BIZA: Design of Self-Governing Block-Interface ZNS AFA for Endurance and Performance

Shushu Yi, Shaocong Sun, Li Peng, Yingbo Sun
Ming-Chang Yang, Zhichao Cao, Qiao Li,
Myoungsoo Jung, Ke Zhou, Jie Zhang

CHASELab Computer Hardware And System Evolution Laboratory
北京大學 PEKING UNIVERSITY
KAIST

Why We Need All-Flash Array (AFA)?

DELL EMC VMAX

PureStorage FlashArray

AFA → Superb **performance** & **reliability**!

Datacenters

Supercomputers

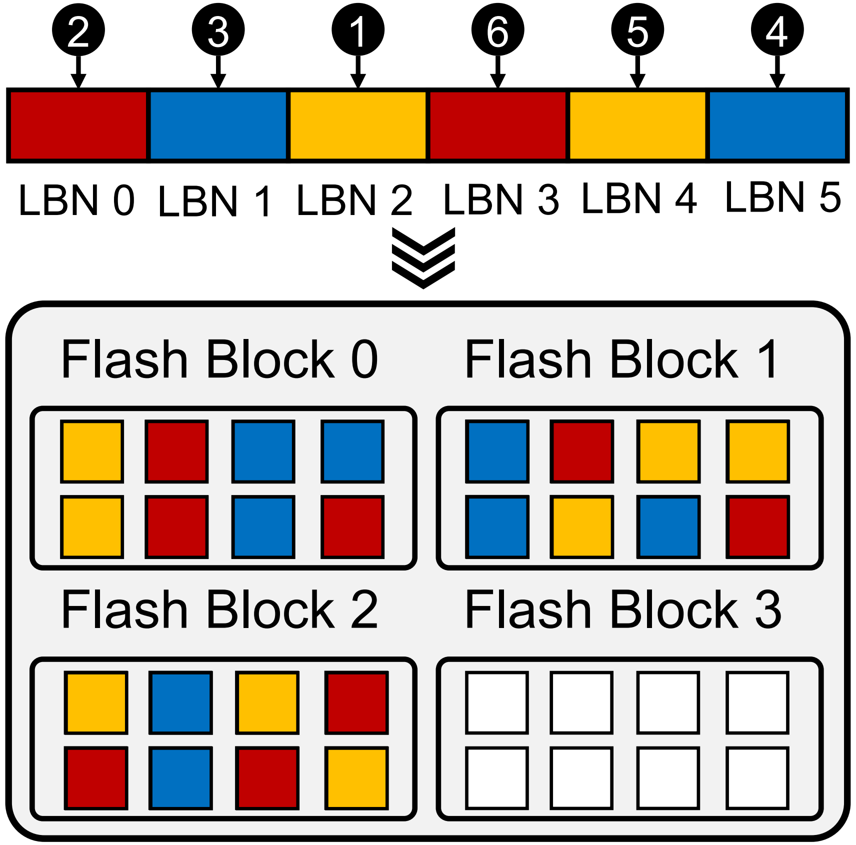
WD OPENFLEX

NetApp AFF

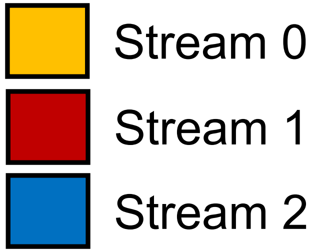
All-flash array are widely adopted in diverse domains.

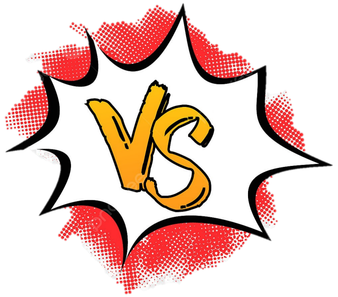
Open Question: I/O Interface Choices

Block Interface

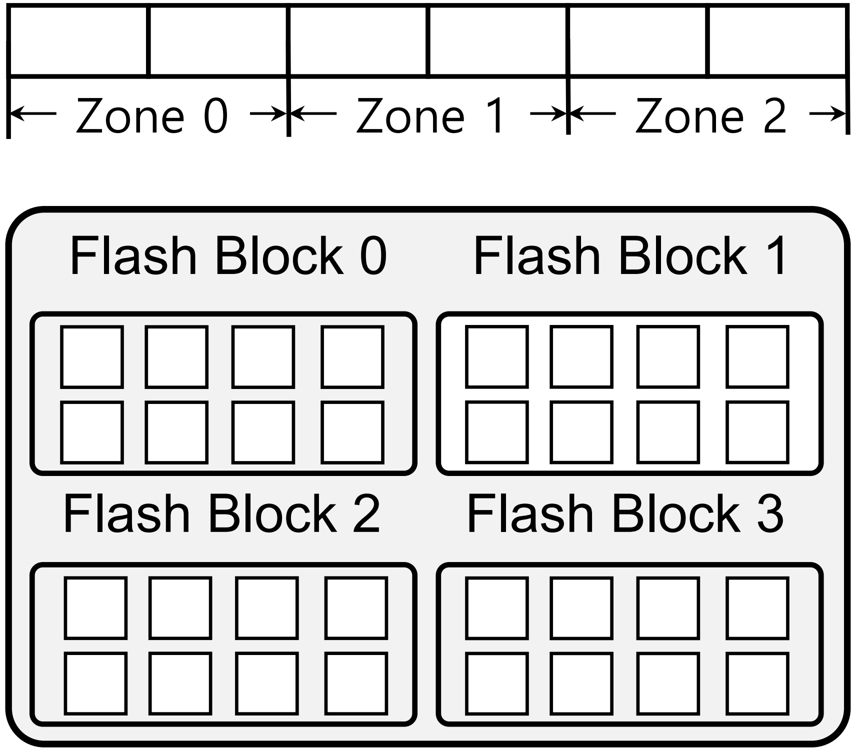


Conventional SSD



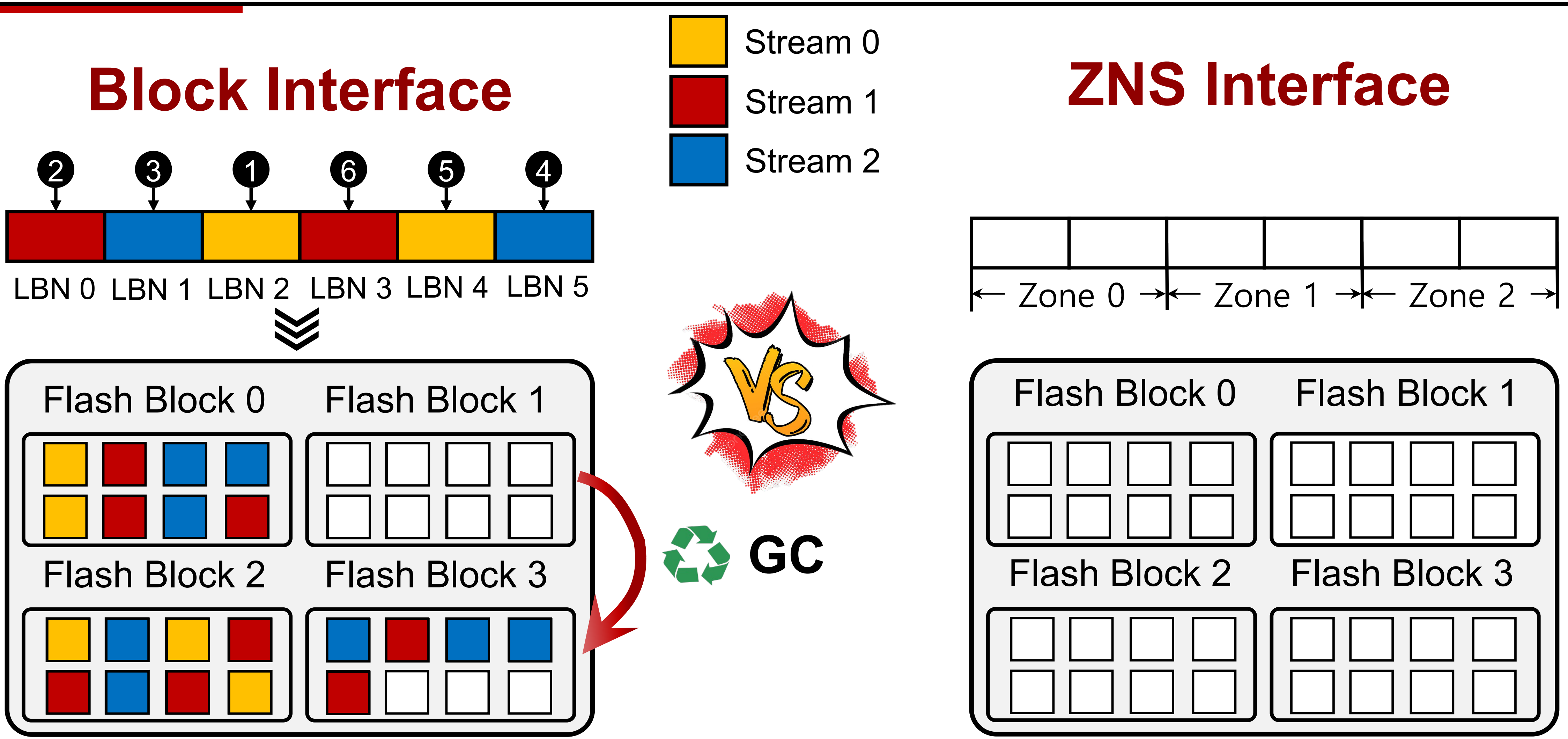


ZNS Interface



Zoned Namespace (ZNS) SSD

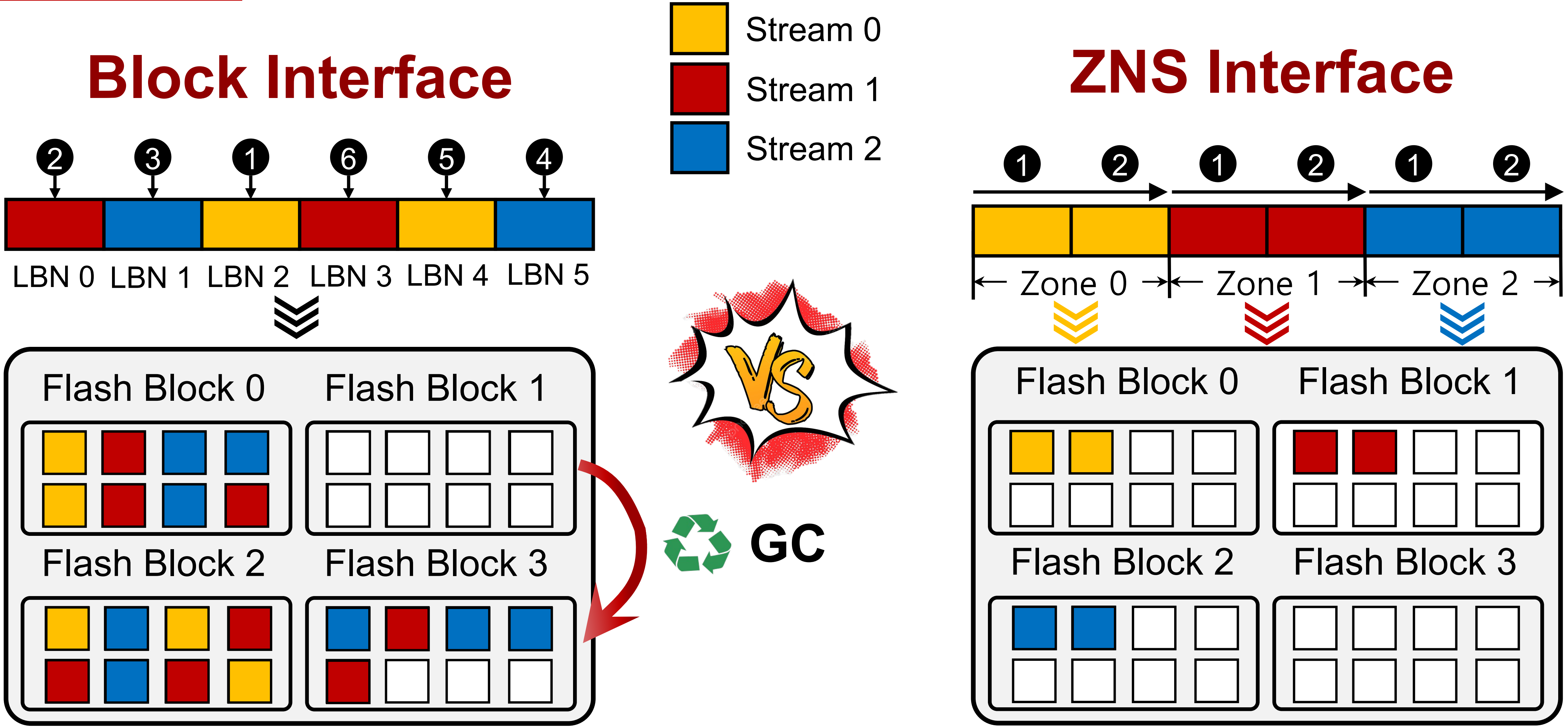
Open Question: I/O Interface Choices



Conventional SSD

Zoned Namespace (ZNS) SSD

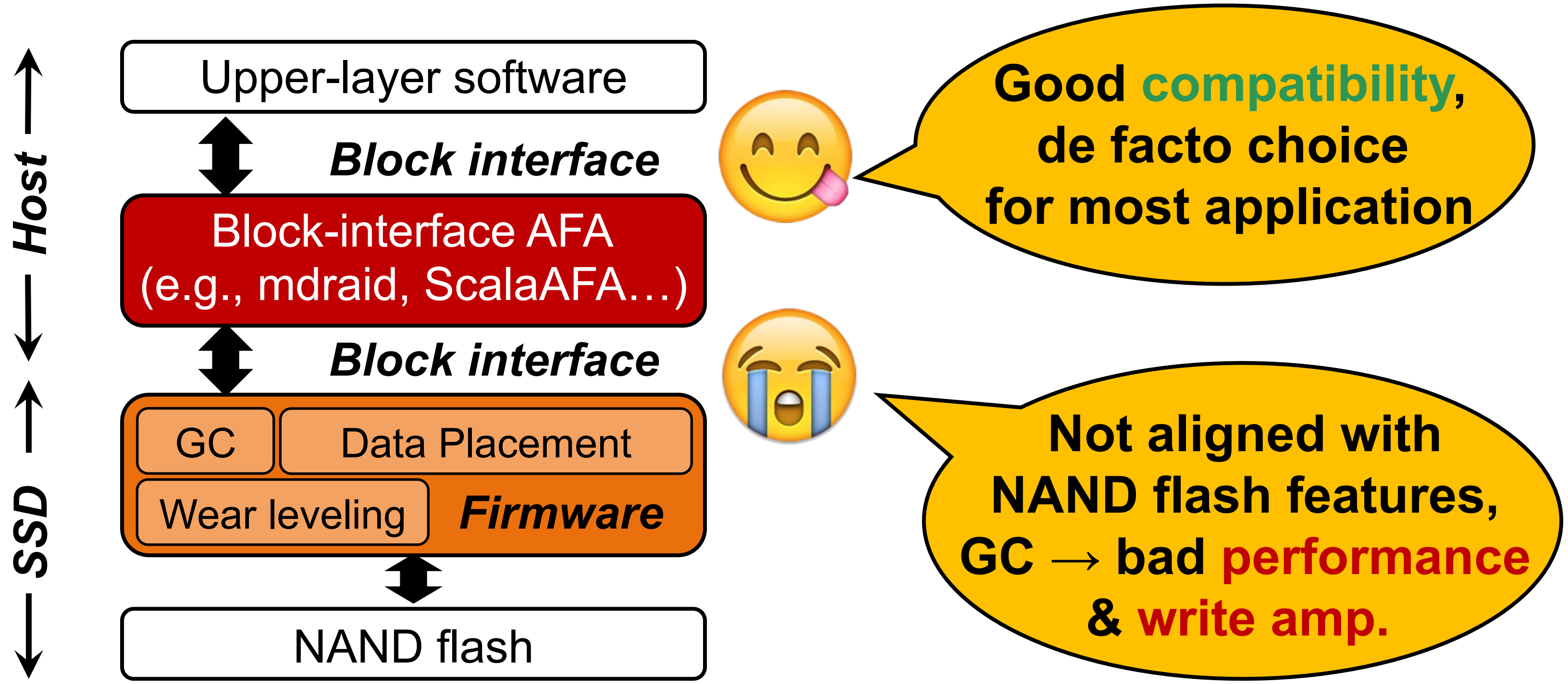
Open Question: I/O Interface Choices



Conventional SSD | **Zoned Namespace (ZNS) SSD**

Existing Designs: Rob Peter to Pay Paul

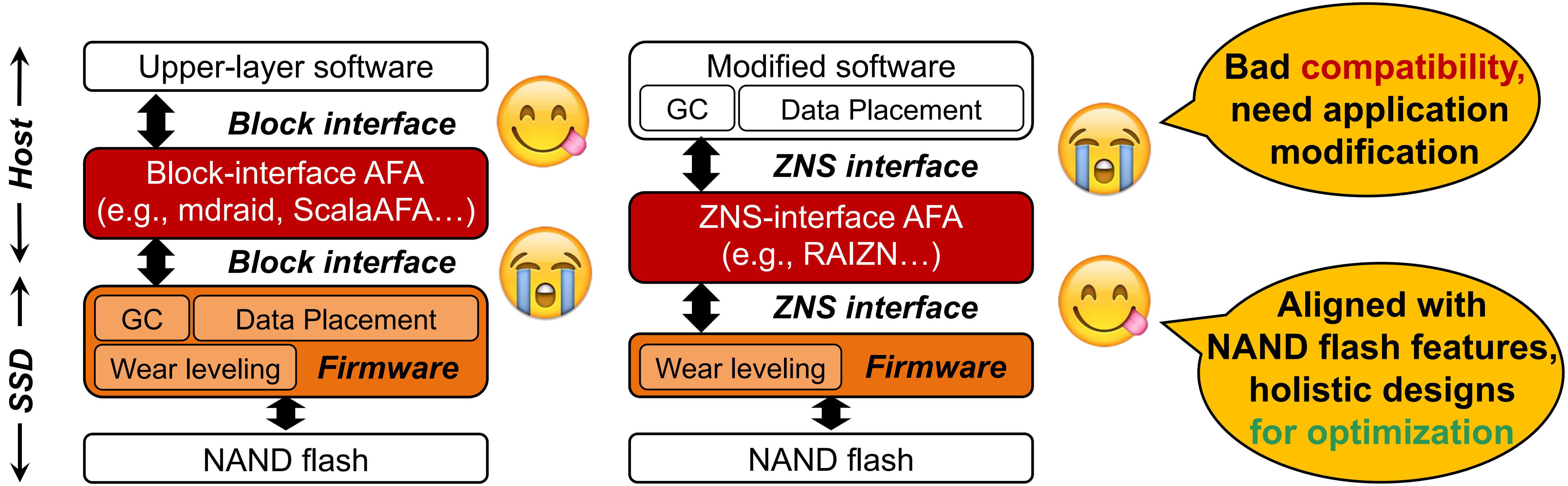
- **Block-interface AFA:** mdraid (Linux default)[1], ScalaAFA (ATC'24)[2]
  - Random writes → good compatibility, SSD-internal tasks → bad performance & write amplification



[1] Linux. mdraid layer. h!ps://github.com/torvalds/linux/tree/master/drivers/md.
[2] ScalaAFA: Constructing User-Space All-Flash Array Engine with Holistic Designs (ATC'24)

Existing Designs: Rob Peter to Pay Paul

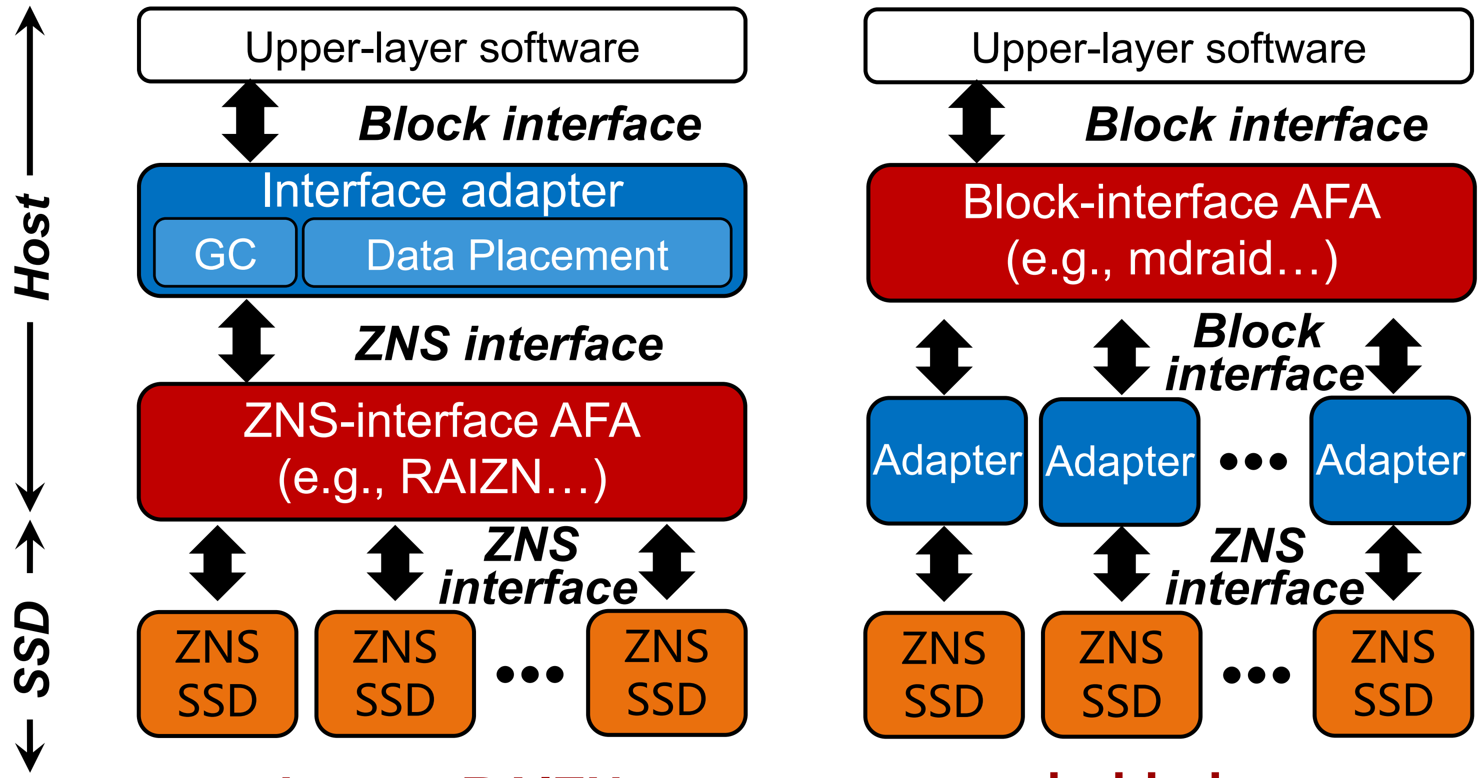
- **Block-interface AFA:** mdraid (Linux default), ScalaAFA (ATC'24)
  - Random writes → good compatibility, SSD-internal tasks → bad performance & write amplification
- **ZNS-interface AFA:** RAIZN (ASPLOS'23)[1]
  - host-managed tasks → holistic designs for optimization, Sequential writes → bad compatibility



[1] RAIZN: Redundant Array of Independent Zoned Namespaces (ASPLOS'23)

Interface Adapter: A Simple Solution

- **Interface adapter**: dm-zoned (Linux), dm-zap (Western Digital)
  - Converting ZNS interface to block interface
  - Maintaining mappings from block numbers to zone-related addresses



dmzap+RAIZN **mdraid+dmzap**

Bridged semantic gap, but failed to harvest the benefits of ZNS

Quantitative Analysis

- **Experimental setup:** 3+1 (RAID 5) Western Digital ZN540 4TB SSDs
  - Peak throughput: 3265 MB/s and 2170 MB/s for read and write

WD ZN540 SSD

Challenge 1: Short endurance

- dmzap+RAIZN and mdraid+dmzap generate 33.3% and 54.6% more flash writes
- dm-zap muddles data with dfferent lifetimes in the same zones

Challenge 2: Low throughput

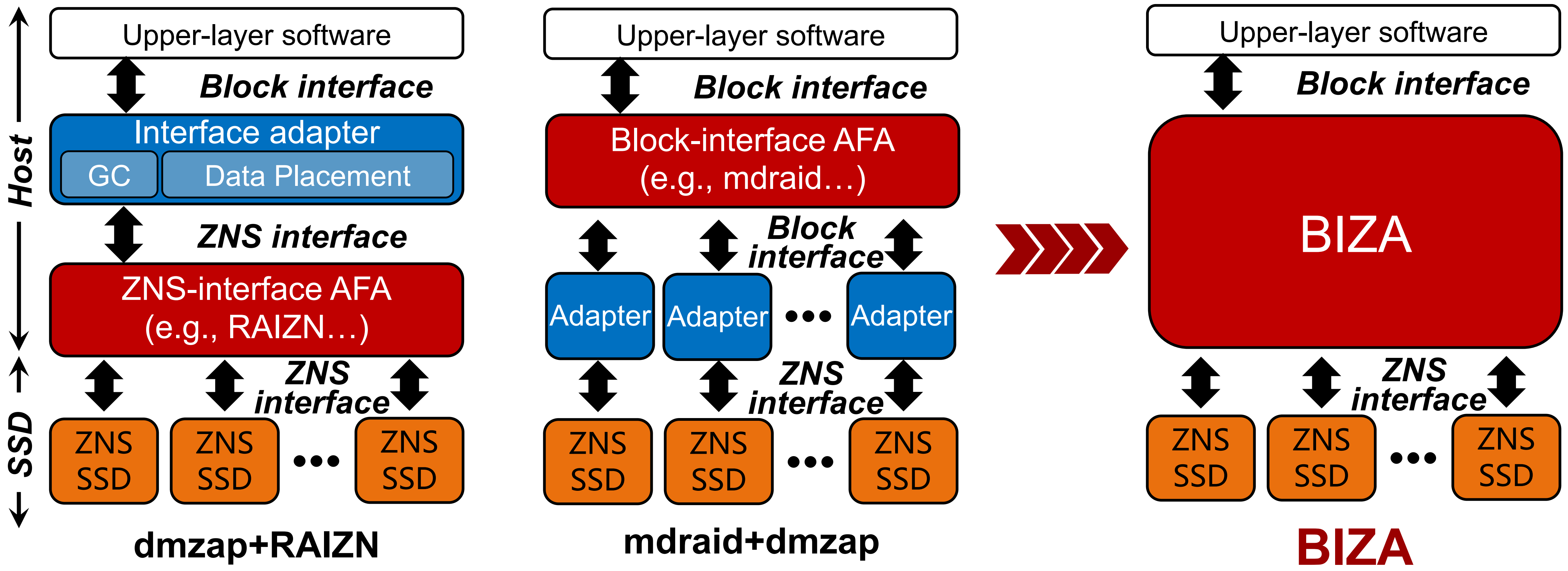
- dmzap+RAIZN and mdraid+dmzap only achieve 47.7% and 18.4% of the ideal throughput
- dm-zap submits writes serially and only allows one in-flight write per zone
- RAIZN relies on a centralized zone to buffer the frequently updated metadata

Challenge 3: High tail latency

- dmzap+RAIZN and mdraid+dmzap increase 99.99^{th} tail latency by 10.3x and 2.2x after GC starts
- dm-zap handle GC and user I/O with the same I/O resources

Please refer to our paper for details.

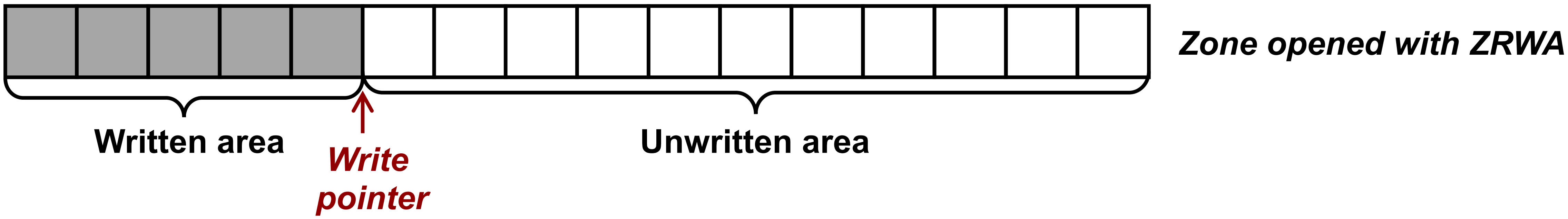
Our Solution: BIZA



Key insight: new features & internal parallelism

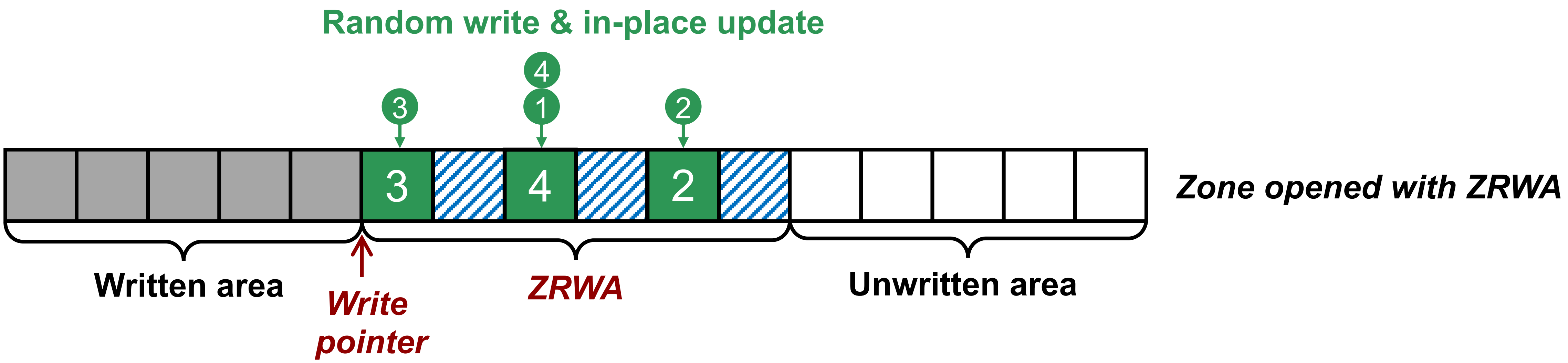
ZNS SSD Exploration: Zone Random Write Area

- NVMe Technical Proposal 4076b: Zone Random Write Area (ZRWA)



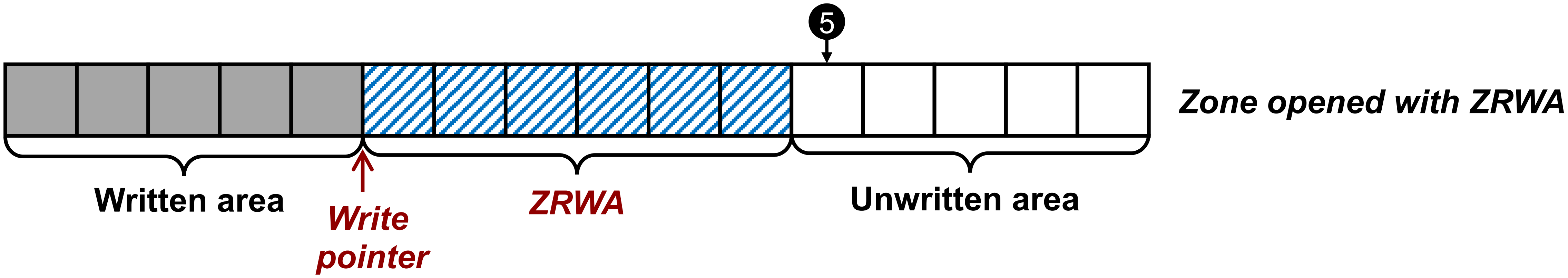
ZNS SSD Exploration: Zone Random Write Area

- **NVMe Technical Proposal 4076b: Zone Random Write Area (ZRWA)**
  - Break the strict sequential write constraint
  - An efficient abstraction of the write buffer within SSDs
  - Battery-backed DRAM, NVM, or FTL-mapped high-endurance flash (e.g., SLC)



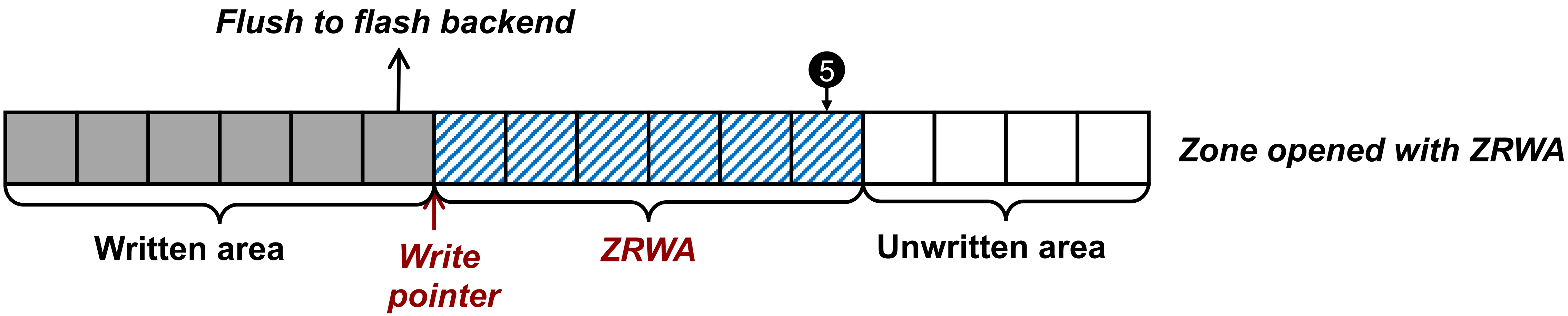
ZNS SSD Exploration: Zone Random Write Area

- **NVMe Technical Proposal 4076b: Zone Random Write Area (ZRWA)**
  - Break the strict sequential write constraint
  - An efficient abstraction of the write buffer within SSDs
  - Battery-backed DRAM, NVM, or FTL-mapped high-endurance flash (e.g., SLC)



ZNS SSD Exploration: Zone Random Write Area

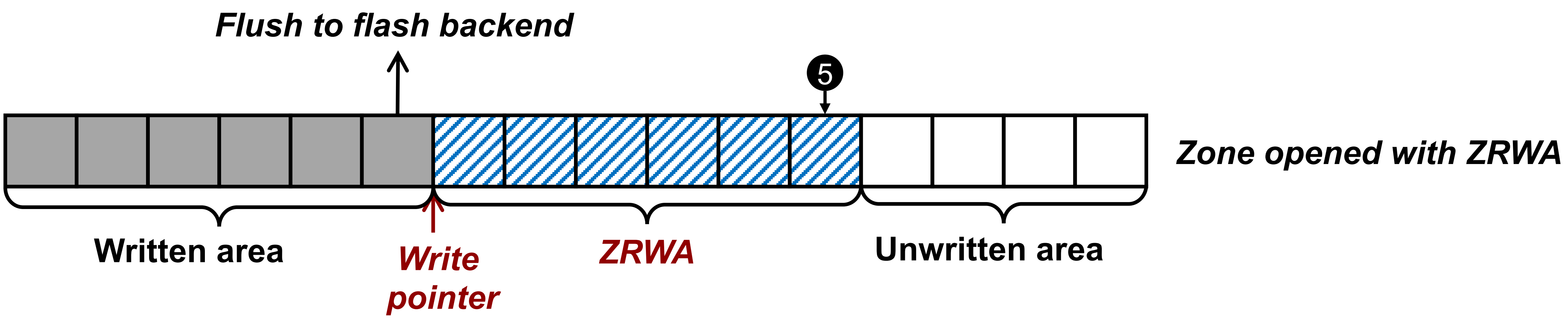
- **NVMe Technical Proposal 4076b: Zone Random Write Area (ZRWA)**
  - Break the strict sequential write constraint
  - An efficient abstraction of the write buffer within SSDs
  - Battery-backed DRAM, NVM, or FTL-mapped high-endurance flash (e.g., SLC)



Idea: absorb the frequently updated data within ZRWA

ZNS SSD Exploration: Zone Random Write Area

- **NVMe Technical Proposal 4076b: Zone Random Write Area (ZRWA)**
  - Break the strict sequential write constraint
  - An efficient abstraction of the write buffer within SSDs
  - Battery-backed DRAM, NVM, or FTL-mapped high-endurance flash (e.g., SLC)



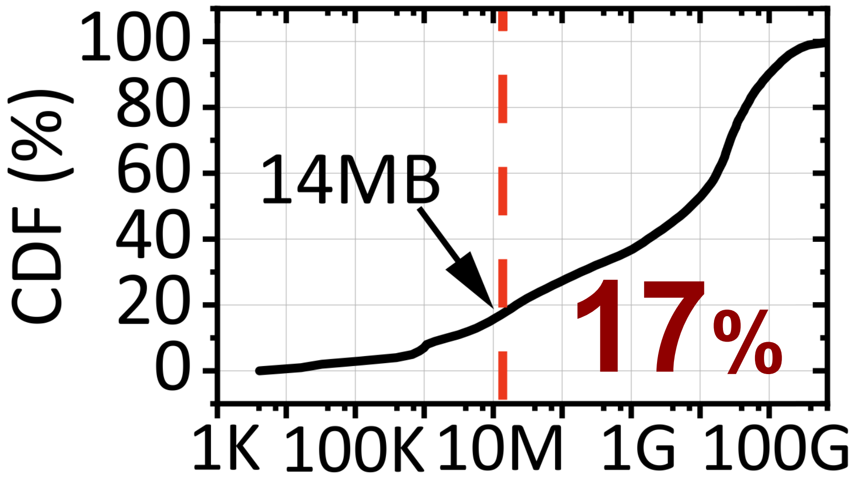
Idea: absorb the frequently updated data within ZRWA

ZNS SSD Exploration: Zone Random Write Area

Idea: absorb the frequently updated data within ZRWA

However, the size of ZRWA is very limited per ZNS SSD

ZNS SSD	Zone Capacity	ZRWA size per open zone	Total ZRWA size
WD ZN540	1077 MB	1 MB	14 MB
DapuStor J5500Z	18144 MB	1 MB	16 MB
Inpur NS8600G	96 MB	1440 KB	11.25 MB



Reuse distance of SYSTOR [1]

[1] Understanding storage traffic characteristics on enterprise virtual desktop infrastructure. (SYSTOR'17)

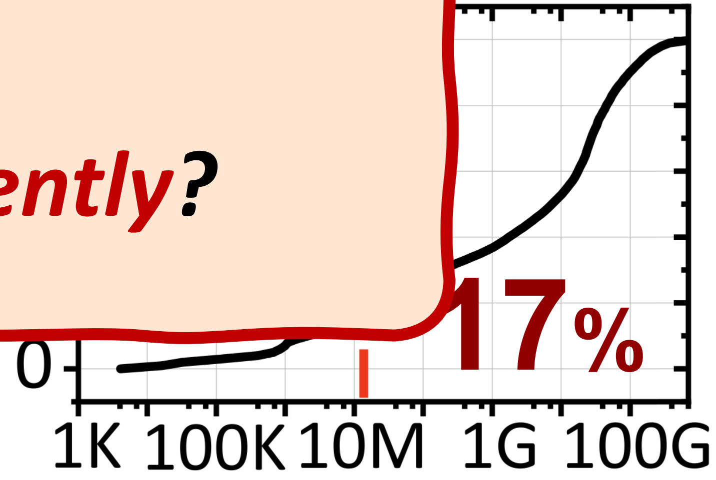
ZNS SSD Exploration: Zone Random Write Area

Idea: absorb the frequently updated data within ZRWA

However, the size of ZRWA is very limited per ZNS SSD

How to use the limited size of ZRWA efficiently?

DapuStor J5500Z	18144 MB	1 MB	16 MB
Inpur NS8600G	96 MB	1440 KB	11.25 MB

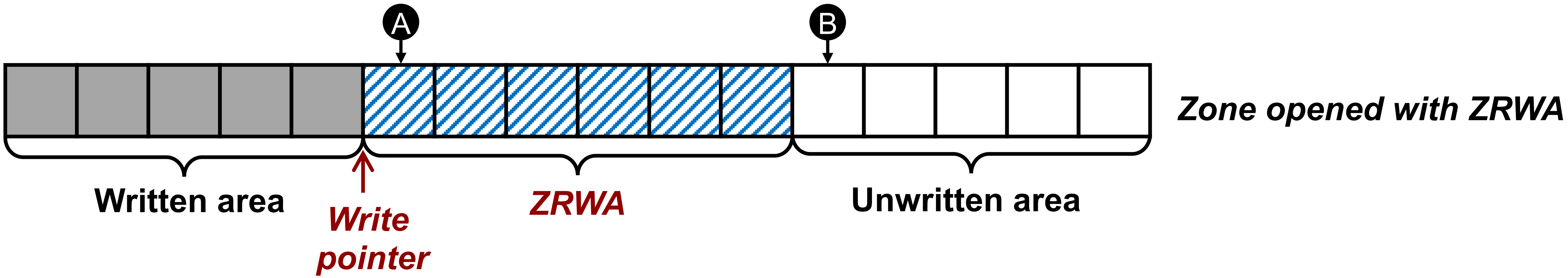


Reuse distance of SYSTOR [1]

[1] Understanding storage traffic characteristics on enterprise virtual desktop infrastructure. (SYSTOR'17)

ZNS SSD Exploration: Intra-Zone Parallelism

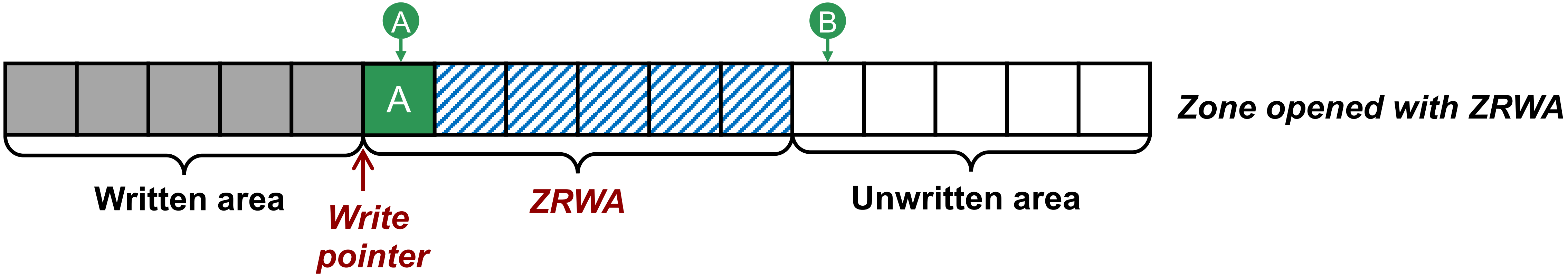
- **I/O stack has no guarantee on the orders of request submissions**
  - Reorder / merge I/O for high performance [1]
- **Implicit I/O reorders may cause failures for parallel writes**



[1] Linux Block IO: Introducing Multi-queue SSD Access on Multi-core Systems (SYSTOR'13)

ZNS SSD Exploration: Intra-Zone Parallelism

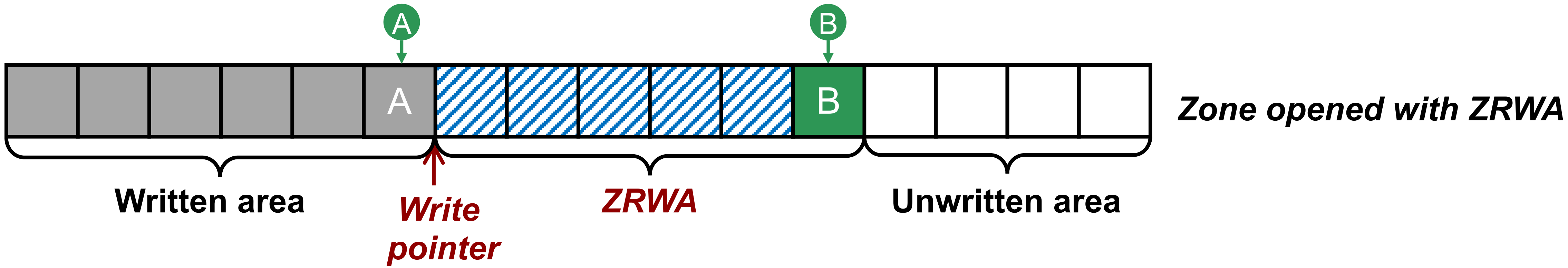
- **I/O stack has no guarantee on the orders of request submissions**
  - Reorder / merge I/O for high performance [1]
- **Implicit I/O reorders may cause failures for parallel writes**



[1] Linux Block IO: Introducing Multi-queue SSD Access on Multi-core Systems (SYSTOR'13)

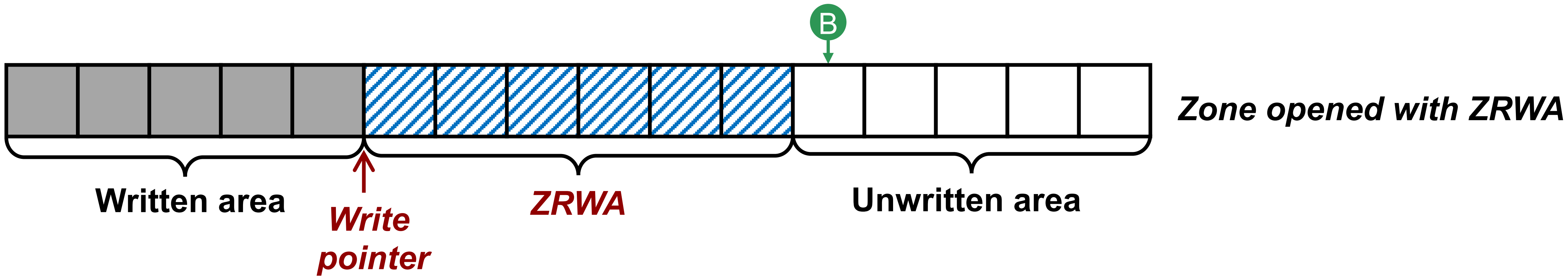
ZNS SSD Exploration: Intra-Zone Parallelism

- **I/O stack has no guarantee on the orders of request submissions**
  - Reorder / merge I/O for high performance [1]
- **Implicit I/O reorders may cause failures for parallel writes**



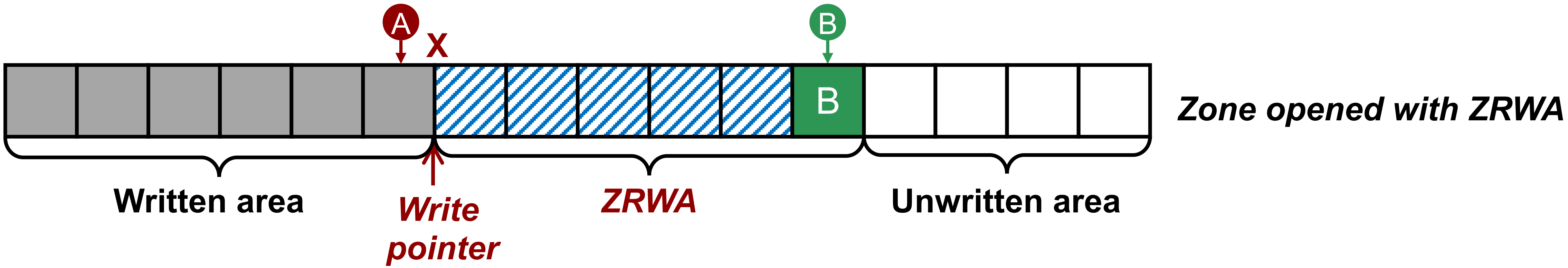
ZNS SSD Exploration: Intra-Zone Parallelism

- **I/O stack has no guarantee on the orders of request submissions**
  - Reorder / merge I/O for high performance [1]
- **Implicit I/O reorders may cause failures for parallel writes**



ZNS SSD Exploration: Intra-Zone Parallelism

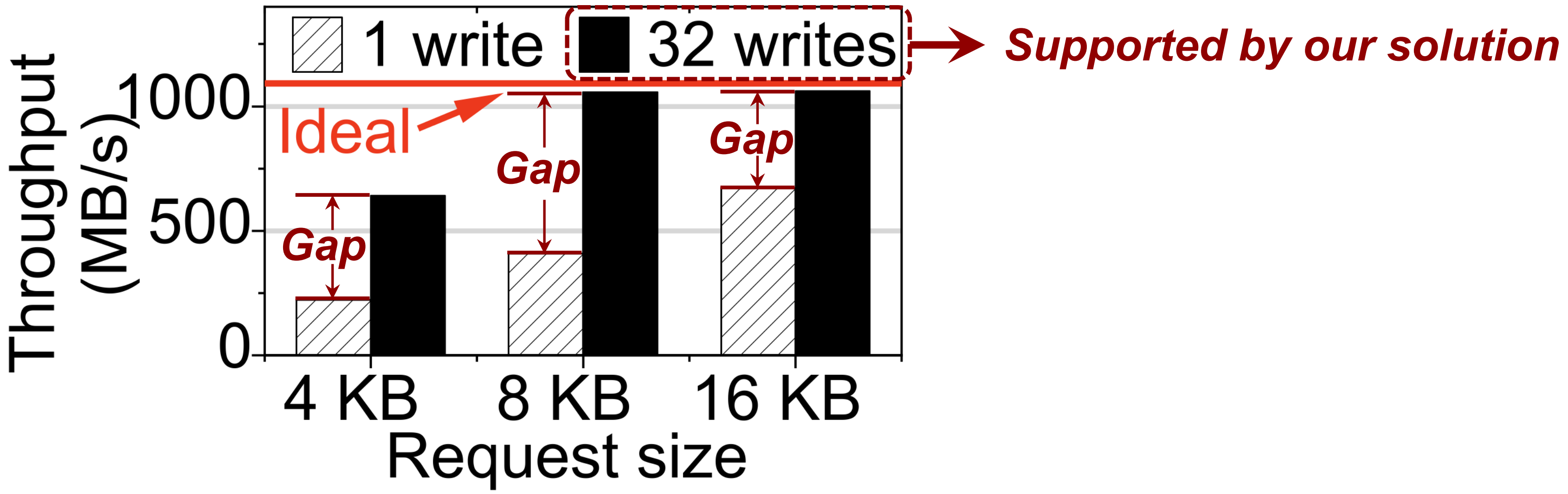
- **I/O stack has no guarantee on the orders of request submissions**
  - Reorder / merge I/O for high performance [1]
- **Implicit I/O reorders may cause failures for parallel writes**



[1] Linux Block IO: Introducing Multi-queue SSD Access on Multi-core Systems (SYSTOR'13)

ZNS SSD Exploration: Intra-Zone Parallelism

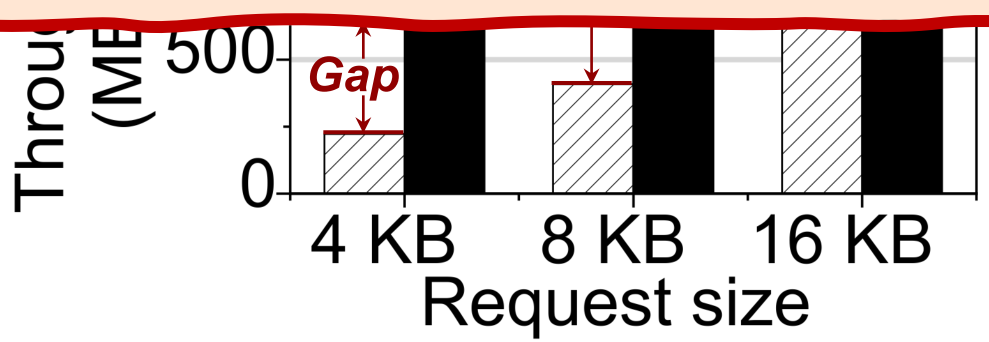
- **I/O stack has no guarantee on the orders of request submissions**
  - Reorder / merge I/O for high performance [1]
- **Implicit I/O reorders may cause failures for parallel writes**
- **Only allow one-inflight write → loses up to 65.3% throughput**



ZNS SSD Exploration: Intra-Zone Parallelism

- **I/O stack has no guarantee on the orders of request submissions**
  - Reorder / merge I/O for high performance [1]
- **Implicit I/O reorders may cause failures for parallel writes**
- **Or**

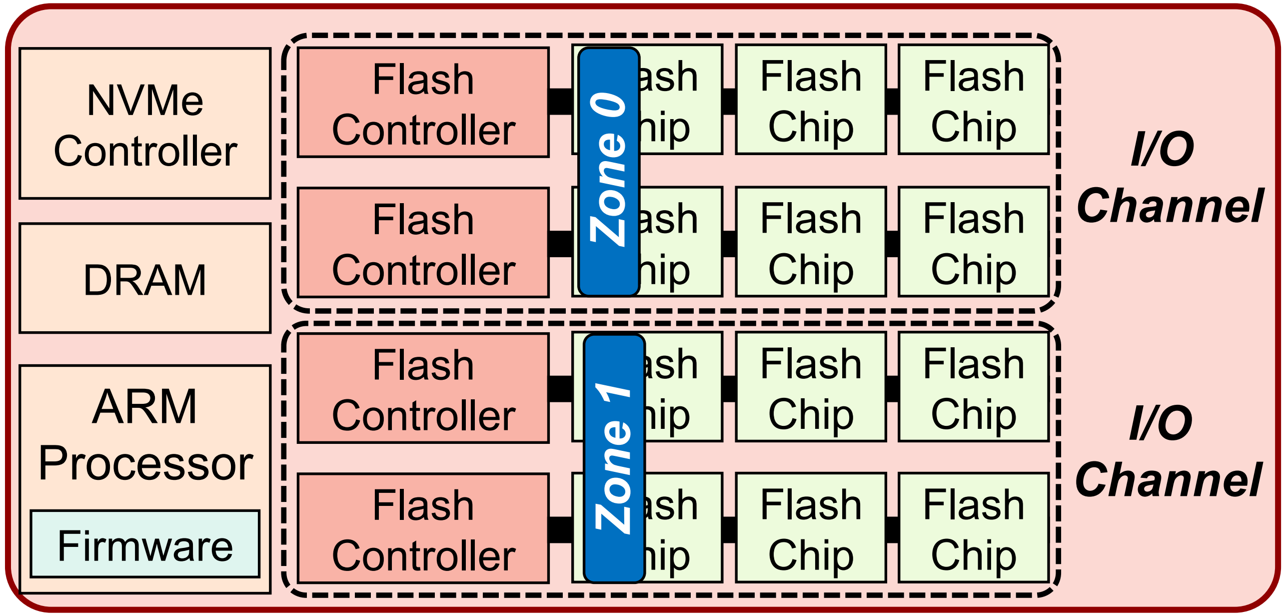
How to enjoy ZRWA while maximizing intra-zone parallelism?



[1] Linux Block IO: Introducing Multi-queue SSD Access on Multi-core Systems (SYSTOR'13)

ZNS SSD Exploration: Inter-Zone Parallelism

- **ZNS SSDs have multiple parallel I/O resources**
  - We call them I/O channels (may consist of multiple flash channels or chips)
  - Zones from with separated I/O channels can handle I/O requests in parallel



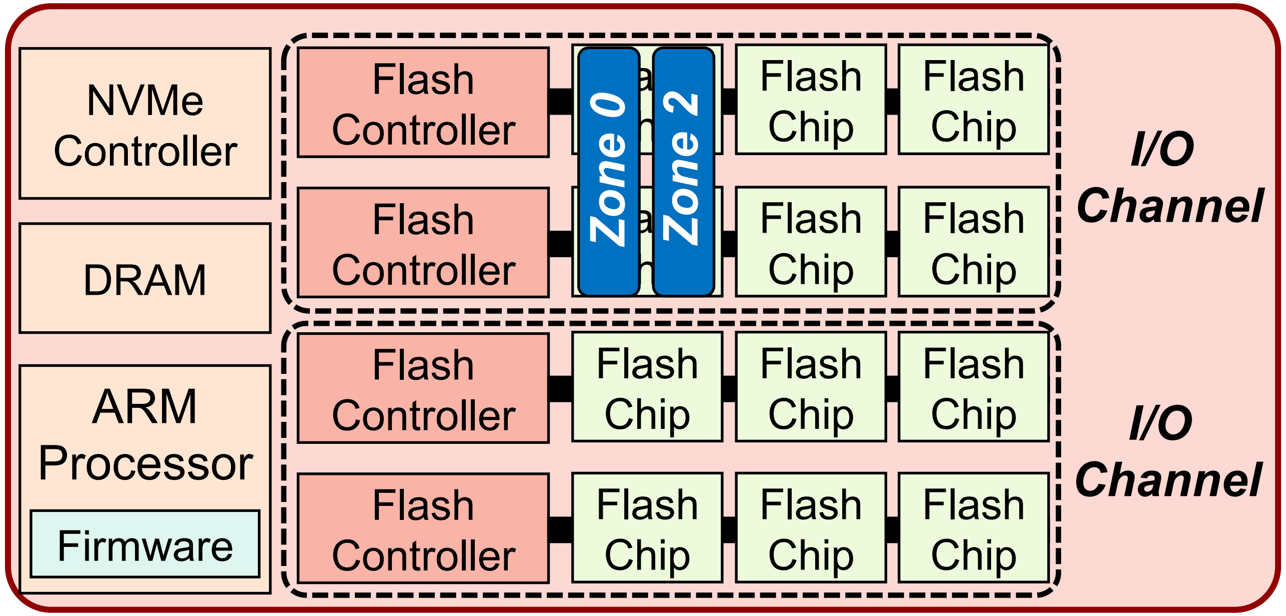
SSD Architecture

Scenario	Bandwidth (MB/s)	Avg. lat. (us)	99.99th lat. (us)
Single zone	1092	59	570
Two zones in **diverse** I/O channels	2170	59	685

Write Tests

ZNS SSD Exploration: Inter-Zone Parallelism

- **ZNS SSDs have multiple parallel I/O resources**
  - We call them I/O channels (may consist of multiple flash channels or chips)
  - Zones from with separated I/O channels can handle I/O requests in parallel
- **Give up the centralized metadata zone**
- **Schedule user I/O and GC with parallel zones**



SSD Architecture

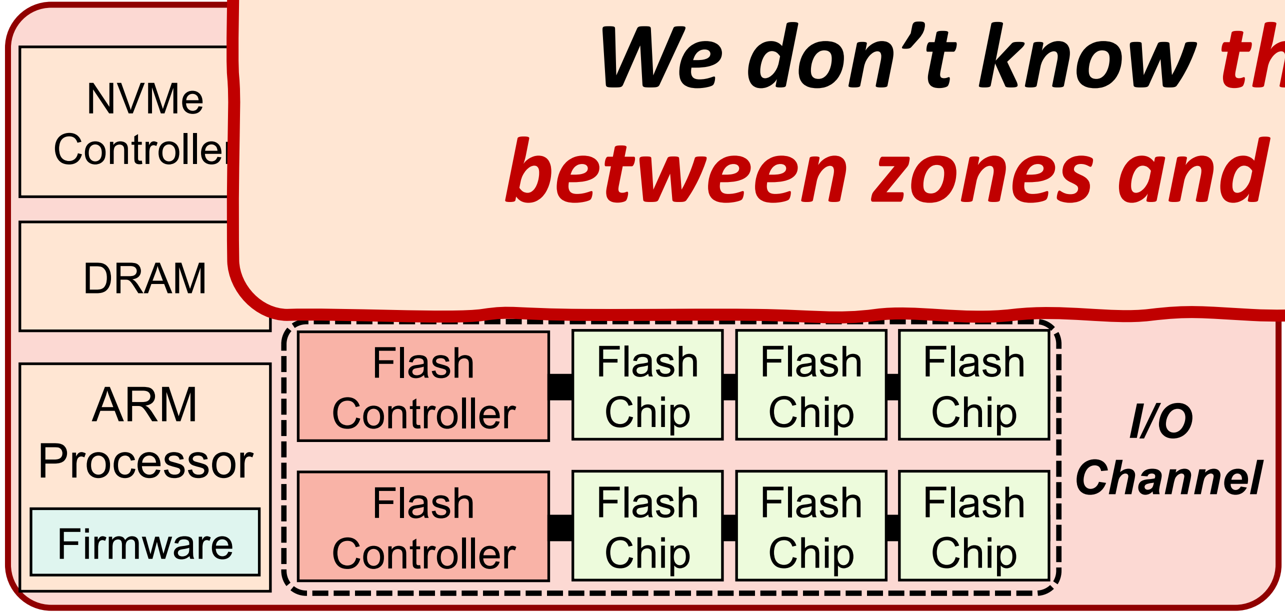
Scenario	Bandwidth (MB/s)	Avg. lat. (us)	99.99th lat. (us)
Single zone	1092	59	570
Two zones in **diverse** I/O channels	2170	59	685
Two zones in **Identical** I/O channel	1092	118	2310

Write Tests

ZNS SSD Exploration: Inter-Zone Parallelism

- **ZNS SSDs have multiple parallel I/O resources**
  - We call them I/O channels (may consist of multiple flash channels or chips)
  - Zones from with separated I/O channels can handle I/O requests in parallel
- **Give up the centralized metadata zone**
- **Sch**

We don't know the mappings between zones and I/O resources!



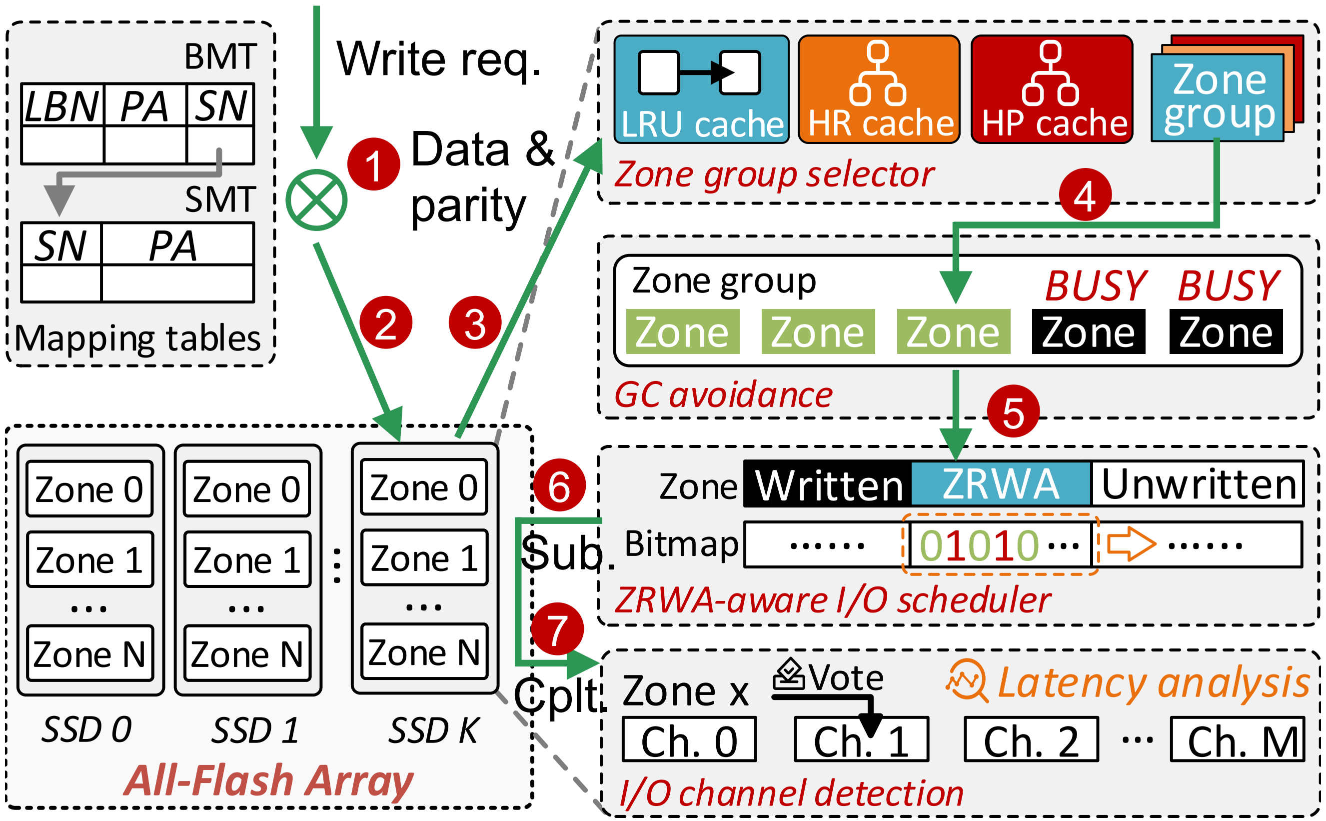
SSD Architecture

[illegible]	[illegible]	. lat. (us)	99.99th lat. (us)
[illegible]	[illegible]	9	570
Two zones in diverse I/O channels	2170	59	685
Two zones in Identical I/O channel	1092	118	2310

Write Tests

BIZA Overview

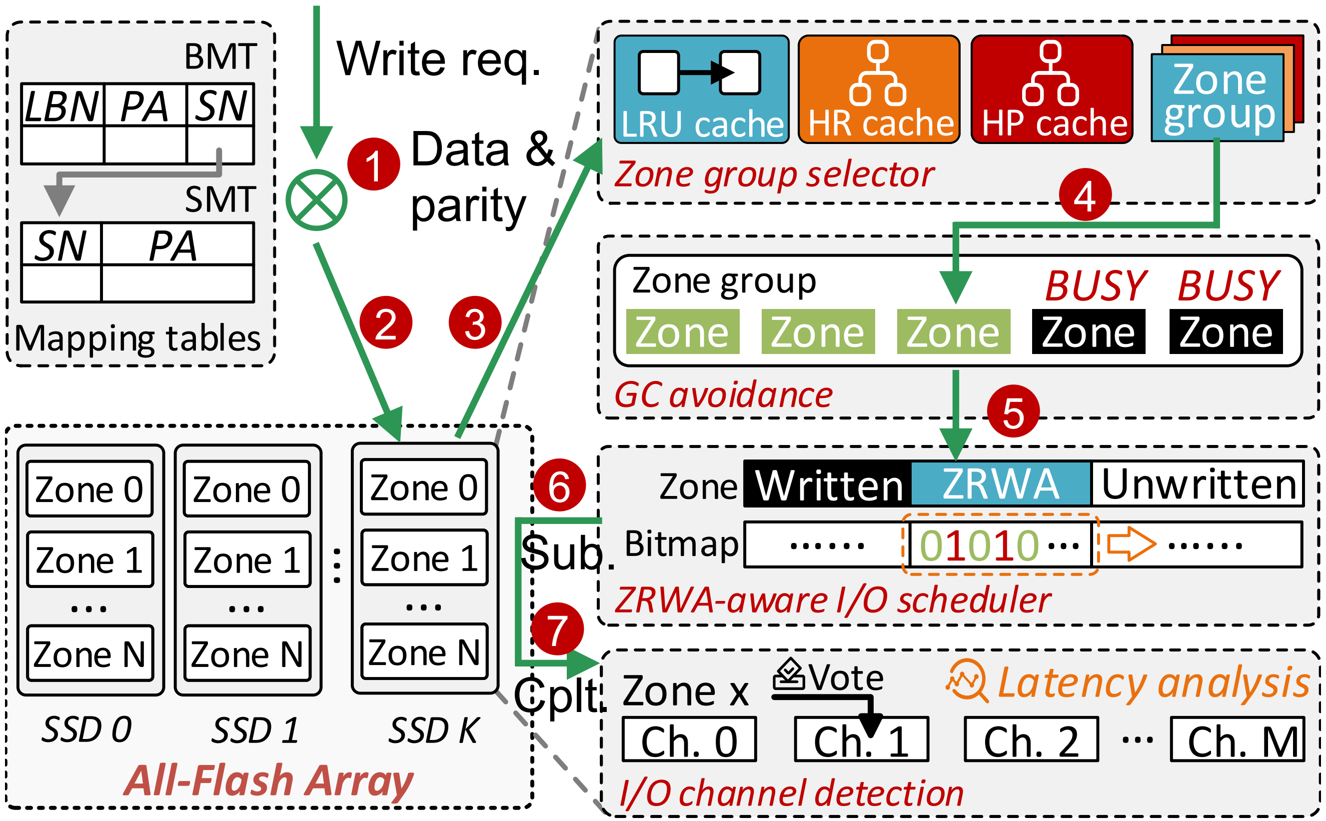
- **Zone group selector**
  - Isolate hot chunks from cold ones
  - For better utilization of the scarce ZRWA
  - A set of zones mapped with diff. channels
- **GC avoidance mechanism**
  - Schedule I/O and GC with parallel zones
  - Exhaust inter-zone parallelism
- **ZRWA-aware I/O scheduler**
  - Dispatches I/O requests in parallel
  - Exhaust intra-zone parallelism
- **I/O channel detection**
  - Collect I/O completion latency
  - Confirm the mappings between channels & zones



Overview of BIZA

BIZA Overview

- **Zone group selector**
  - Isolate hot chunks from cold ones
  - For better utilization of the scarce ZRWA
  - A set of zones mapped with diff. channels
- **GC avoidance mechanism***
  - Schedule I/O and GC with parallel zones
  - Exhaust inter-zone parallelism
- **ZRWA-aware I/O scheduler***
  - Dispatches I/O requests in parallel
  - Exhaust intra-zone parallelism
- **I/O channel detection***
  - Collect I/O completion latency
  - Confirm the mappings between channels & zones

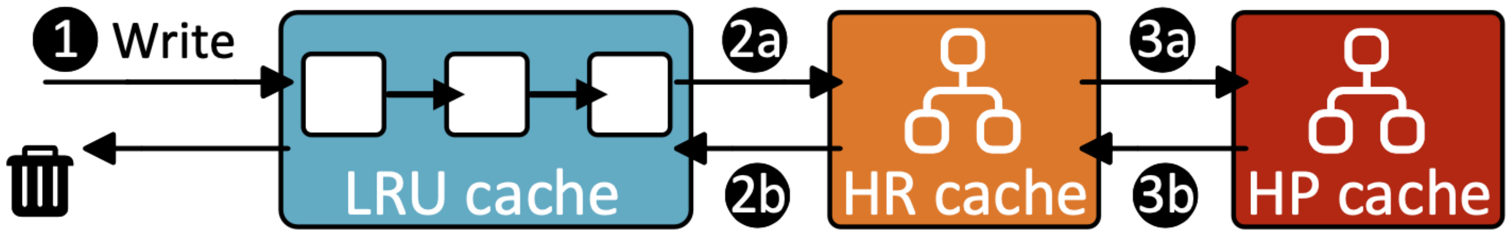


Overview of BIZA

*Please refer to our paper for more details.

Zone Group Selector

- **ZRWA will shift right when multiple new writes arrives**
- **Isolate hot data with cold ones → Keep hot data in ZRWA for a longer time**
- **Ghost-cache-based algorithm** [1]
  - High revenue = the chunk will be updated multiple times
  - High profit = High revenue + Small cost (the chunk has short reuse distance)



HR cache: 2a Reaccess number > threshold T_1

2b Evict minimum reaccess number

HP cache: 3a Reuse distance < threshold T_2

3b Evict maximum reuse distance

[1] S-RAC: SSD friendly caching for data center workloads. (SYSTOR'16)

Zone Group Selector

- **ZRWA will shift right when multiple new writes arrives**
- **Isolate hot data with cold ones → Keep hot data in ZRWA for a longer time**
- **Ghost-cache-based algorithm [1]**
  - High revenue = the chunk will be updated multiple times
  - High profit = High revenue + Small cost (the chunk has short reuse distance)
- **Select different zone group for chunks in different cache**
  - High profit: can be updated multiple times within the limited ZRWA
  - High revenue: hard to be absorbed in ZRWA but is the main source of GC

[1] S-RAC: SSD friendly caching for data center workloads. (SYSTOR'16)

Experimental Setup

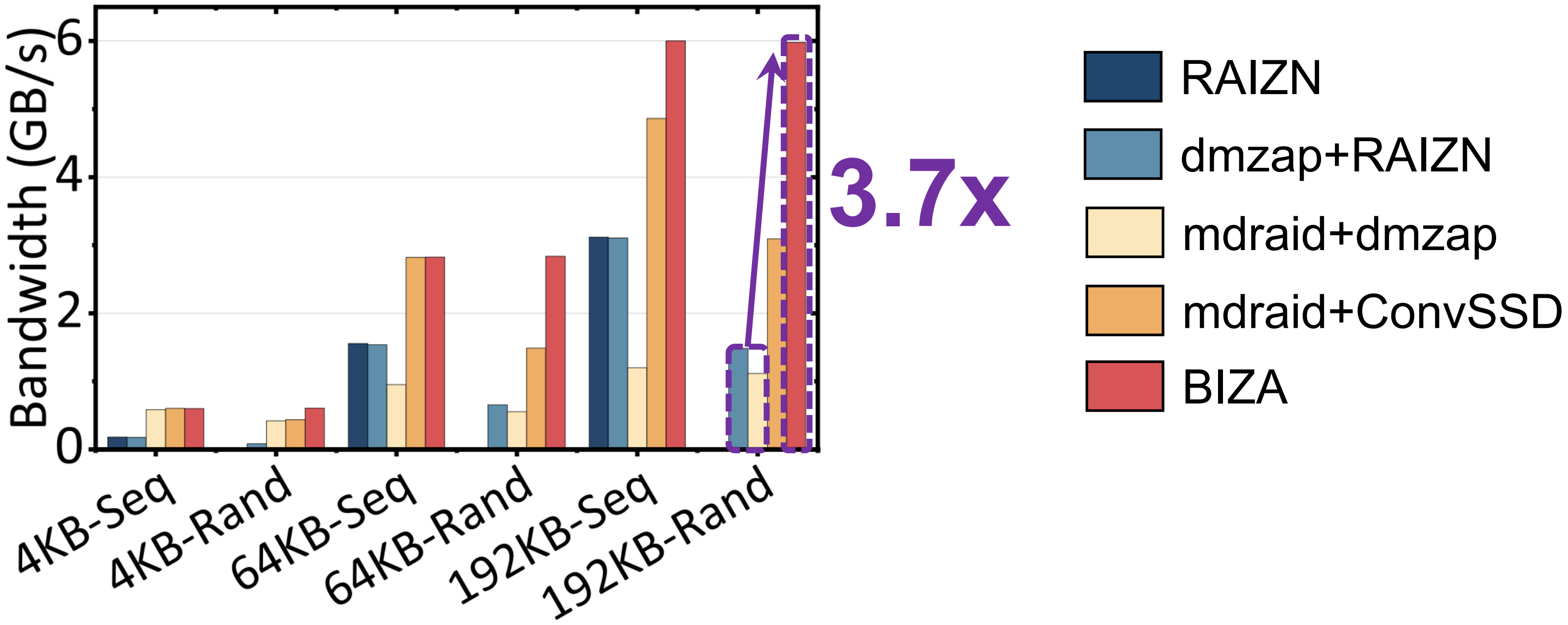
- **Implemented as a device mapper in Linux kernel**
- **Constructing AFA with 4 commodity SSDs as RAID 5**
- **ZNS SSD:** Western Digital ZN540 SSD
  - Read: 3265 MB/s, Write: 2170 MB/s
  - Up to 14 open zones & 14 MB ZRWA
- **Conventional SSD:** Western Digital SN640 SSD
  - Read: 3331 MB/s, Write: 2250 MB/s
  - Developed on a similar hardware basis as ZN540 SSD
- **Comparison**
  - RAIZN: only support sequential write
  - dmzap+RAIZN: stacking dm-zap on RAIZN for random writes
  - mdraid+dmzap: stacking dm-zap on each ZNS SSD
  - mdraid+ConvSSD: constructing AFA with conventional SSDs

WD ZN540 SSD

WD SN640 SSD

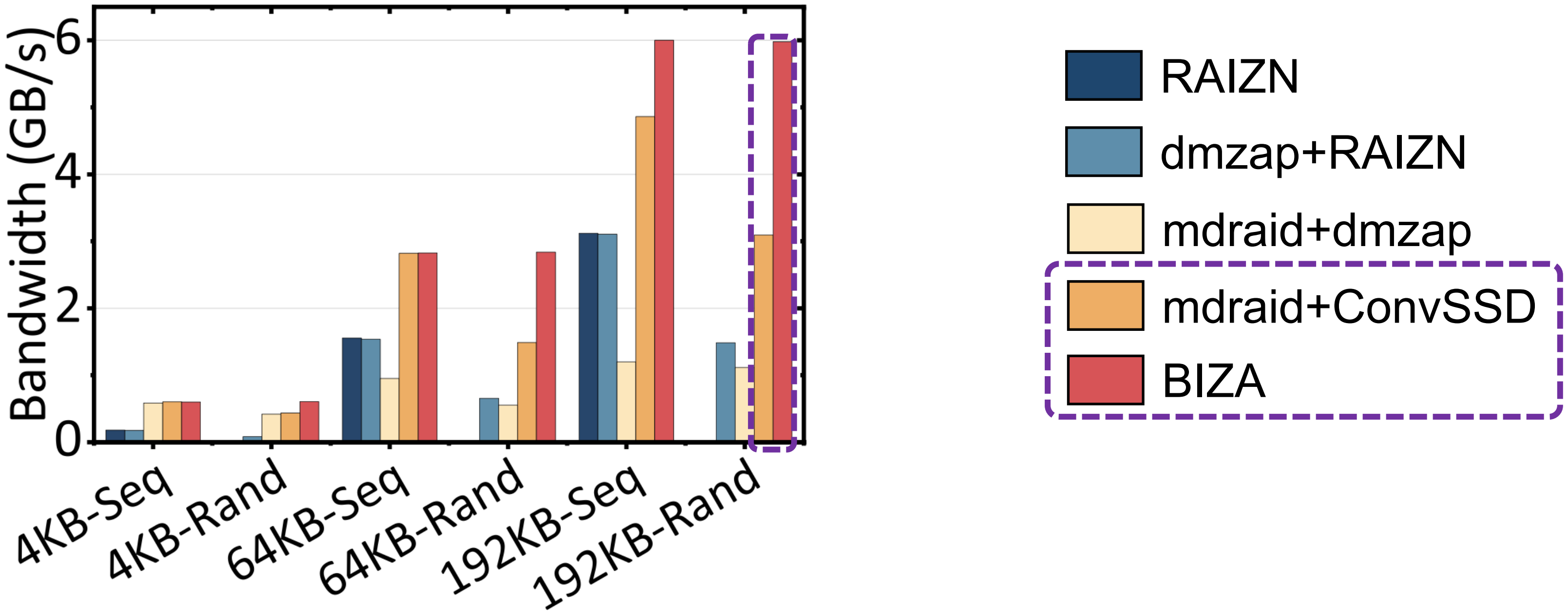
Microbenchmark

- **Bandwidth**
  - BIZA achieves 3.7x and 3.5x bandwidth than dmzap+RAIZN and mdraid+dmzap



Microbenchmark

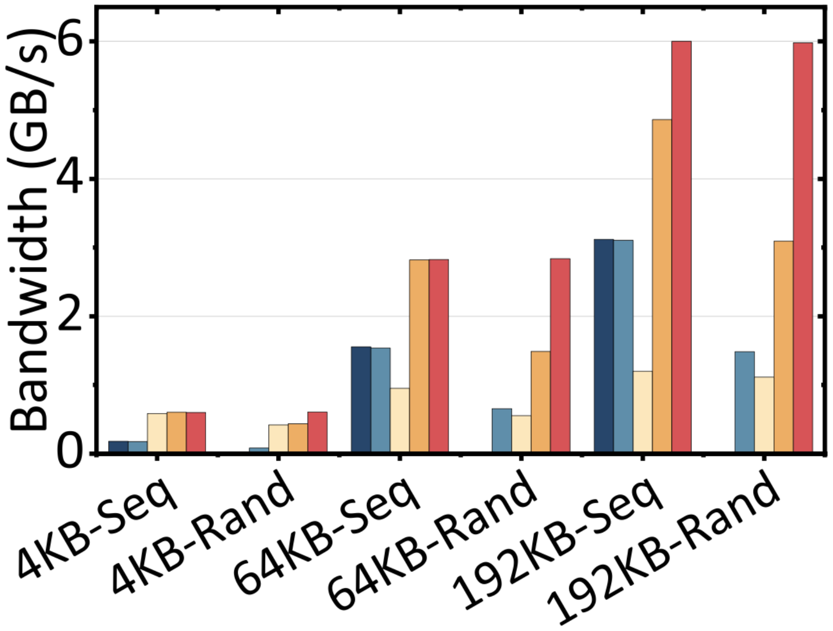
- **Bandwidth**
  - BIZA achieves 3.7x and 3.5x bandwidth than dmzap+RAIZN and mdraid+dmzap
  - Even 1.4x bandwidth than mdraid+ConvSSD (software overhead of mdraid[1-2])
  - Almost exhaust the throughput of 4 ZNS SSDs (92.2% of the ideal)

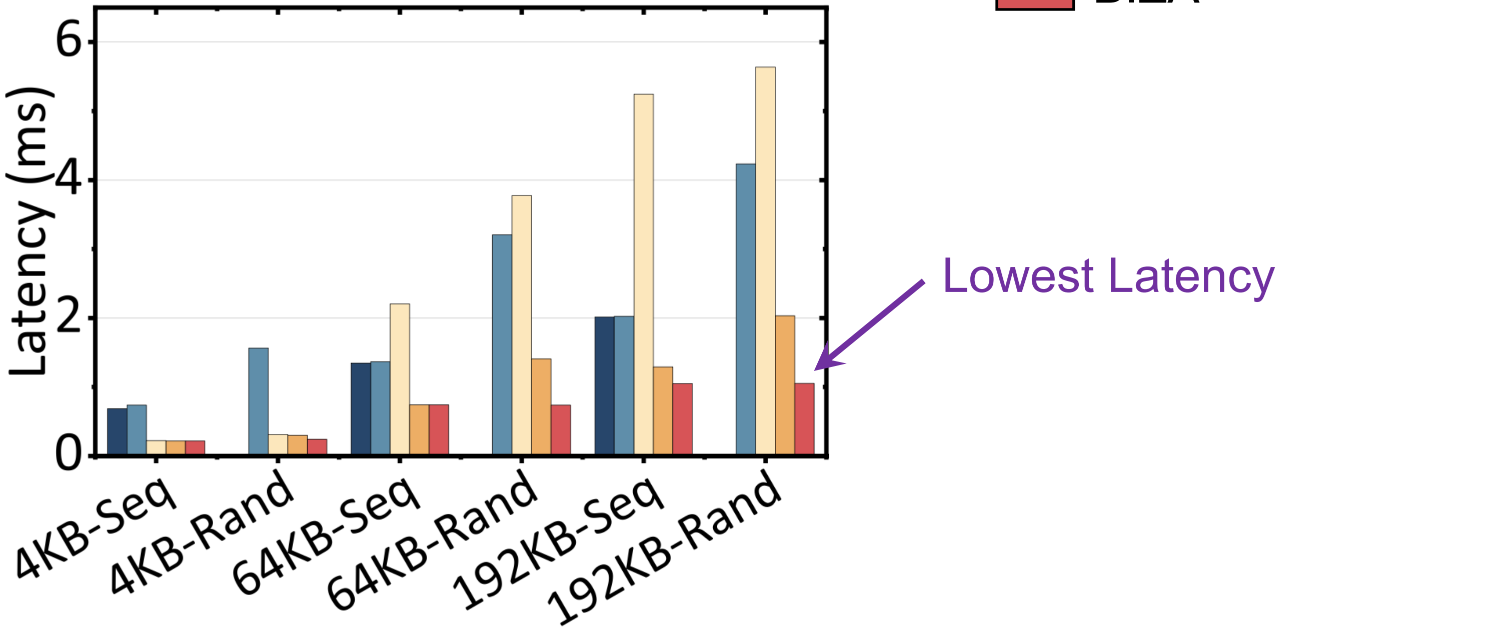


[1] ScalaRAID: Optimizing linux software raid system for next-generation storage. (ATC'24)
[2] stRAID: Stripe-threaded architecture for parity-based RAIDs with ultra-fast SSDs. (ATC'22)

Microbenchmark

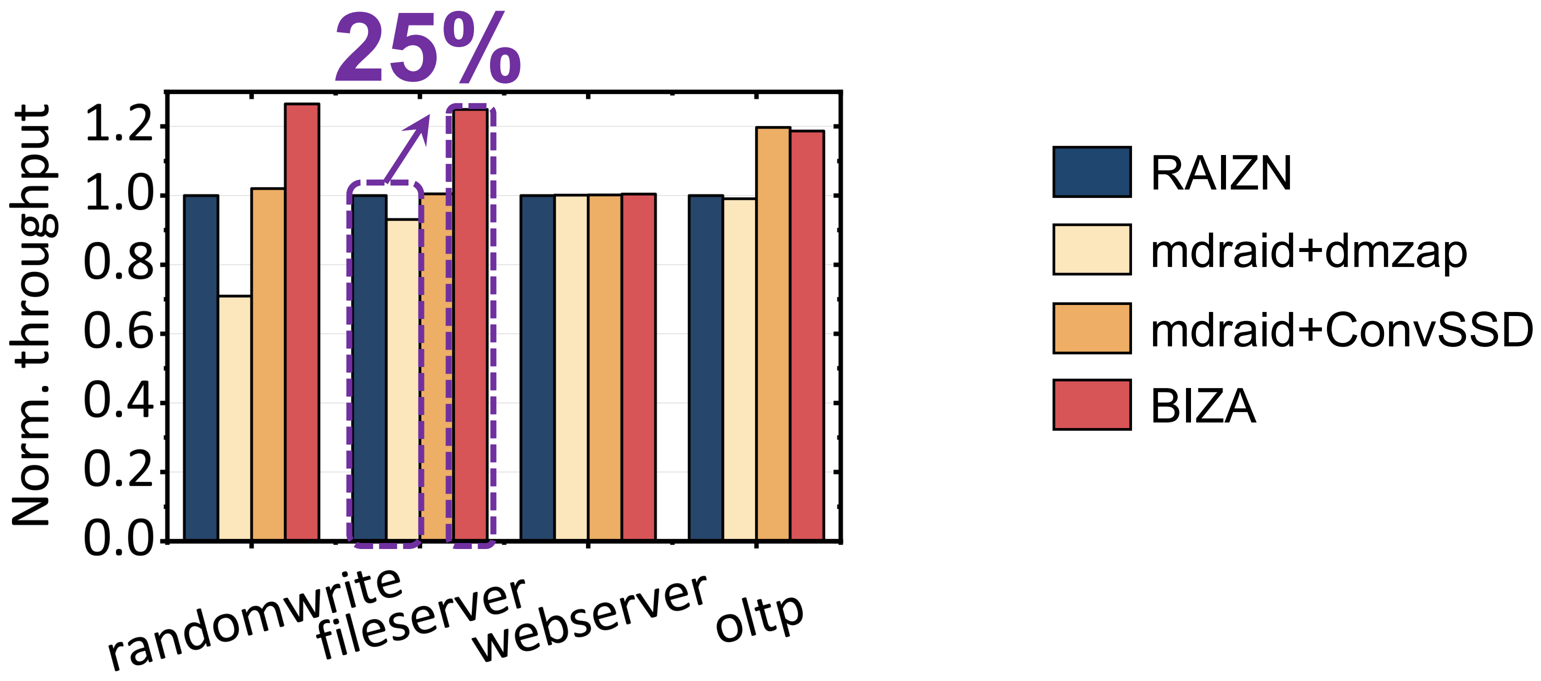
- **Bandwidth**
  - BIZA achieves 3.7x and 3.5x bandwidth than dmzap+RAIZN and mdraid+dmzap
  - Even 1.4x bandwidth than mdraid+ConvSSD (software overhead of mdraid[1-2])
  - Almost exhaust the throughput of 4 ZNS SSDs (92.2% of the ideal)
- **Average latency**
  - Outperform RAIZN by 53.8% for sequential writes





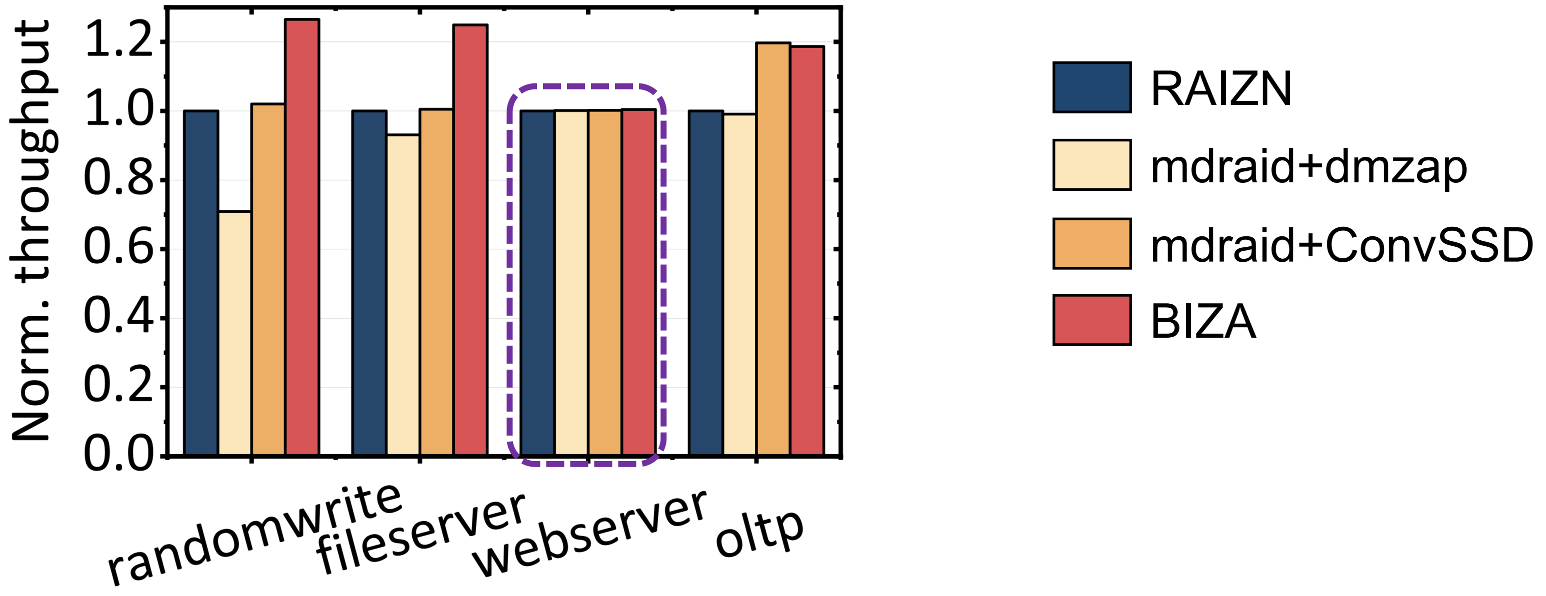
Applications

- **Filesystem:** F2FS + filebench
  - BIZA outperforms RAIZN by 26.6%, 24.9%, and 18.7% in randomwrite, fileserver, and oltp



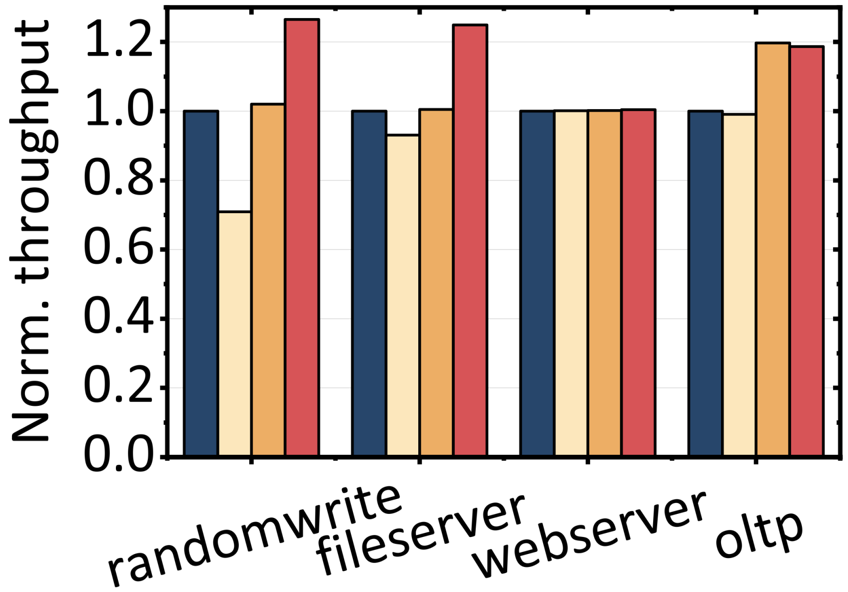
Applications

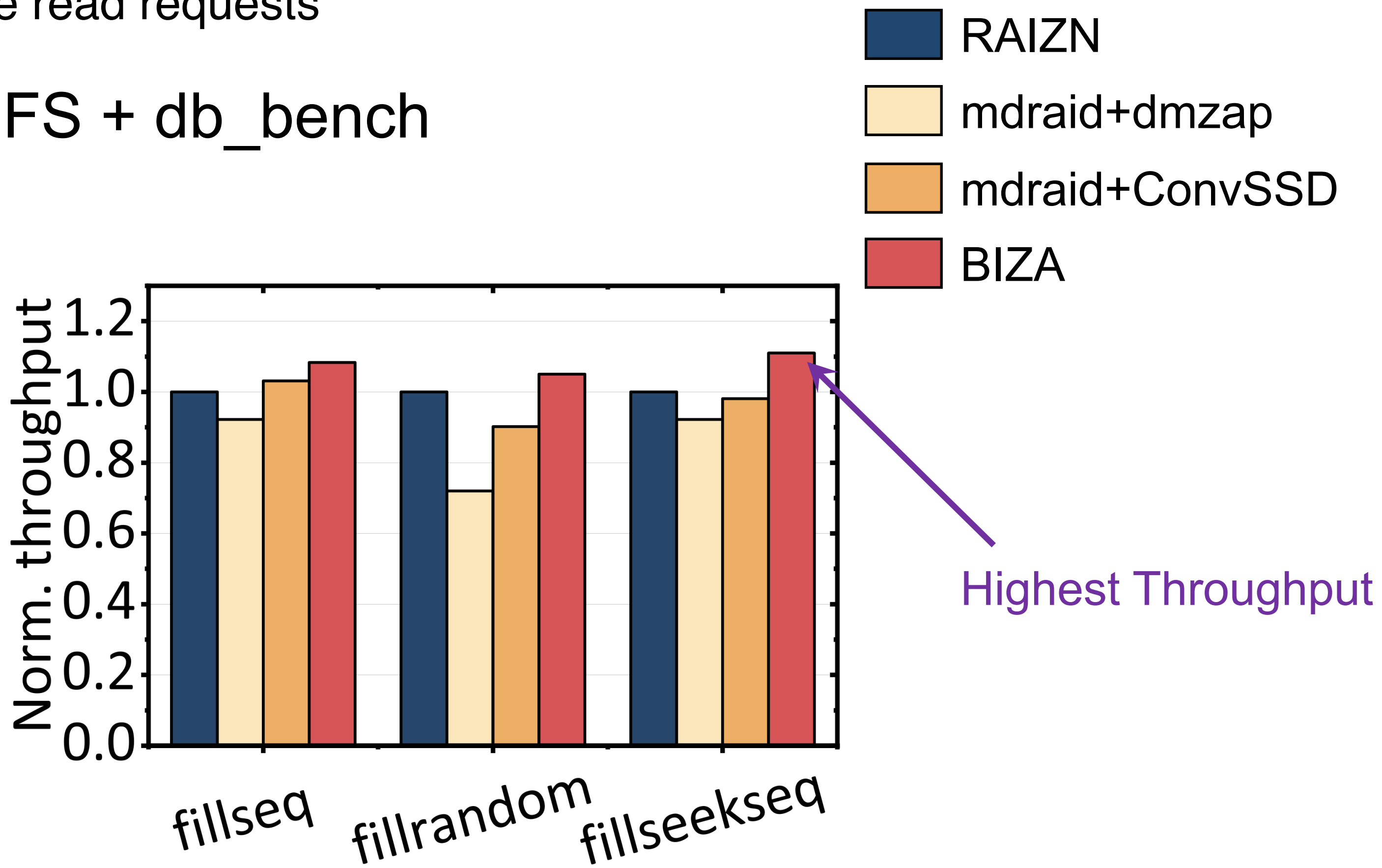
- **Filesystem:** F2FS + filebench
  - BIZA outperforms RAIZN by 26.6%, 24.9%, and 18.7% in randomwrite, fileserver, and oltp
  - Most (95.2%) requests in webserver are read requests



Applications

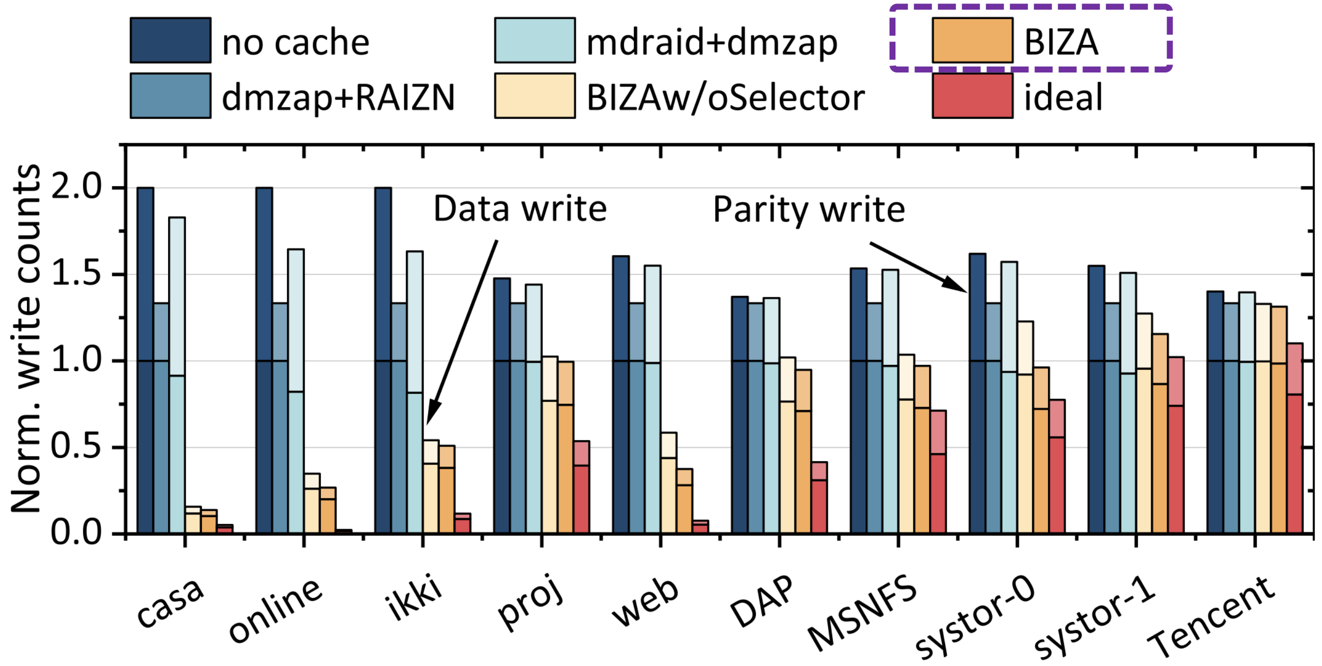
- **Filesystem:** F2FS + filebench
  - BIZA outperforms RAIZN by 26.6%, 24.9%, and 18.7% in randomwrite, fileserver, and oltp
  - Most (95.2%) requests in webserver are read requests
- **Key-value store:** RocksDB + F2FS + db_bench
  - 10.5% higher throughput than RAIZN



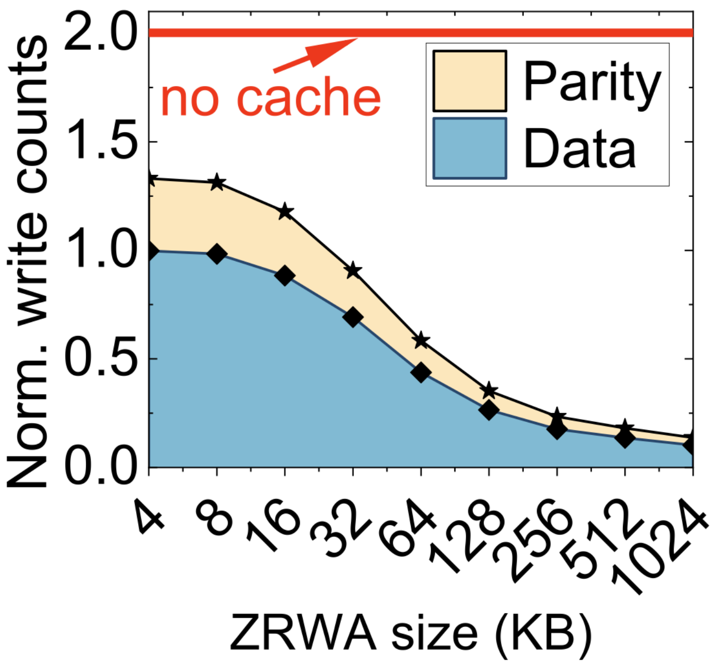


Write Amplification Reduction

- **Reducing 41.0% flash writes (compared with mdraid+dmzap)**
- **Different sizes of ZRWA:** larger ZRWA, less flash writes



Write amplification



Sensitivity study

Conclusion

- **Existing interface choices of AFA are all unsatisfactory**
  - Block-interface alone, ZNS-interface alone, and adapter
  - Endurance, performance, and compatibility
- **BIZA: benefit from the openness of ZNS whilst exposing block interface**
  - Fully exploiting the emerging ZRWA feature
  - Exhausting the intra-zone and inter-zone parallelism
- **Significantly improve I/O performance while mitigates write amplification**
- **Source code is accessible at:**
  - *https://github.com/ChaseLab-PKU/BIZA*

Thanks & QA

BIZA: Design of Self-Governing Block-Interface ZNS AFA for Endurance and Performance

https://github.com/ChaseLab-PKU/BIZA

Shushu Yi, Shaocong Sun, Li Peng, Yingbo Sun
Ming-Chang Yang, Zhichao Cao, Qiao Li,
Myoungsoo Jung, Ke Zhou, Jie Zhang

CHASELab
Computer Hardware And System Evolution Laboratory

北京大學 PEKING UNIVERSITY