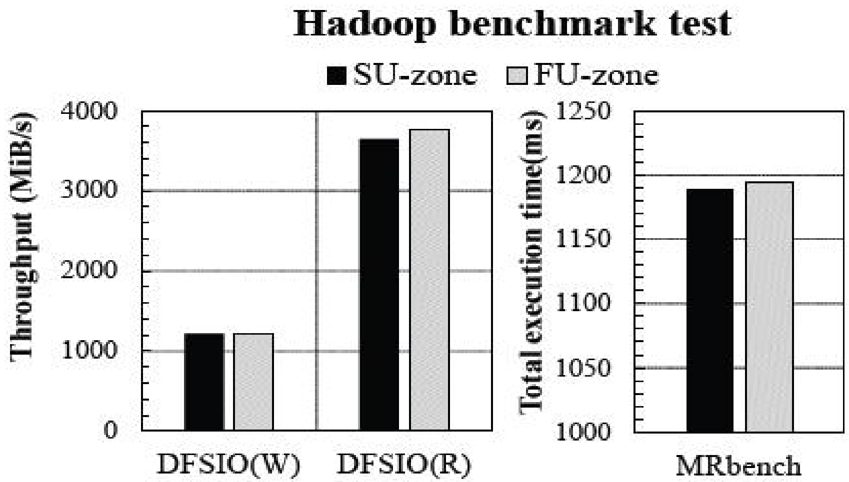
- Host SW analysis
  - ✓ F2FS on ConfZNS
    - Fio benchmark
      - Sync, iodepth=1, iosize=64MB, blocksize=128KB, numjobs=thread
      - FU-zone: scalable as threads increase
      - SU-zone: less scale (F2FS seems to utilize intra-zone parallelism for write (1X) and inter/intra-zone parallelism for read (3X))
    - Hadoop benchmark
      - Create 128MB files concurrently: distribute files into different zones
      - SU-zone comparable to FU-zone ➔ need zone-awareness for SU-zone



(a) ConfZNS FIO write on F2FS

(b) ConfZNS FIO read on F2FS

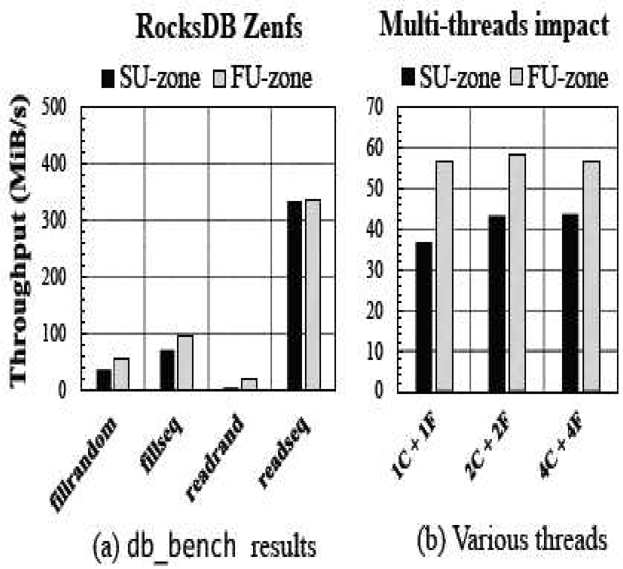
(Fio Benchmark)

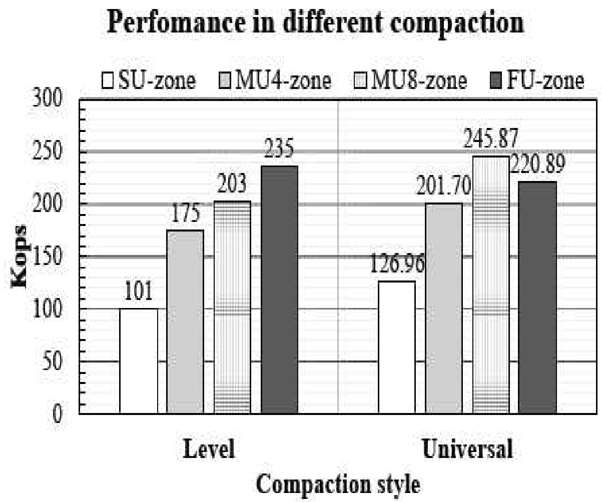


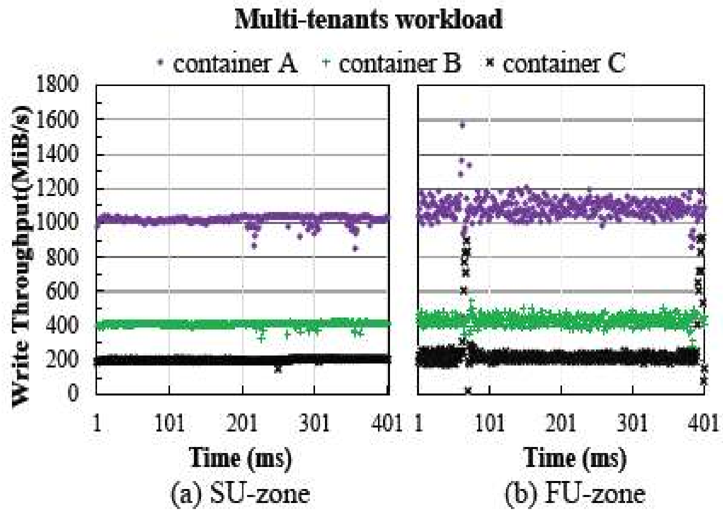
(Hadoop Benchmark)

ZNS SSD Emulator (8/8)

- Host SW analysis
  - ✓ RocksDB on ConfZNS
    - Using ZenFS (ATC'21 paper)
    - Can compare different configurations under different policies
      - Workload, thread (flush/compaction), compaction policy, ZNS configuration
  - ✓ Multi-tenants workload on ConfZNS
    - Require different QoSs: 200MB/s, 400MB/s, and 1000MB/s
    - Allocate different number of zones according to QoS







(RocksDB) **(Multi-tenants workload)**

RocksDB on ZNS SSDs (1/9)

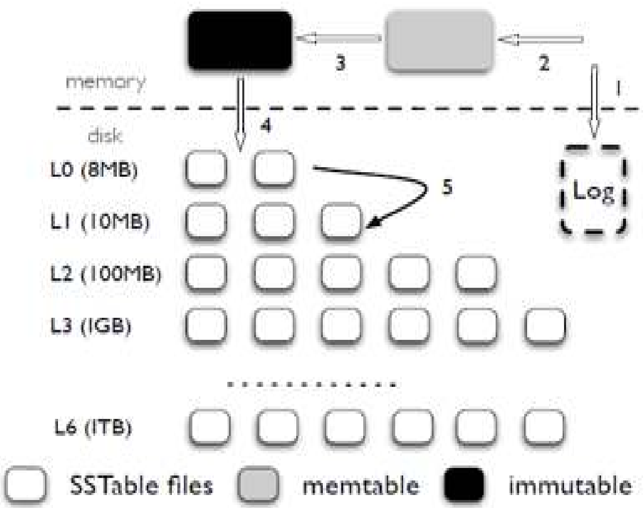
- Key Value Store + ZNS SSD
  - ✓ Well matched (gganbu)

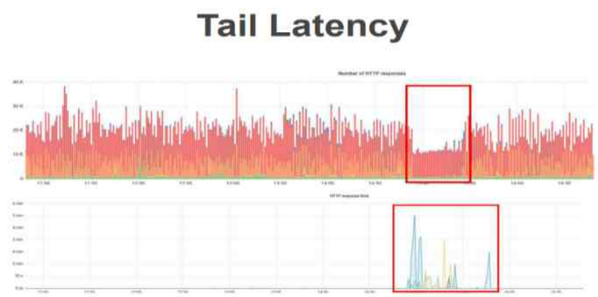
KVS

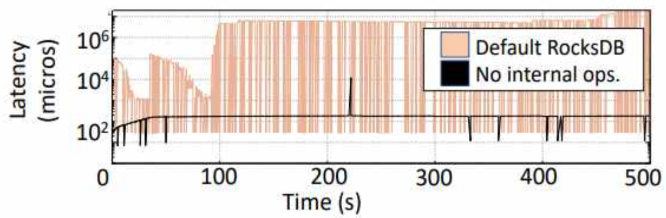
- Based on LSM-tree
- Level differences
- Interference problem

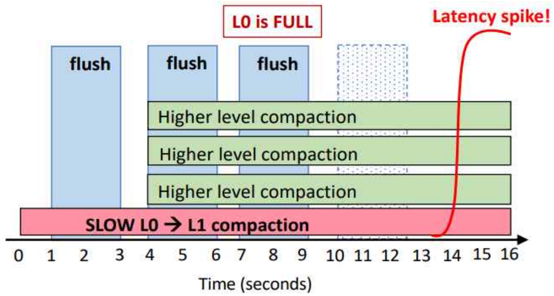
ZNS SSD

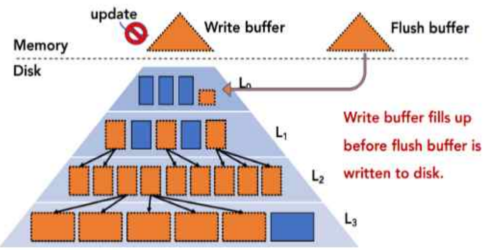
- Large sequential writes
- Workload separation
- Isolation











(Source: Wisckey paper in FAST'16 and SILK papers in ATC'19)

RocksDB on ZNS SSDs (2/9)

- ZenFS (from ATC'21)
  - ✓ A new storage backend for RocksDB
    - Extent, Journal, Log (look like a simple version of Ext4)
    - Based on Large-zone ZNS SSDs
  - ✓ Evaluation **➔ How about small-zone ZNS SSDs?**
    - 4 setup: 1) XFS on TrSSD, 2) F2FS on TrSSD, 3) F2FS (ZNS), 4) ZenFS

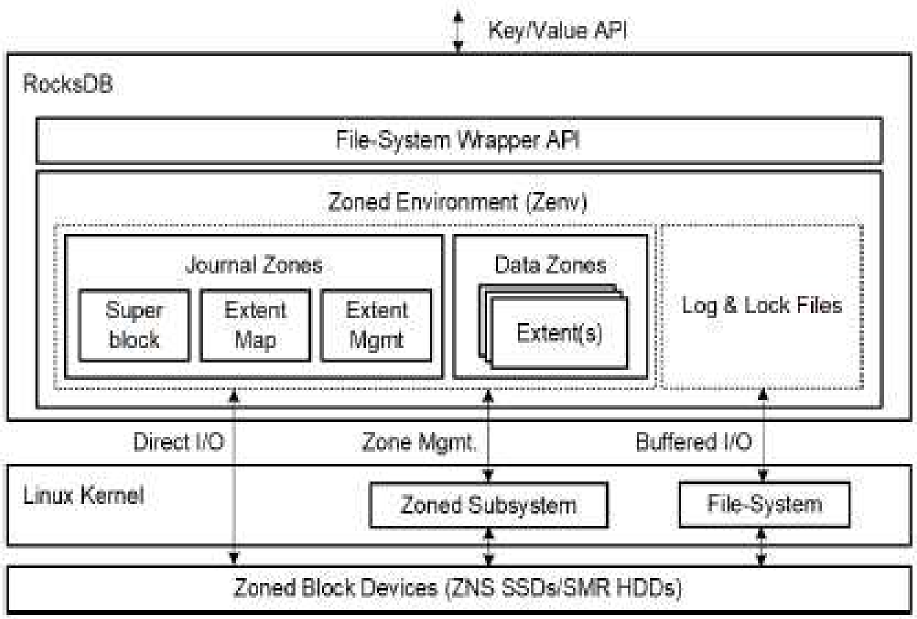


Figure 4: ZenFS Architecture

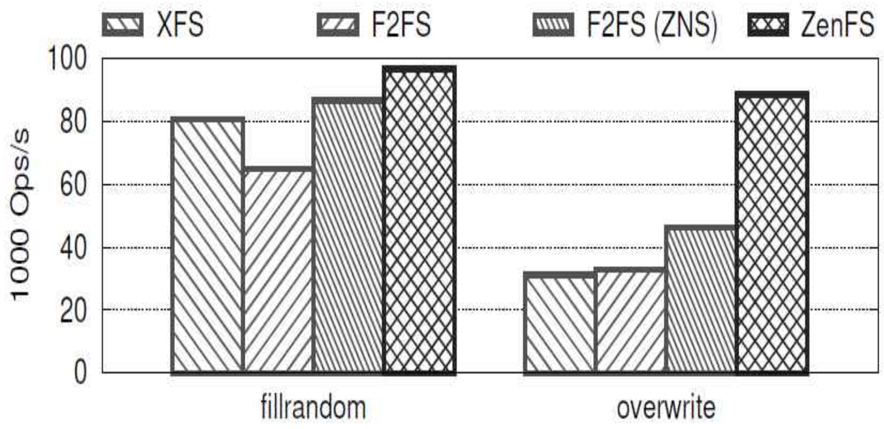
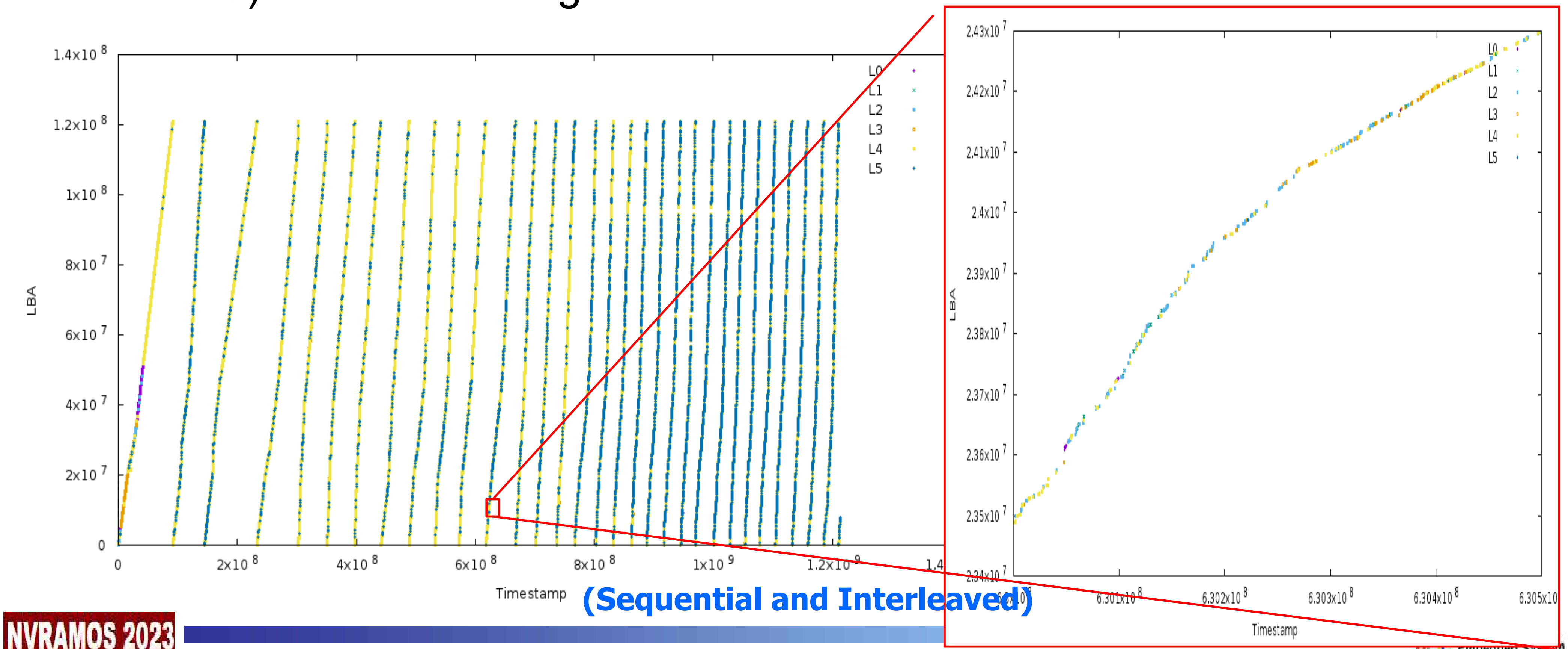


Figure 6: Throughput of RocksDB with write-heavy benchmarks— *fillrandom* followed by *overwrite* using the block-interface SSD with 28% OP and the ZNS SSD.

(Source: Avoiding BI Tax, ATC'21)

RocksDB on ZNS SSDs (3/9)

- Motivation: RocksDB workload analysis
  - ✓ 1) Sequential pattern
    - Good: go well with ZNS
    - Issues: 1) mainly intra-zone(which is bad on small-zone), 2) level mixed
  - ✓ 2) Interference of compaction to flush
  - ✓ 3) Hotness among levels



(Sequential and Interleaved)

RocksDB on ZNS SSDs (4/9)

- Motivation: RocksDB workload analysis
  - ✓ 1) Sequential pattern
  - ✓ 2) Interference of compaction to flush
    - Compaction: read, merge, and write ➔ time consuming job
    - Delayed flush incurs latency spike of user requests
  - ✓ 3) Hotness among levels
    - Hotter as lower level

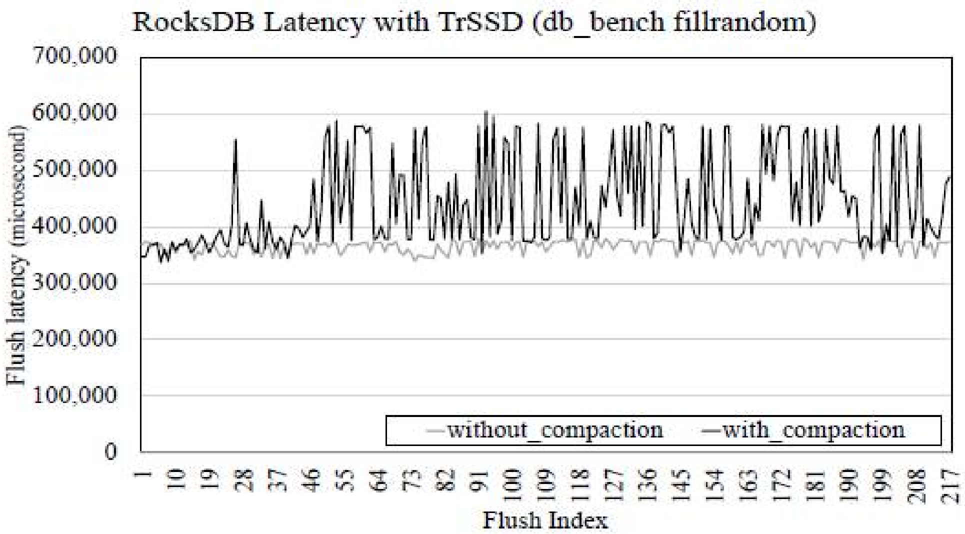
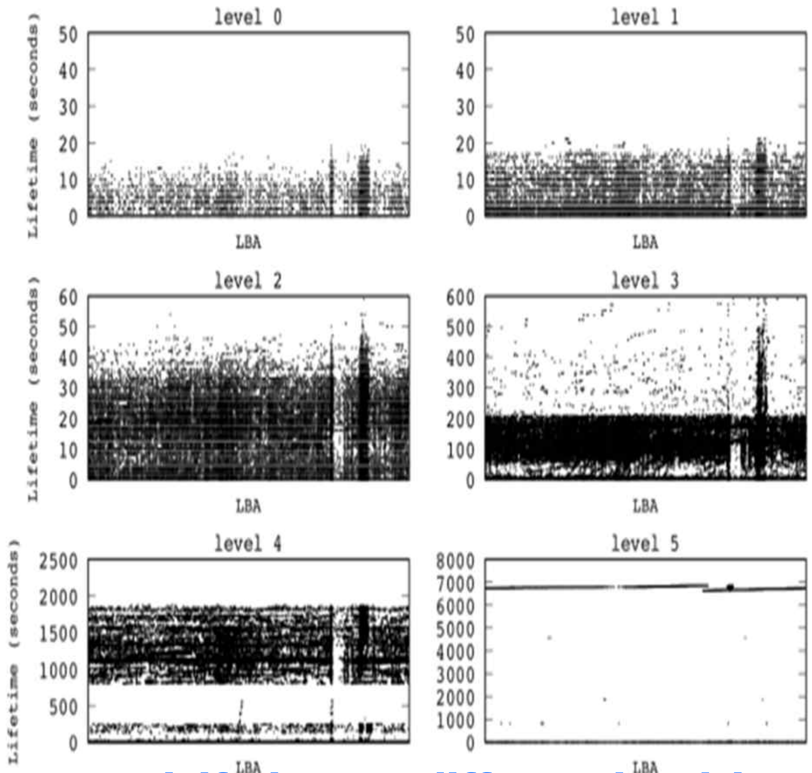


Figure 3: Flush latency of RocksDB with TrSSD

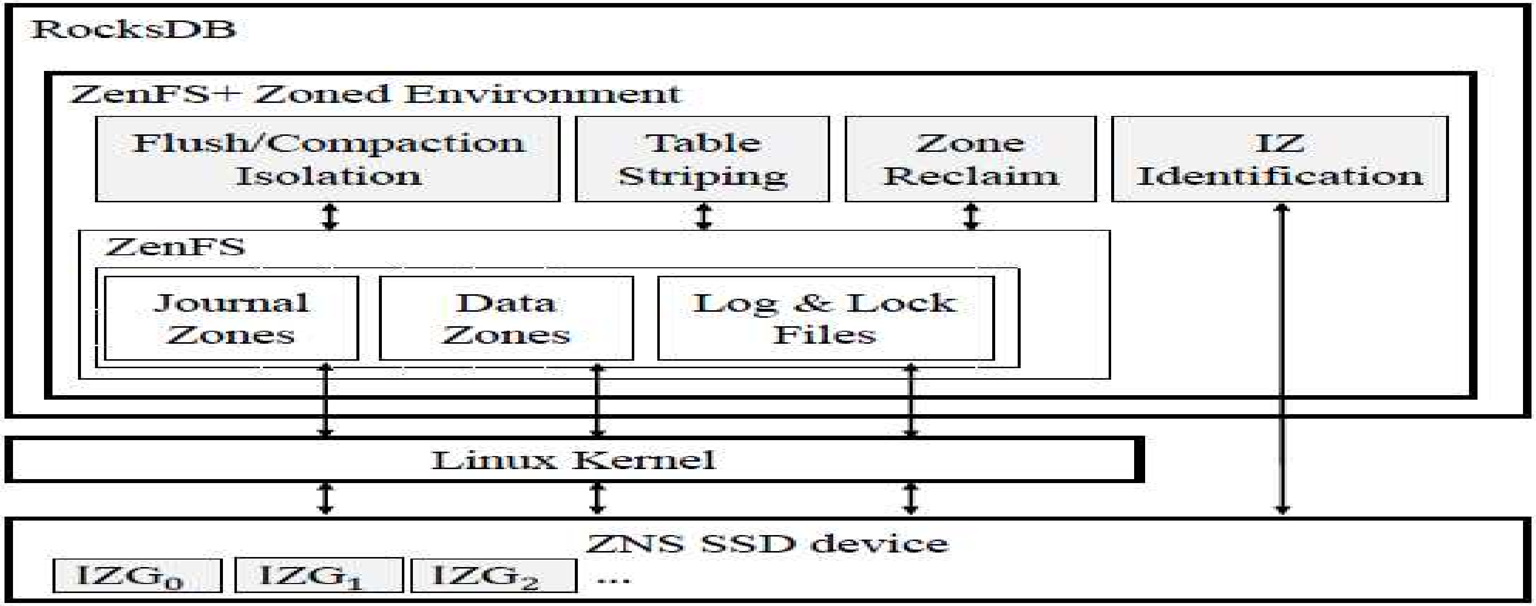
(Interference)



(Lifetime on different levels)

RocksDB on ZNS SSDs (5/9)

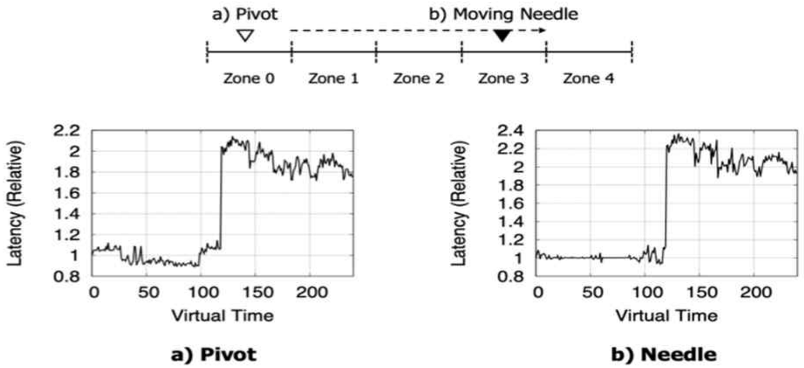
- RocksDB optimization for Small-zone: ZenFS+
  - ✓ Idea 1: Flush and compaction isolation
    - Identify IZs (Independent zones) and allocate in an isolated manner
    - Dynamic vs Static
  - ✓ Idea 2: Table striping
    - exploit inter-zone parallelism
  - ✓ Idea 3: Separate higher levels from lower levels
    - For efficient zone reclaiming (minor/major reclaim)



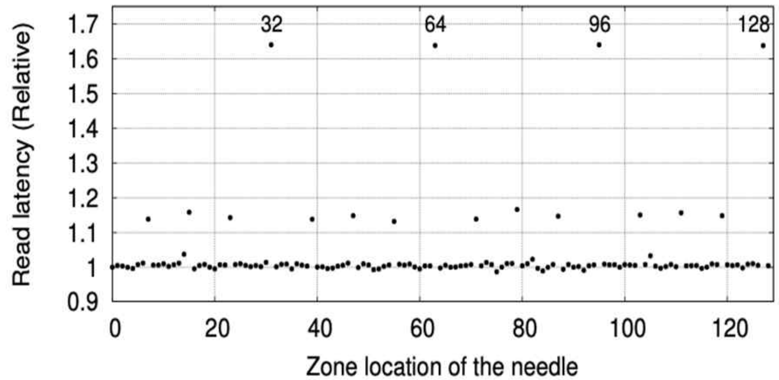
IZG : Independent Zone Group

RocksDB on ZNS SSDs (6/9)

- RocksDB optimization for Small-zone: ZenFS+
  - ✓ Idea 4: Independent Zone Identification Technique
    - What is the Independent Zone?
      - Zones that are not interfered with
      - Important for isolation and striping
    - How to?
      - Based on Latency (or Power consumption)
      - Pivot: stay a zone, Needle: move zones ➔ Both read at the same time
    - Identification
      - Latency jump ➔ dependent zone
      - This technique can be used to explore internals of other ZNS SSDs.



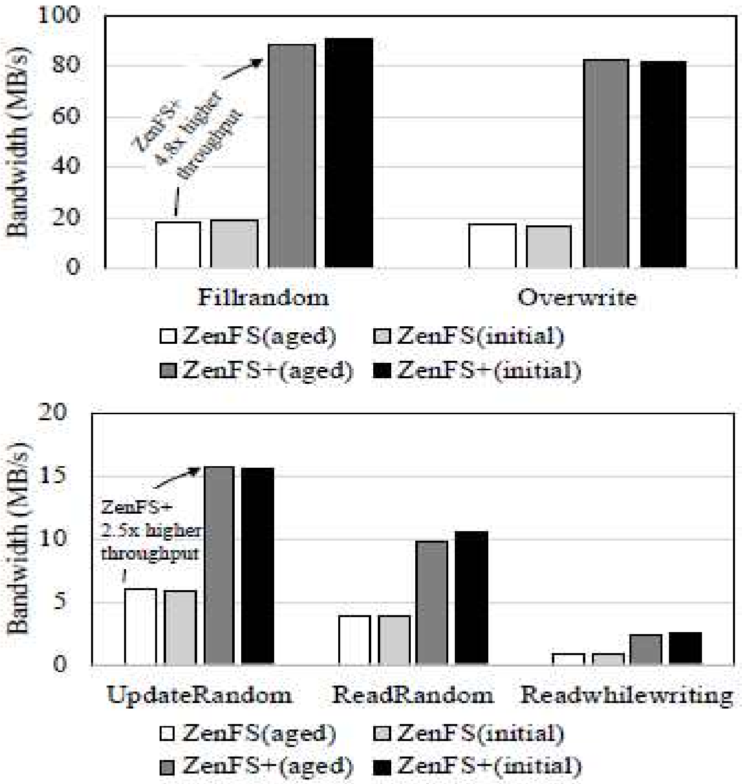
(Proposal)



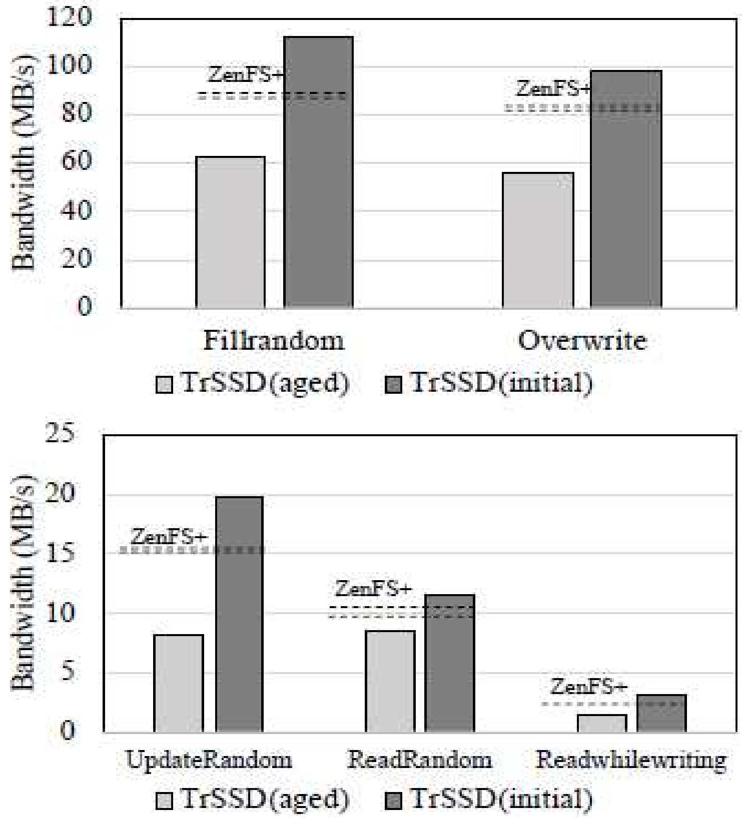
(Evaluation)

RocksDB on ZNS SSDs (7/9)

- Evaluation
  - ✓ Throughput
    - ZenFS+ vs ZenFS: better performance for diverse workloads
      - Less sensitive to aged/initial
    - ZenFS+ vs TrSSD: depend on BI tax



(ZenFS vs. ZenFS+)



(TrSSD vs. ZenFS+)

RocksDB on ZNS SSDs (8/9)

- Evaluation
  - ✓ YCSB results
    - Not only write-heavy (A, F), but also read-heavy (others)
  - ✓ Isolation capability
    - More predictable bandwidth

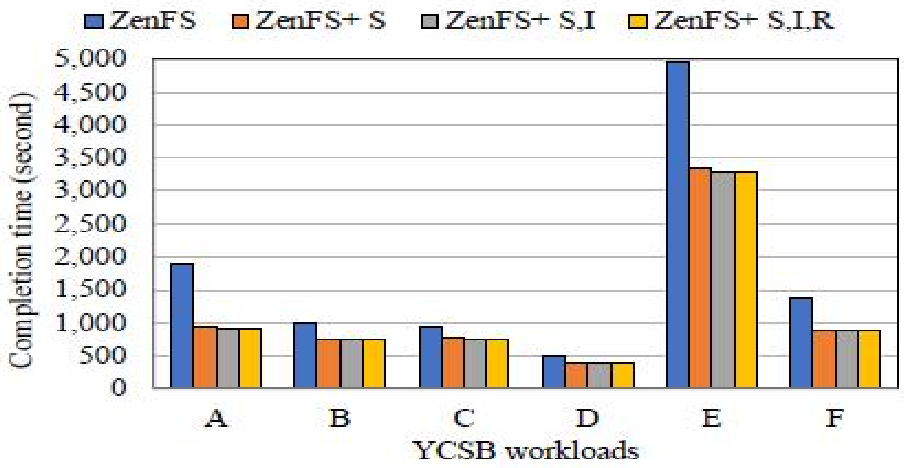
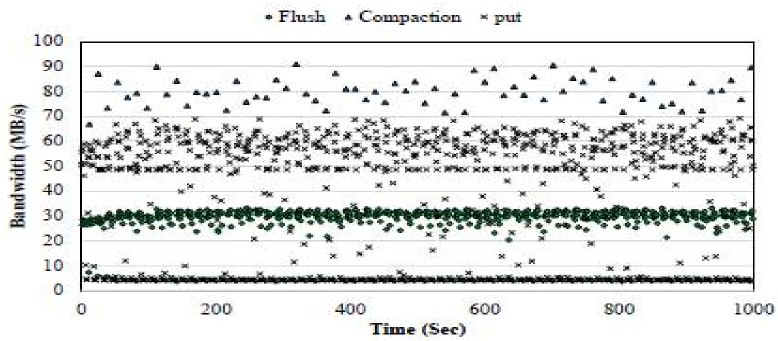


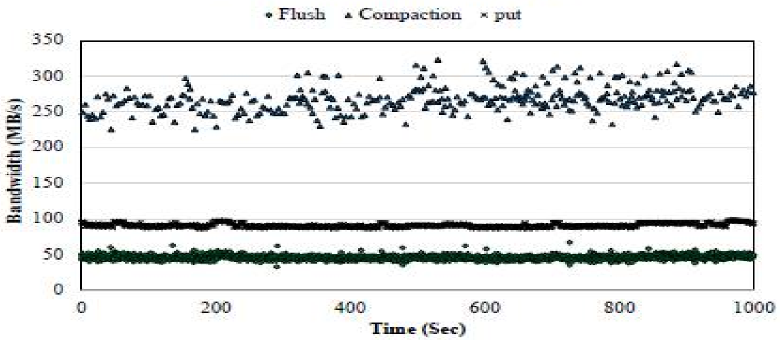
TABLE 2: YCSB workload

	A	B	C	D	E	F
r:w:u ratio	1:0:1	95:0:5	1:0:0	95:5:0	95:5:0	1:1:1
descrip-tion	Update-heavy	Read-mostly	Read-only	Read-latest	Short range query	Read-modify-write
Req. dist.	Zipfian	Zipfian	Zipfian	Latest	Zipfian	Zipfian

(YCSB results)



(a) Bandwidth usage with ZenFS

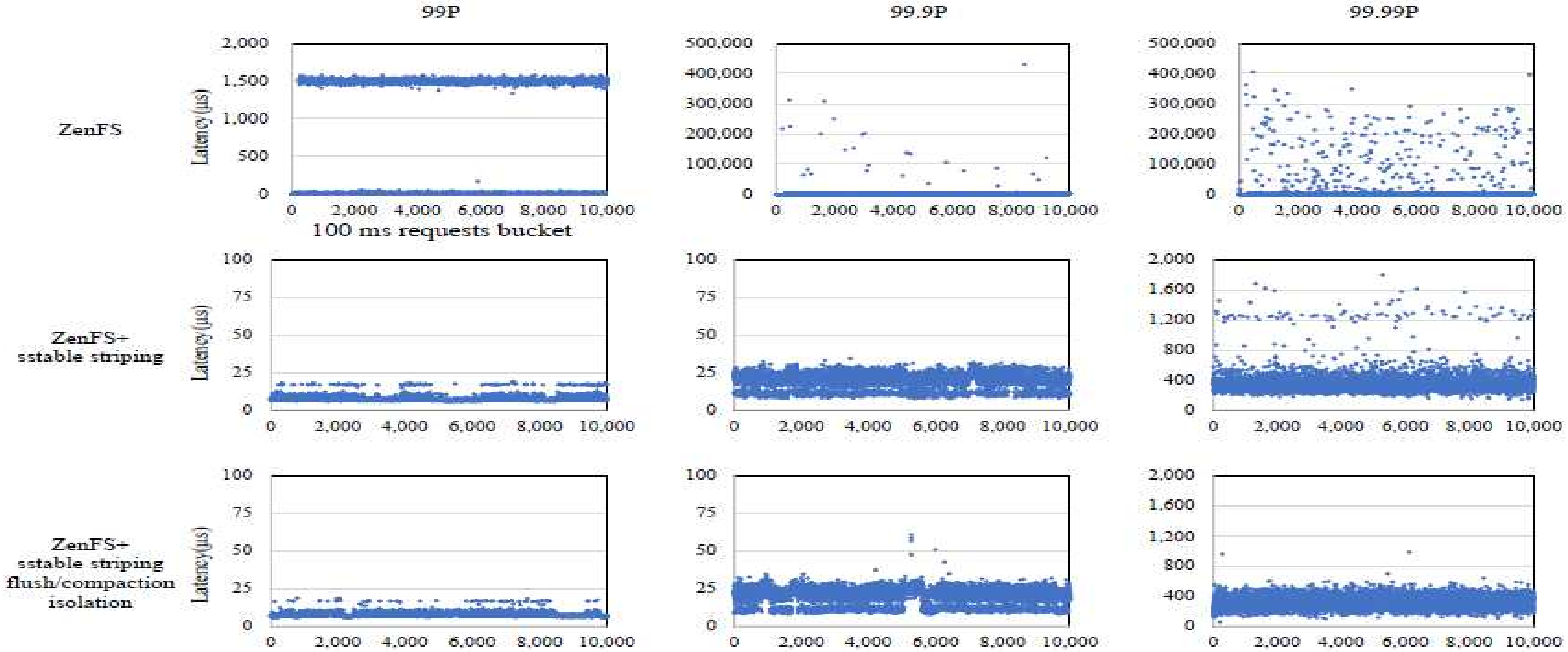


(b) Bandwidth usage with ZenFS+

(Isolation capability)

RocksDB on ZNS SSDs (9/9)

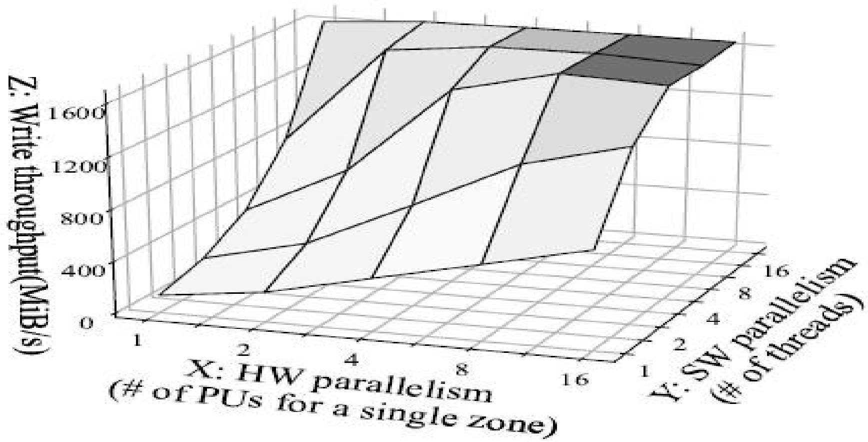
- Evaluation
  - ✓ Latency
    - ZenFS: latency spikes due to 1) utilize single zone and 2) compaction interfere flush
    - ZenFS+: 1) striping and 2) isolation ➔ can reduce latency spikes



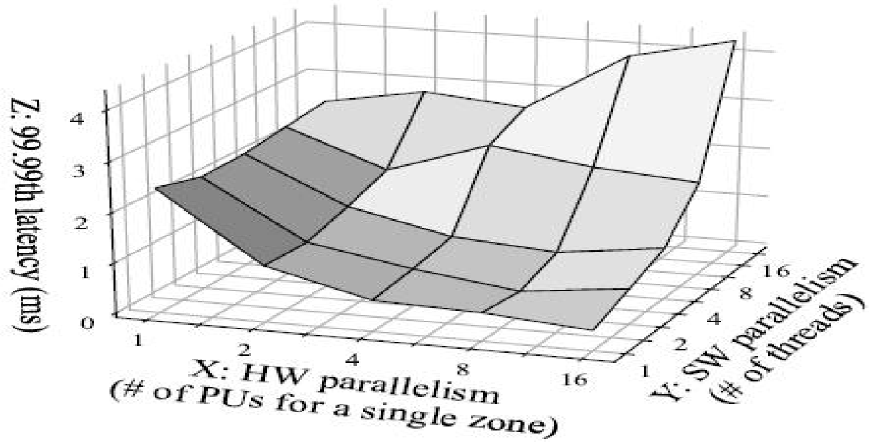
(Put latency under ZenFS and ZenFS+)

Discussion (1/3)

- ZNS SSDs
  - ✓ Expose a new consideration of parallelism
    - TrSSD: SSD-level parallelism vs OCSSD: Host-level parallelism
    - ZNS SSD: Both SSD-level parallelism and Host-level parallelism
      - Zone-to-PU mapping (HW-level) vs Thread-to-Zone mapping (SW-level)
  - ✓ Affect both performance and isolation (from ConfZNS)
    - TrSSD: utilize HW parallelism aggressively, less sensitive to SW parallelism at the cost of isolation
    - OCSSD: depend on SW parallelism too much
    - ZNS SSD: can provide a knob to exploit both (yet less flexible)



(a) ZNS SSD Performance

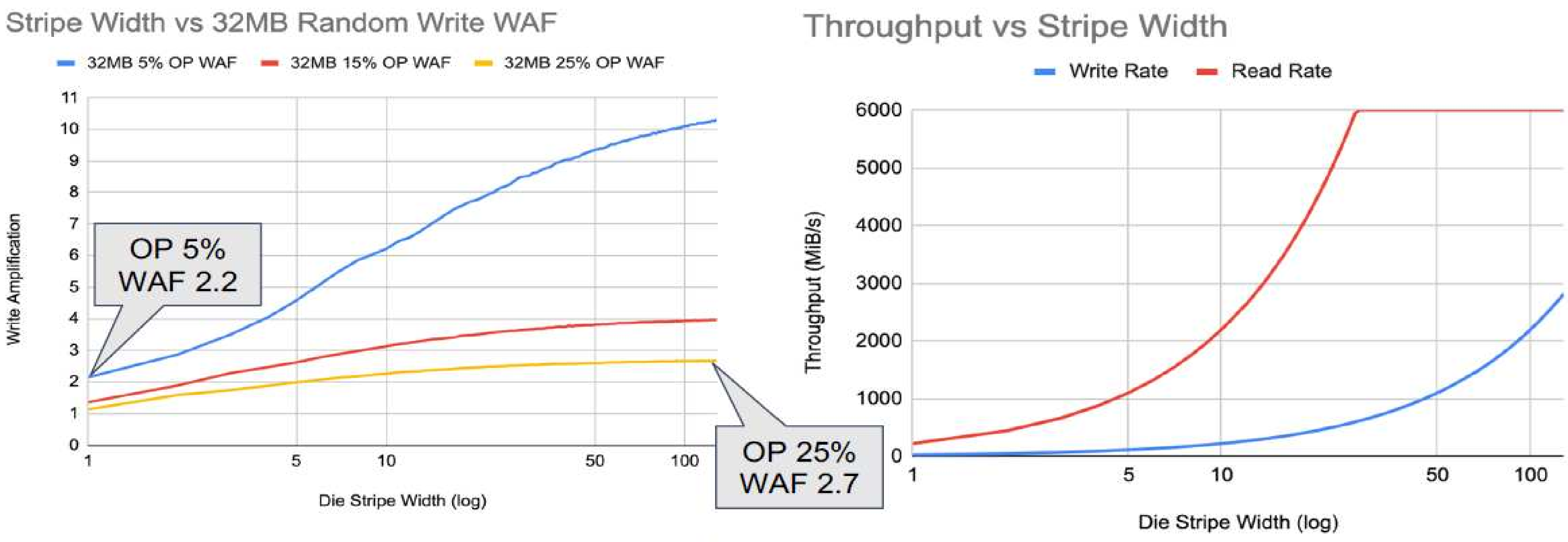


(b) ZNS SSD Isolation

Discussion (2/3)

- How about WAF?
  - ✓ SmartFTL

    - WAF reduction from 2.5 to 1.25 ➔ Reduce OP ➔ Save 18% Capex
  - ✓ Reconsider striping (parallelism)
    - Zhang et al., “Excessive parallelism considered harmful”, HotStorage’23



(Source: SmartFTL, OCP’21, https://www.youtube.com/watch?v=3O3zDrpt3uM)

Discussion (3/3)

- Related works
  - ✓ Min et al., "eZNS: An Elastic Zoned Namespace for Commodity ZNS SSDs", OSDI'23.
  - ✓ Kim et al., "RAIZN: Redundant Array of Independent Zoned Namespaces", ASPLOS'23.
  - ✓ Yeom et al., "zCeph: Achieving High Performance On Storage System Using Small Zoned ZNS SSD", ACM SAC'23.
  - ✓ Han et al., "Achieving Performance Isolation in Docker Environments with ZNS SSDs", IEEE NVMSA'23.
  - ✓ Bae et al., "What You Can't Forget: Exploiting Parallelism for Zoned Namespaces", HotStorage'22.
  - ✓ Lee et al., "Compaction-Aware Zone Allocation for LSM based Key-Value Store on ZNS SSDs", HotStorage'22.
  - ✓ Oh et al., "Accelerating RocksDB for Small-Zone ZNS SSDs by Parallel I/O Mechanism", ACM MIDDLEWARE'22.
  - ✓ Han et al., "ZNS+: Advanced Zoned Namespace Interface for Supporting In-Storage Zone Compaction", OSDI'21.
  - ✓ T. Stavrinos et al., "Don't Be a Blockhead: Zoned Namespaces Make Work on Conventional SSDs Obsolete", HotOS'21.
  - ✓ M. Bjørling et al., "ZNS: Avoiding the Block Interface Tax for Flash-based SSDs", ATC'21.
  - ✓ Song et al., "ConfZNS: A Novel Emulator for Exploring Design Space of ZNS SSDs", ACM Systor'23.
  - ✓ Oh et al., "ZenFS+: Nurturing Performance and Isolation to ZenFS" IEEE ACCESS'23.
  - ✓ …

Discussion