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

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

Presented by Jingze, Qingyuan, Lijun

2024-11-26

Outline

- Background & Challenge
- ZNS SSD Exploration
- Design & Implementation
- Evaluation

Outline

- Background & Challenge
- ZNS SSD Exploration
- Design & Implementation
- Evaluation

Background

- All-flash array (AFA)
  - Only flash memory drives
  - Array → RAID

Background

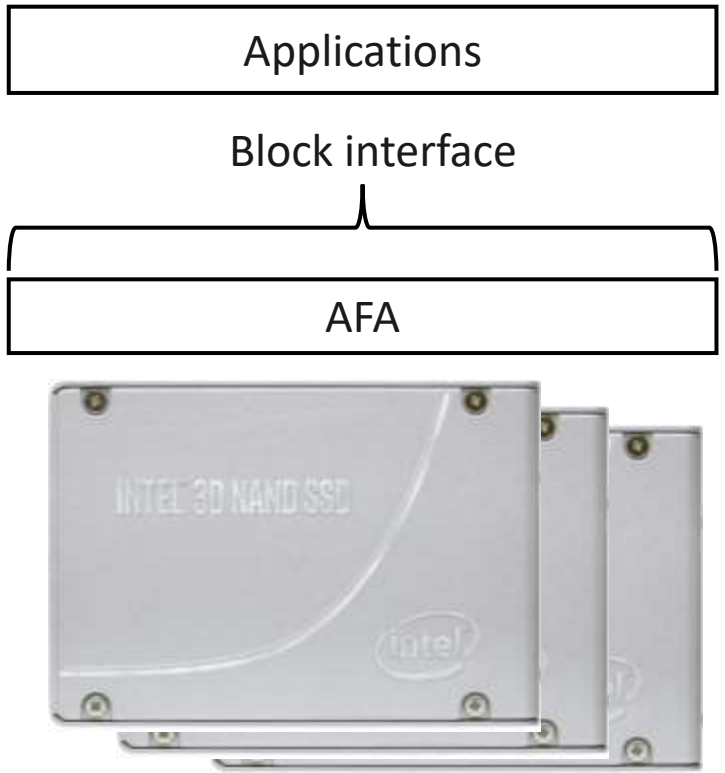
- All-flash array (AFA)
  - Only flash memory drives
  - Array → RAID

- High speed!

Background

- Issue
  - Block interface  SSD internal hardware characteristics

Mismatch



Background

- Issue
  - Block interface  SSD internal hardware characteristics

Mismatch

  - NAND SSDs
    - Out-place update

Block X			
A	B	C	D
E	free	free	free
free	free	free	free
free	free	free	free

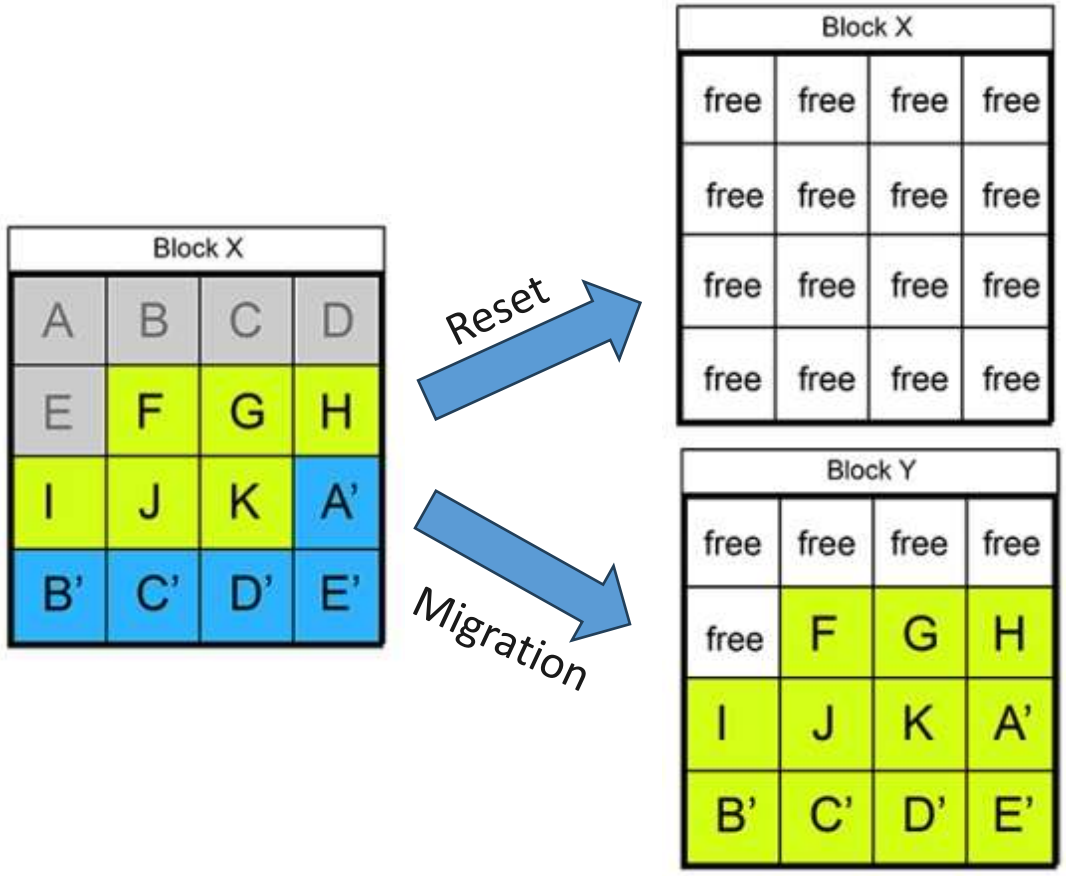
Block X			
A	B	C	D
E	F	G	H
I	J	K	A’
B’	C’	D’	E’

Background

- Issue
  - Block interface  SSD internal hardware characteristics

Mismatch

  - NAND SSDs
    - Out-place update
    - GC

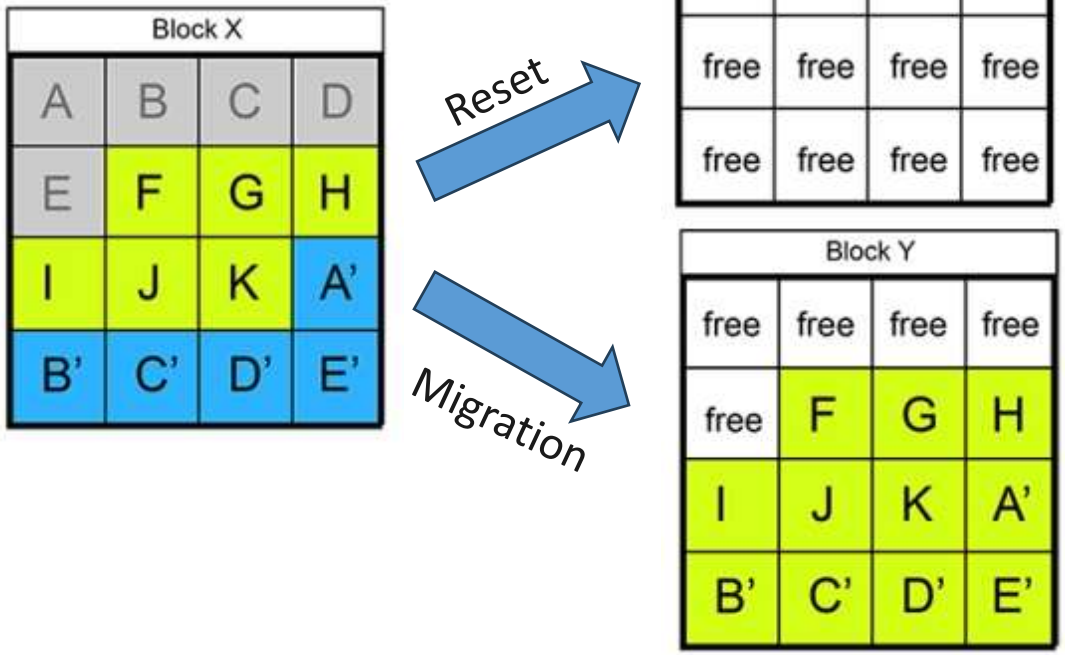


Background

- Issue
  - Block interface  SSD internal hardware characteristics

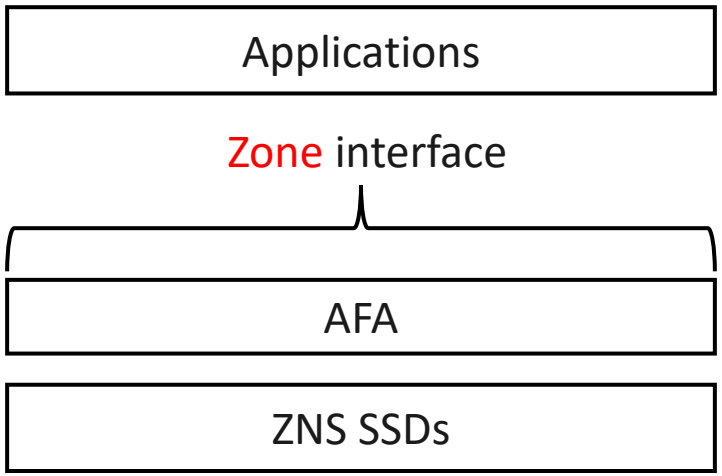
Mismatch

  - NAND SSDs
    - Out-place update
    - GC → High interference



Background

- RAIZN[1]
  - Uses ZNS SSDs for AFA
  - Zone interface matches SSD hardware characteristics

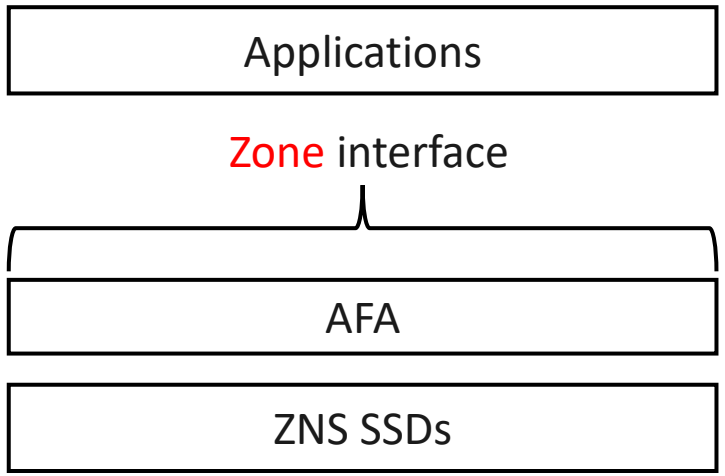


[1] ASPLOS'23 RAIZN: Redundant Array of Independent Zoned Namespaces

Background

- RAIZN[1]
  - Uses ZNS SSDs for AFA
  - Zone interface matches SSD hardware characteristics

- RAIZN constrain
  - Zone interface
  - Host layer GC
  - Host layer data management
  - → Poor application compatibility

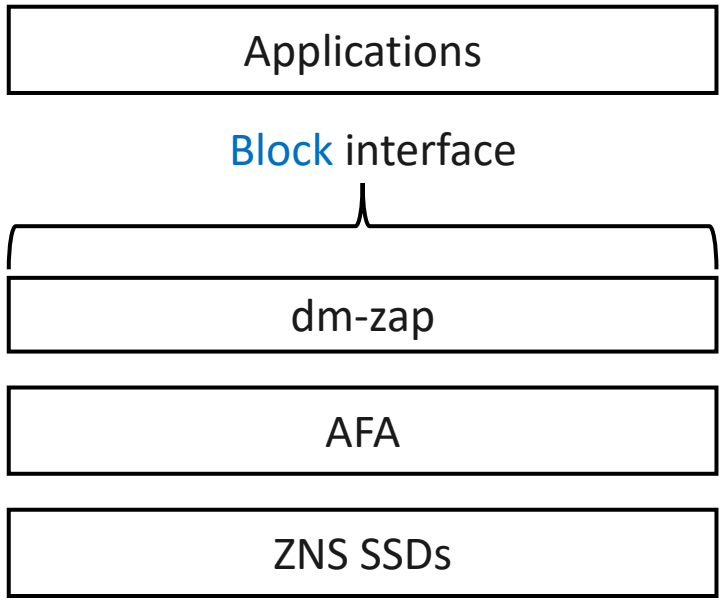


[1] ASPLOS'23 RAIZN: Redundant Array of Independent Zoned Namespaces

Background

- RAIZN + dm-zap[2]
  - Block interface again

- New issues
  - Worse endurance
  - Poor performance
  - Higher tail latency
  - Write amplification

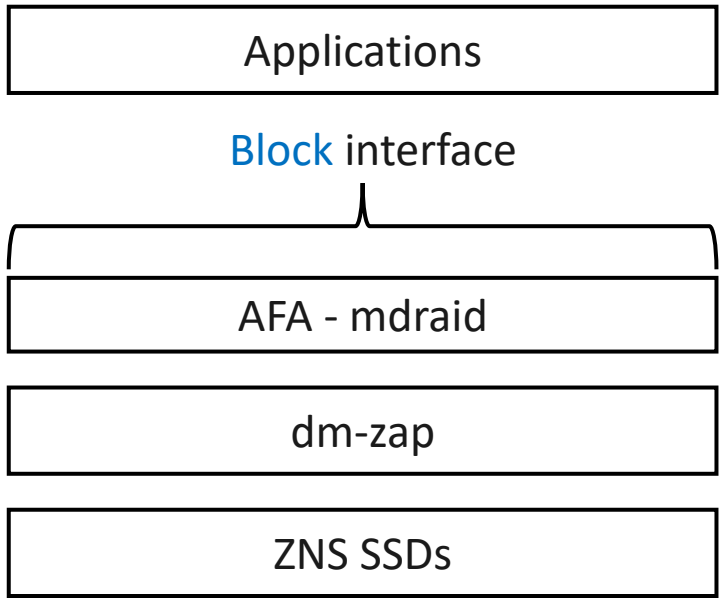


[2] dm-zap, https://github.com/westerndigitalcorporation/dm-zap

Background

- dm-zap + mdraid
  - Also block interface

- Also has the above issues



Challenge

- Issues → No design with zone interface benefits

Challenge

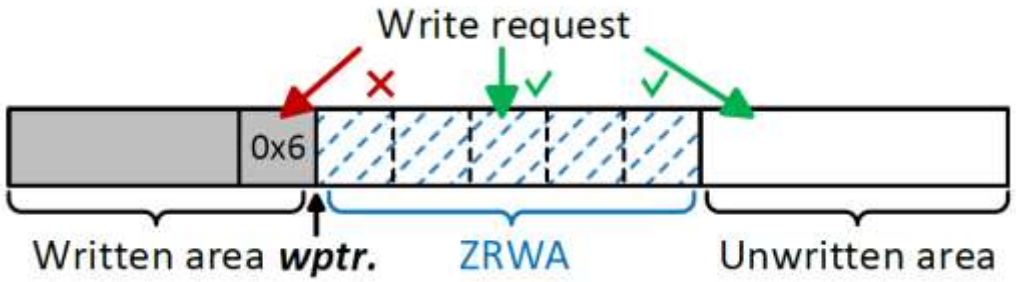
- Issues → No design with zone interface benefits
  - Worse endurance
    - Write amplification
  - Poor performance
    - Limited zone used and low concurrency
  - Higher tail latency
    - GC

Outline

- Background & Challenge
- ZNS SSD Exploration
- Design & Implementation
- Evaluation

Zone Random Write Area

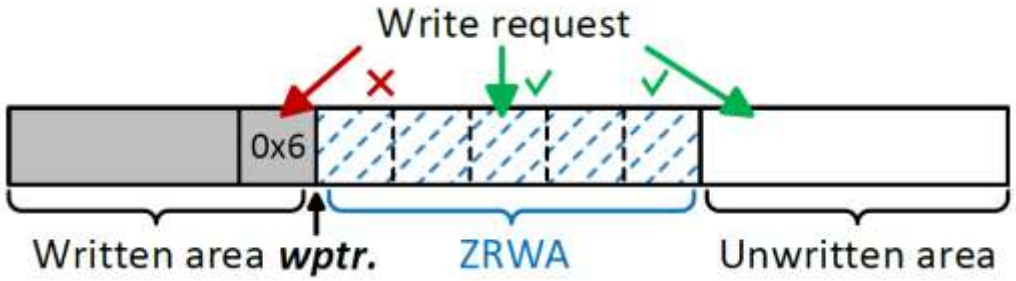
- ZRWA
  - An abstraction of write cache inside SSDs
  - Range: [wptr, wptr + ZRWA size]



Zone Random Write Area

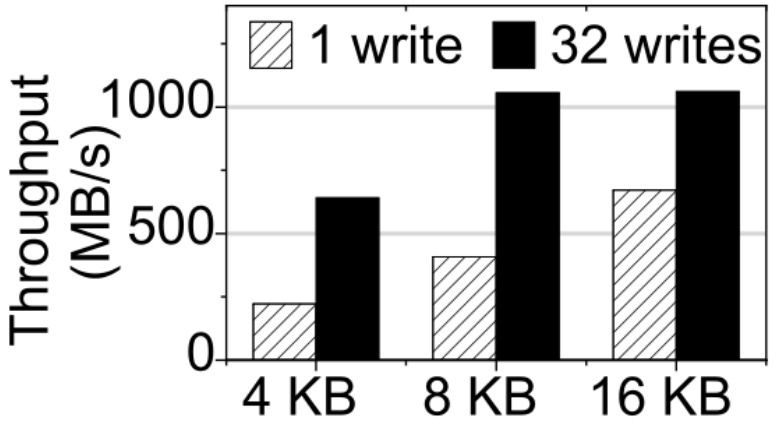
- ZRWA
  - An abstraction of write cache inside SSDs
  - Range: [wptr, wptr + ZRWA size]

  - Write iodepth > 1
  - In-place update
  - Write beyond ZRWA range



Zone Random Write Area

- Performance potential
  - Improves 4-16KB I/O concurrency and throughput for each zone
  - 4KB: About 240MB → 700MB
  - 8KB: About 460MB → 1100MB
  - 16KB: About 740MB → 1100MB



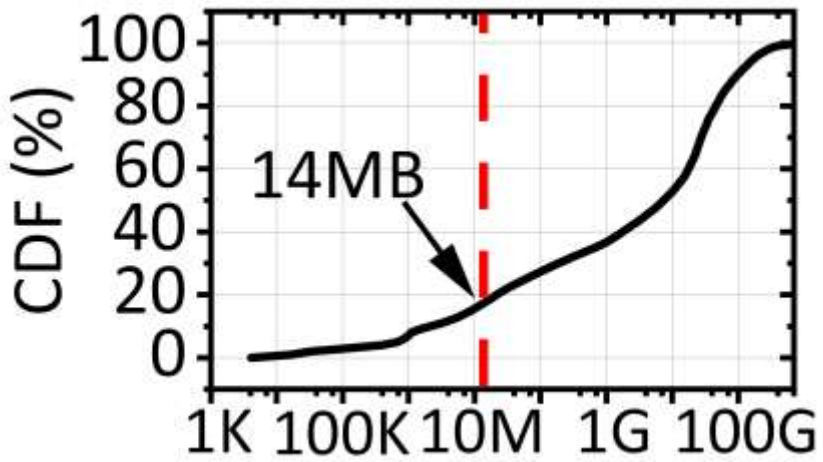
Zone Random Write Area

- ZRWA issue
  - ZRWA size is very limited
  - About 10-20MB

ZNS SSD prototype	Zone capacity	ZRWA size per open zone	Max. # of open zones	LR-SW
WD ZN540	1077 MB	1 MB	14	14 MB
DapuStor J5500Z	18144 MB	1 MB	16	16 MB
Inspur NS8600G	2880 MB	1440 KB	8	11.25 MB
Samsung PM1731a	96 MB	64 KB	384	24 MB

Zone Random Write Area

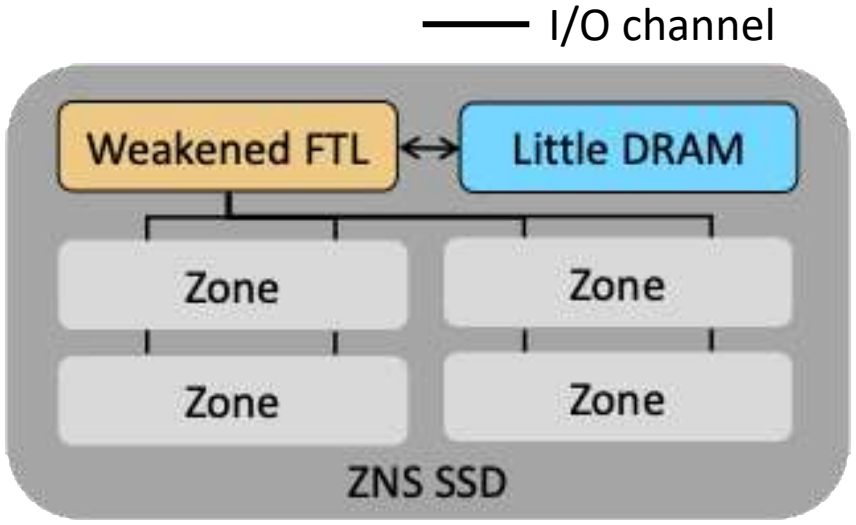
- ZRWA issue
  - ZRWA size is very limited
  - About 10-20MB
  - Application rewrite distances
    - 20% <= 10MB
    - How to let rewrite benefit from ZRWA in-place update?



Inter-zones isolation

- Zones have I/O channel isolation
  - Lower interference write zones with different I/O channel
  - 2x Bandwidth and lower latency

  - → Can only be identified when zone opens
    - How to clearly use the channel isolation?



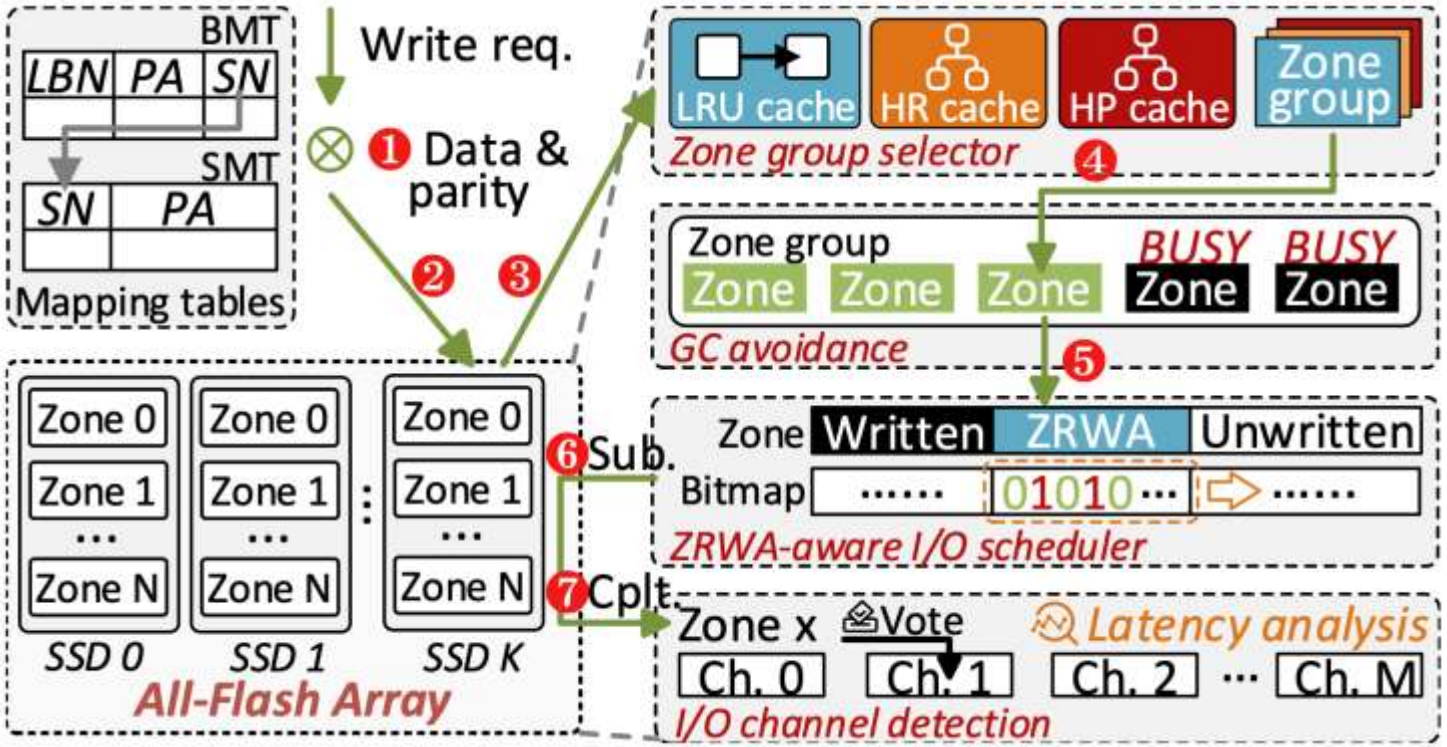
Scenario	Bandwidth (MB/s)	Avg. lat. (us)	50^{th} lat. (us)	99.99^{th} lat. (us)
Write a **single** zone	1092	59	41	570
Write **two** zones in **identical** I/O channel	1092	118	66	2310
Write **two** zones in **diverse** I/O channel	2170	59	46	685

Outline

- Background & Challenge
- ZNS SSD Exploration
- Design & Implementation
- Evaluation

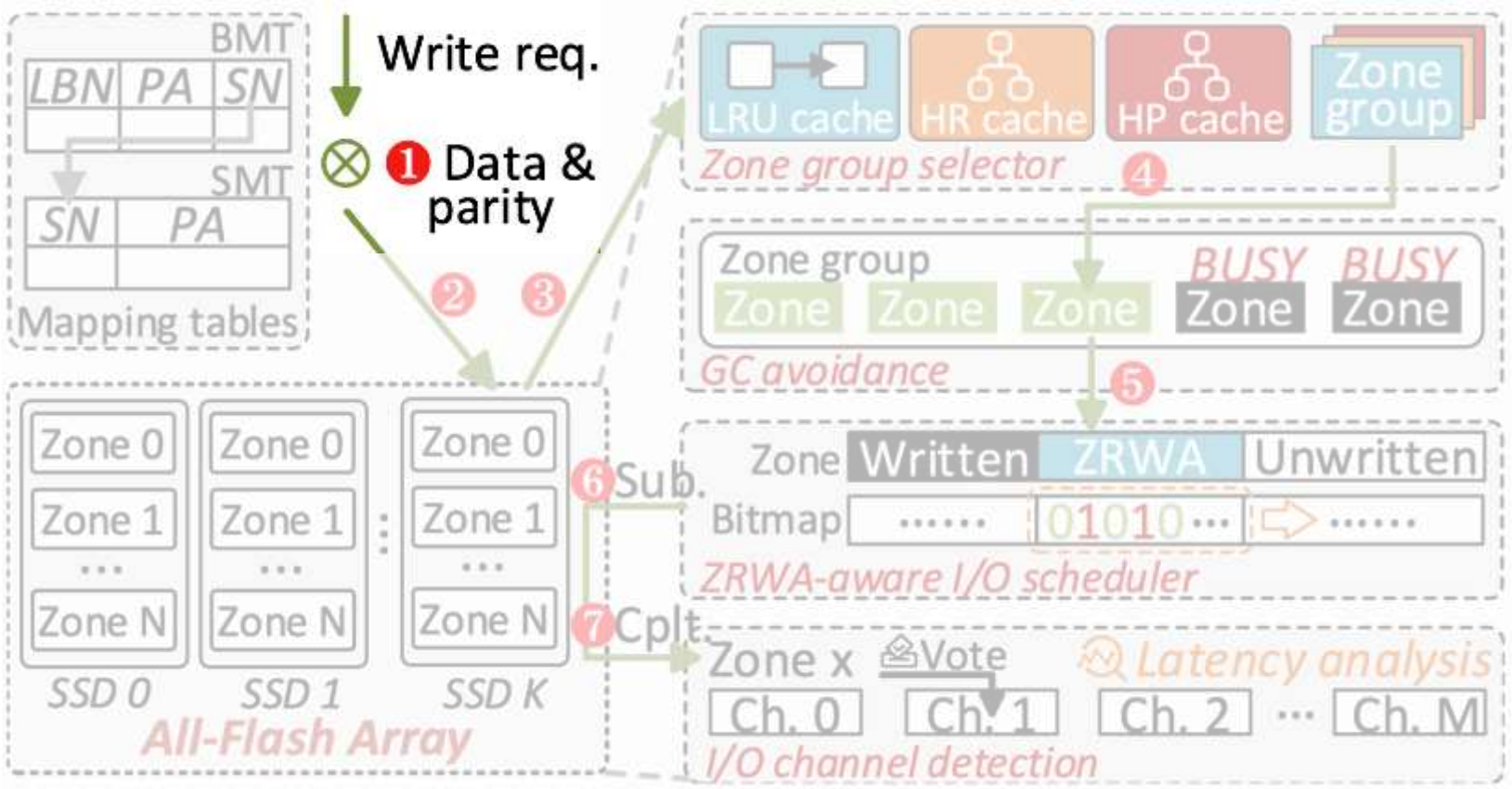
Overview architecture

- BIZA - AFA
  - Mapping table
  - Zone group selector
  - GC avoidance
  - ZRWA-aware I/O scheduler
  - I/O channel detection



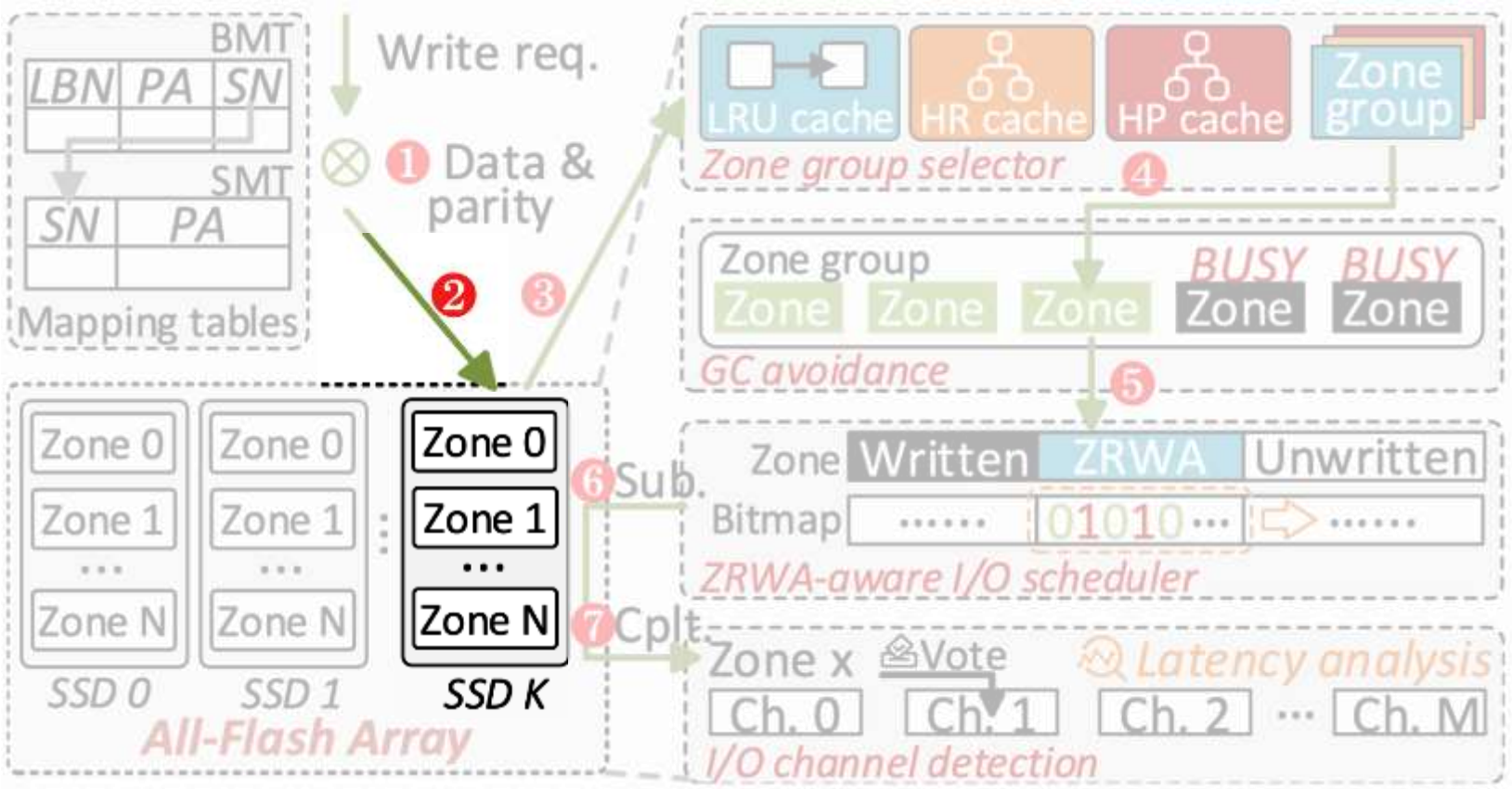
Overview architecture

- Write process
  - Write data & parities



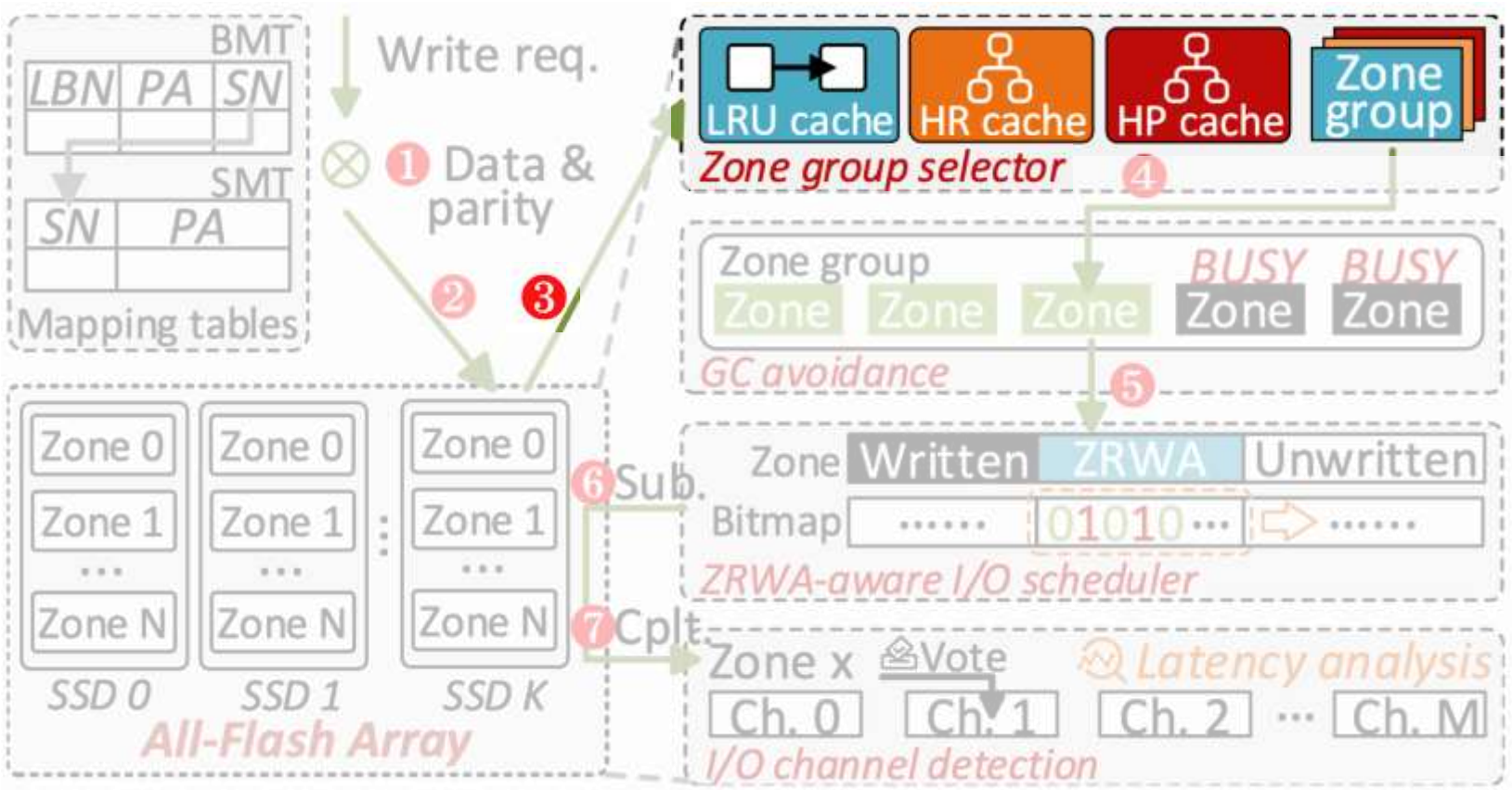
Overview architecture

- Write process
  - Choose SSD to write



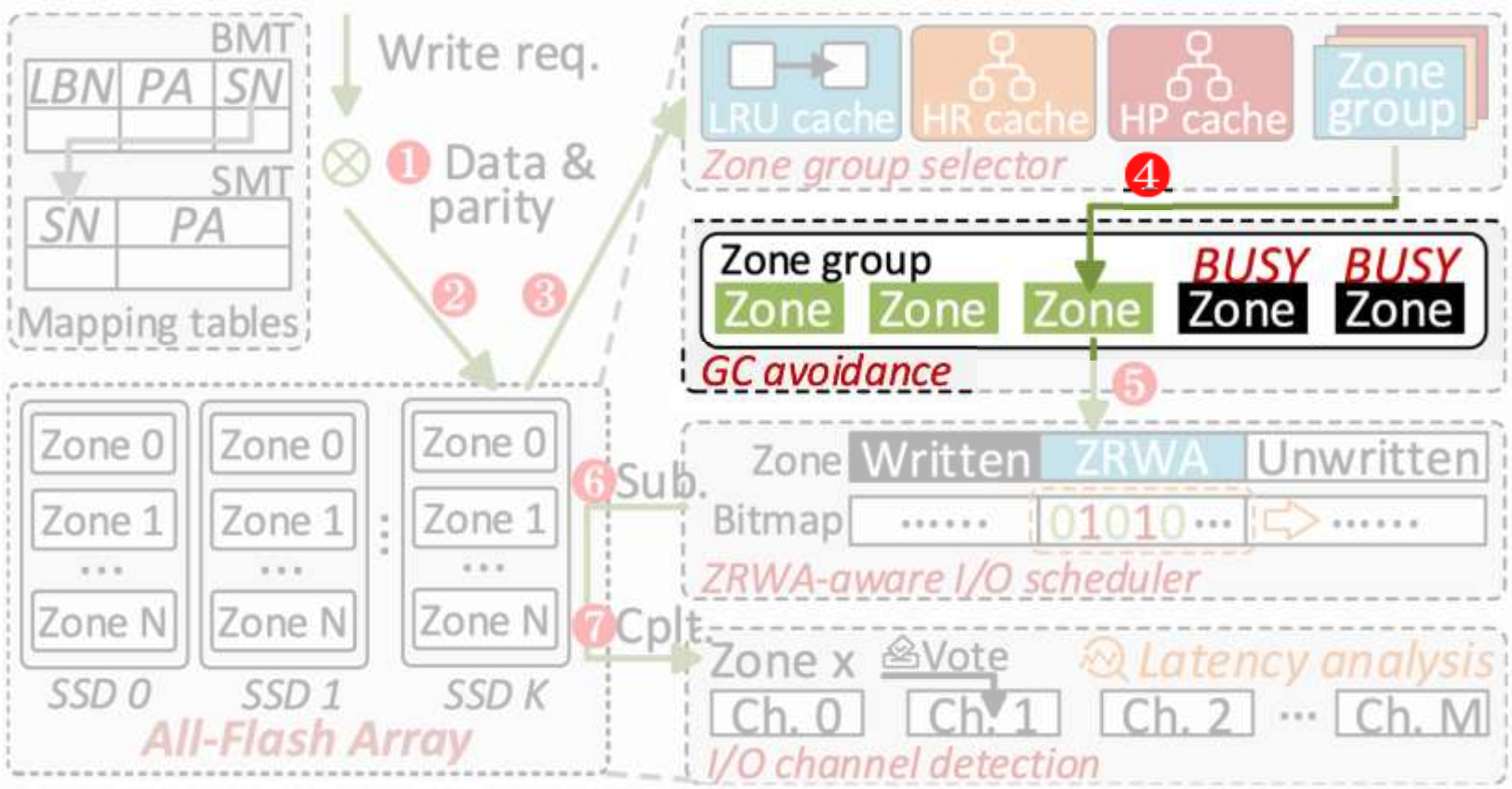
Overview architecture

- Write process
  - Choose which zone group to write
    - Zone group has zones with different I/O channel



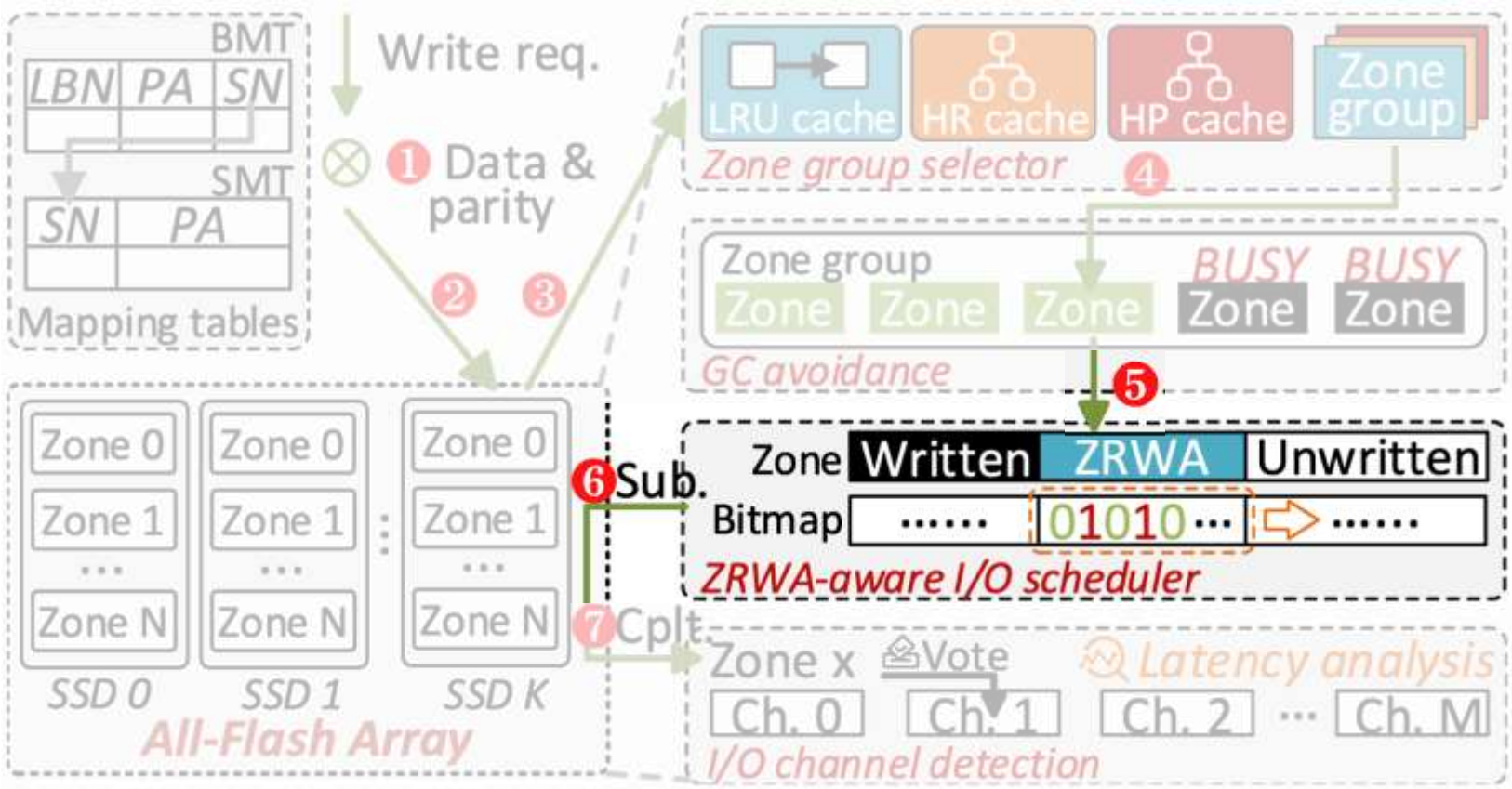
Overview architecture

- Write process
  - Choose which zone to write
    - Busy zones and writable zones are in different I/O channel



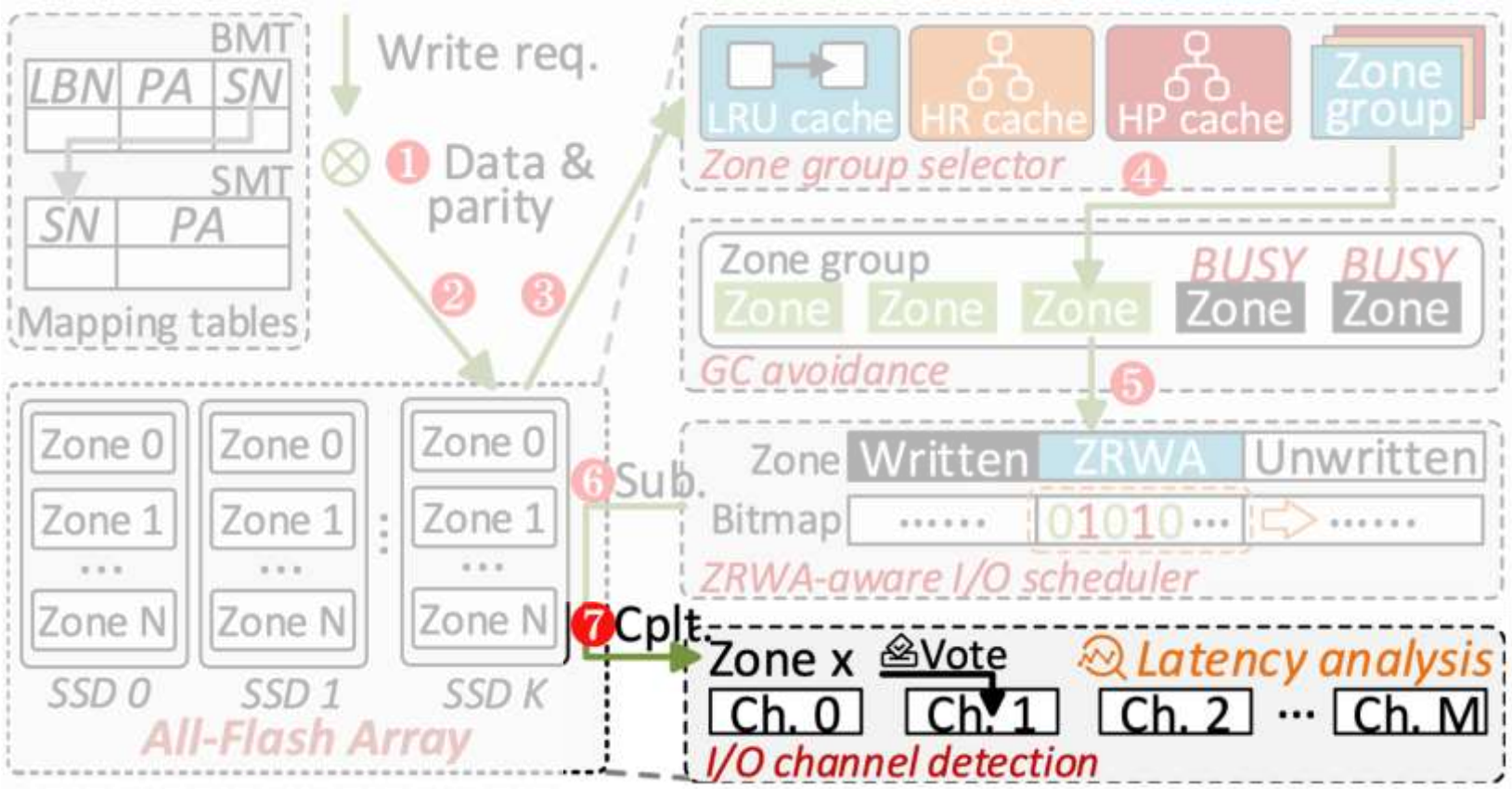
Overview architecture

- Write process
  - I/O dispatch and submit in parallel
    - With a sliding-window-based algorithm



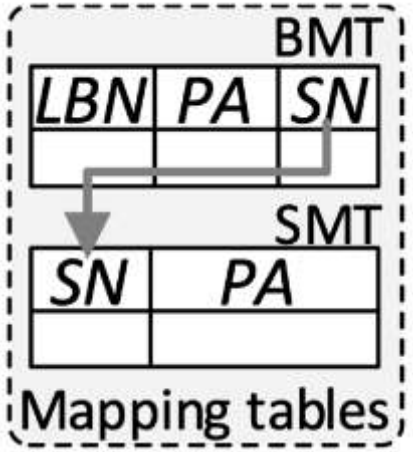
Overview architecture

- Write process
  - Latency analysis after each write completion
    - Identify the zones to I/O channel



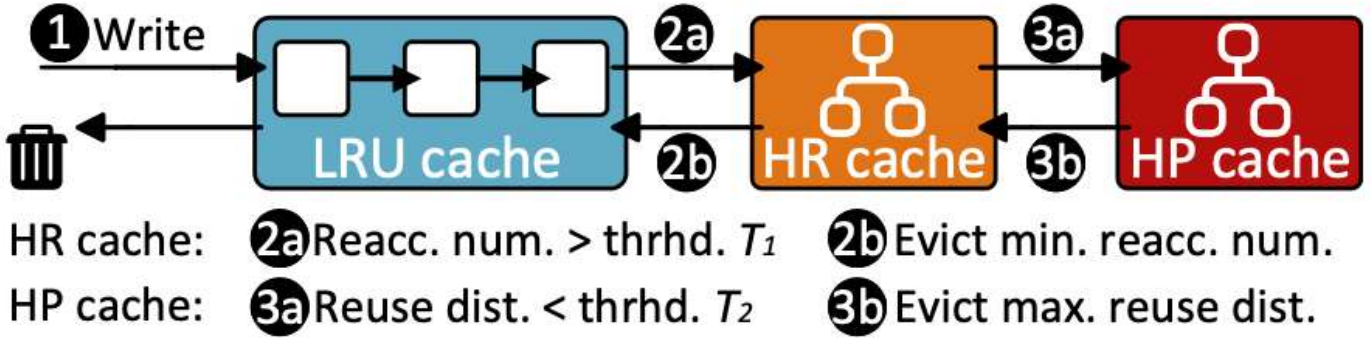
Overview architecture

- Mapping table
  - Block mapping table (BMT)
    - PA: physical address
    - SN: stripe number
    - Records data location
  - Stripe mapping table (SMT)
    - Records parities location



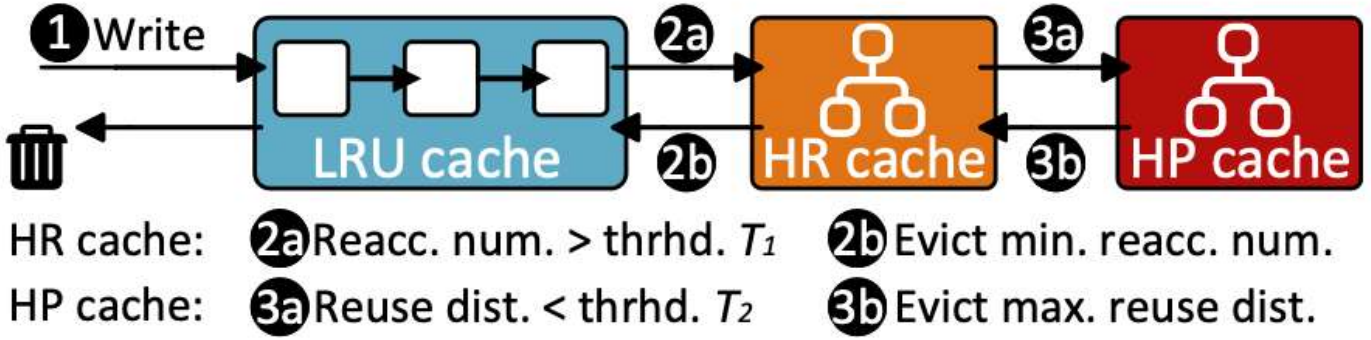
Zone group selector

- Ghost-cache-based algorithm
  - LRU cache
    - Not hit: filter out the chunks with poor temporal locality
    - Hit: predict rewrite distance and reaccess number



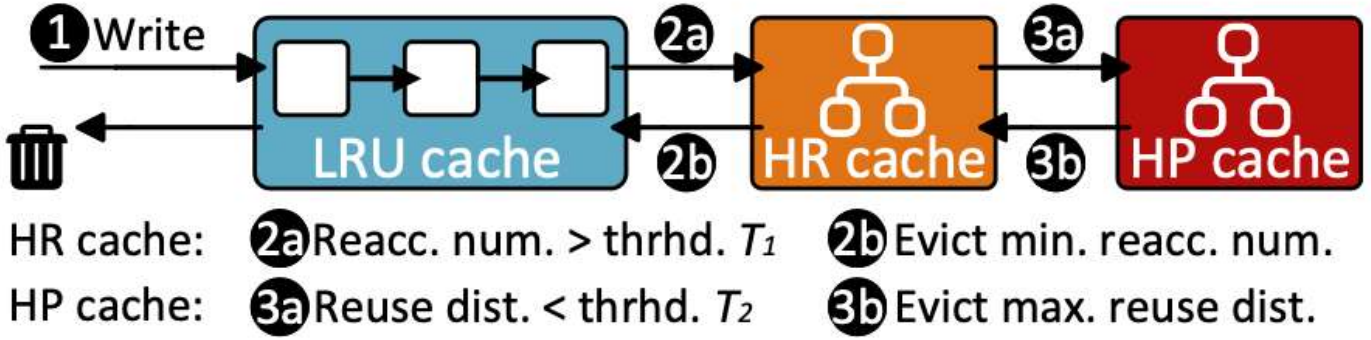
Zone group selector

- Ghost-cache-based algorithm
  - LRU cache
  - High-revenue (HR) cache
    - Predicted reaccess number > threshold (3)
    - Priority queue, evict the chunk with minimum predicted reaccess number



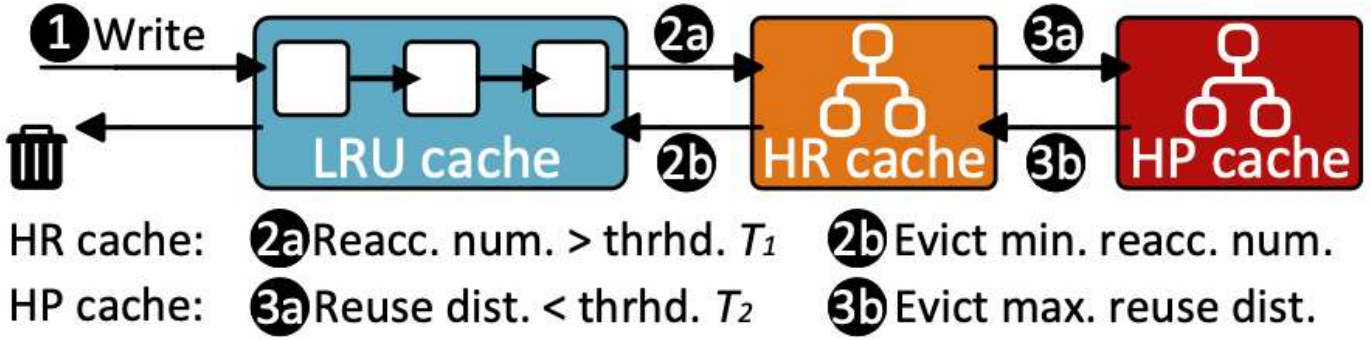
Zone group selector

- Ghost-cache-based algorithm
  - LRU cache
  - High-revenue (HR) cache
  - High-profit (HP) cache
    - Predicted rewrite distance > threshold
    - Similar evict policy as HR cache



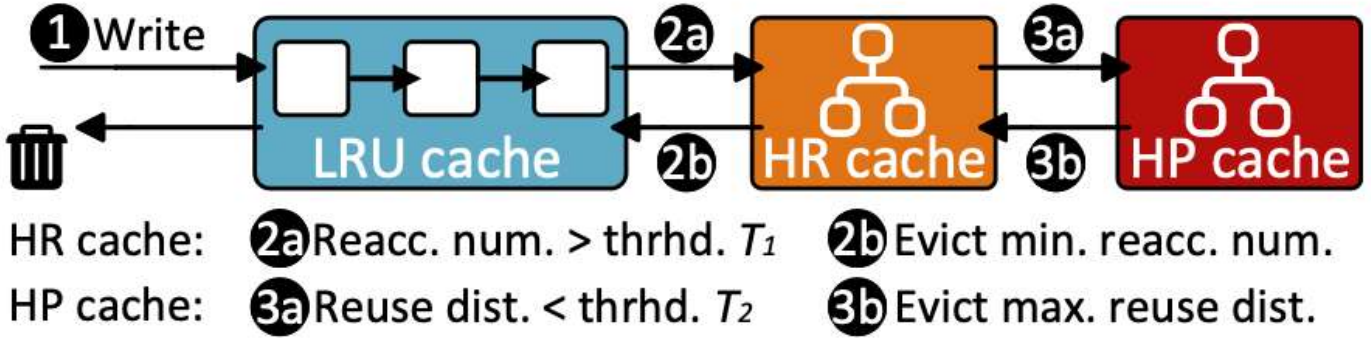
Zone group selector

- 3 zone groups
  - ZRWA-aware
  - GC-aware
  - Trivial



Zone group selector

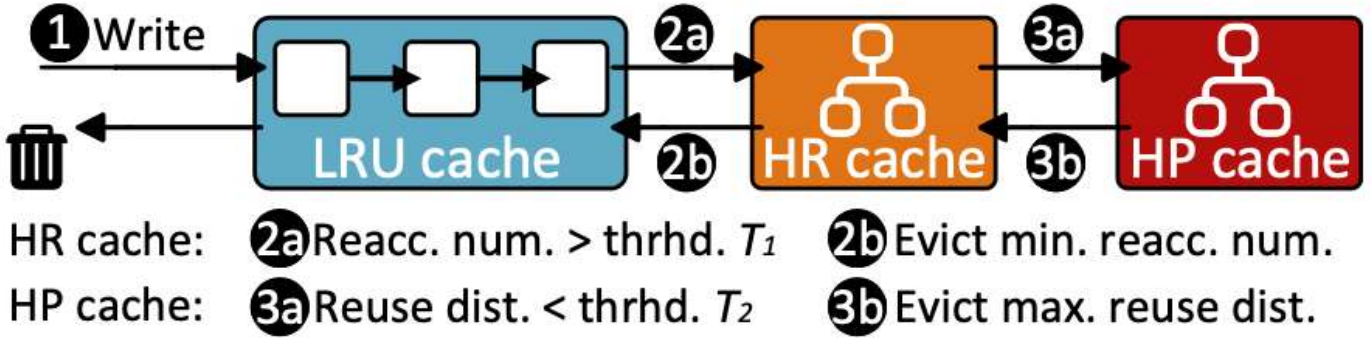
- 3 zone groups
  - ZRWA-aware → HP cache
  - GC-aware → HR cache
  - Trivial → LRU cache



Zone group selector

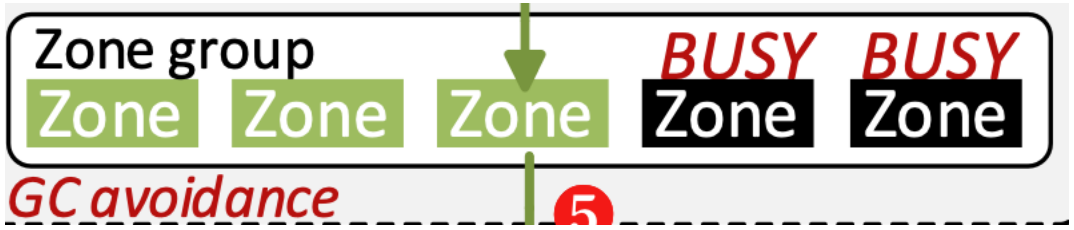
- 3 zone groups
  - ZRWA-aware → HP cache
  - GC-aware → HR cache
  - Trivial → LRU cache

  - ZRWA-aware → High performance for in-place update
  - GC-aware → Less write amplification
  - Trivial → Rare update



Avoiding GC-induced latency spikes

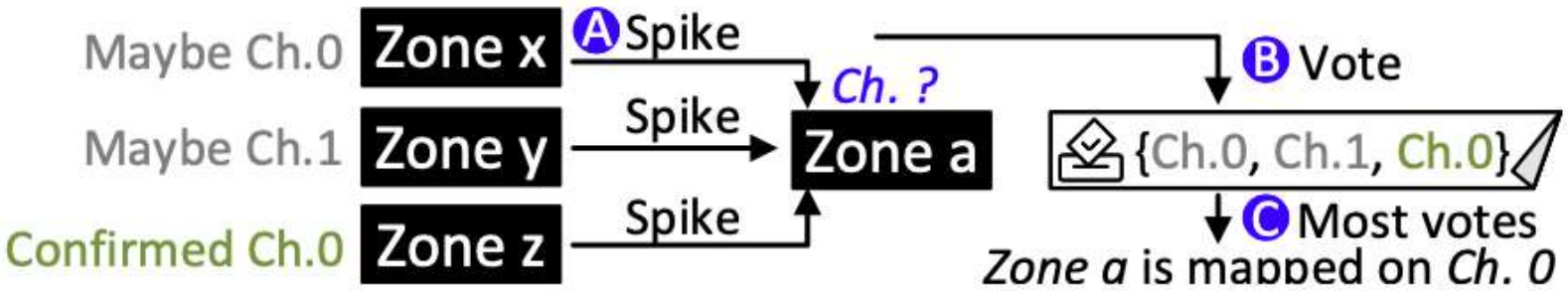
- GC avoidance
  - Use zones on different I/O channels for GC & data write
  - Busy zones: zones processing GC, or at the same channel



Avoiding GC-induced latency spikes

- I/O channel detection
  - Default: round-robin
  - Dynamic change: vote-based algorithm

  - During GC



Outline

- Background & Challenge
- ZNS SSD Exploration
- Design & Implementation
- Evaluation

Experimental Setup

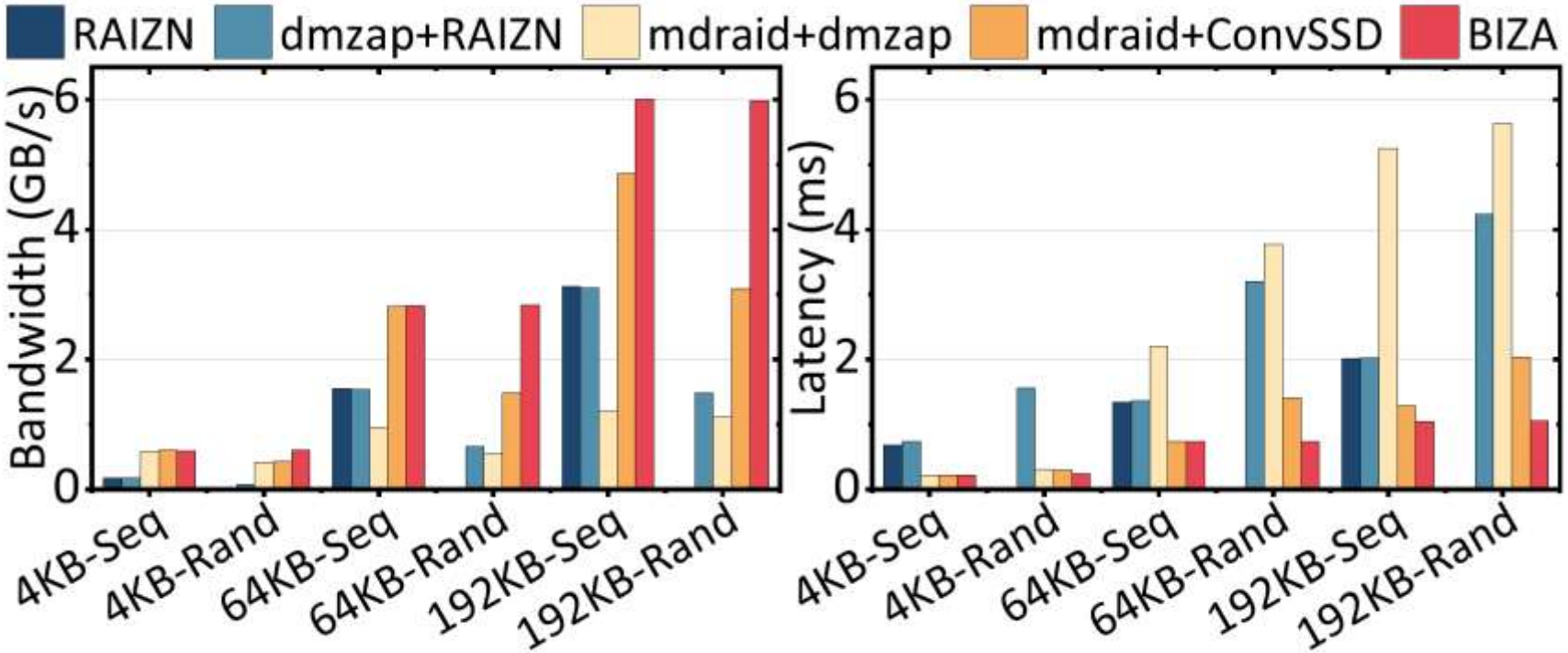
- Server
  - 26-core Intel Xeon 5320 CPU
  - 512 GB DDR4 DRAM
  - Ubuntu 22.04 kernel v5.15
  - RAID 5
  - Western Digital ZN540 4 TB SSD

Experimental Setup

- Baseline
  - Conventional SSD
    - Western Digital SN640 4 TB SSD
  - Zone interface AFA solution
    - RAIZN + ZNS
  - Block interface AFA solution
    - RAIZN + dmzap + ZNS
    - mdraid + dmzap + ZNS
    - mdraid + conventional SSD

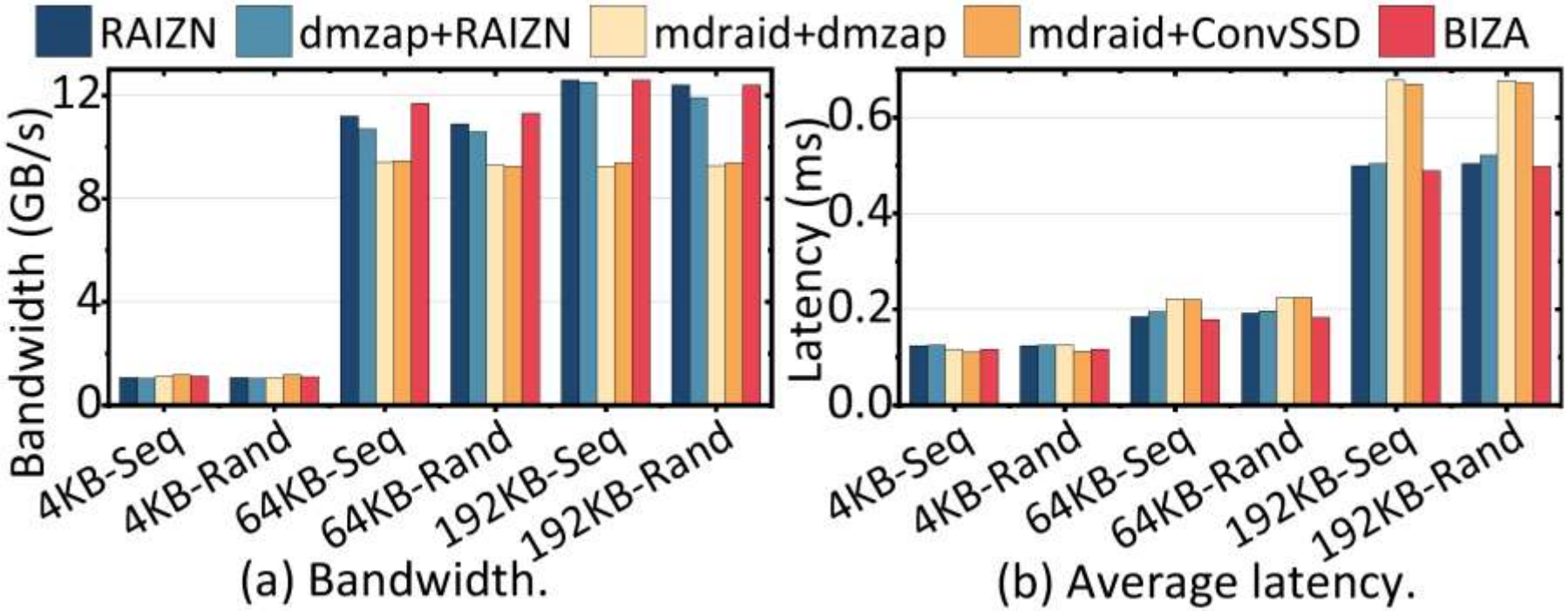
Overall performance

- FIO write microbenchmarks
  - BIZA benefits from its cache design and ZRWA-aware scheduler
  - mdraid uses write buffer and I/O merging to accelerate sequential write



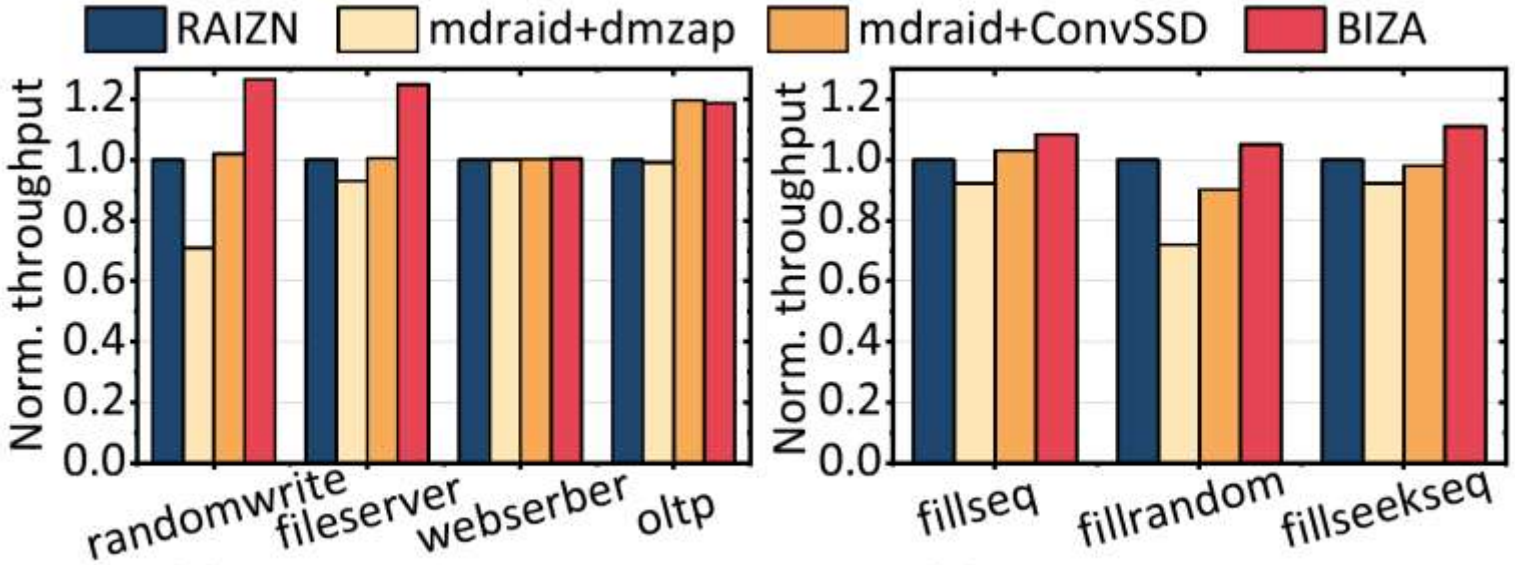
Overall performance

- FIO read microbenchmarks
  - Read performances are similar due to same read path
  - mdraid has software bottlenecks on read



Overall performance

- F2FS & RocksDB
  - BIZA has limited benefits on webserver
    - 4.8% write proportion in webserver
  - BIZA performs best in db-bench



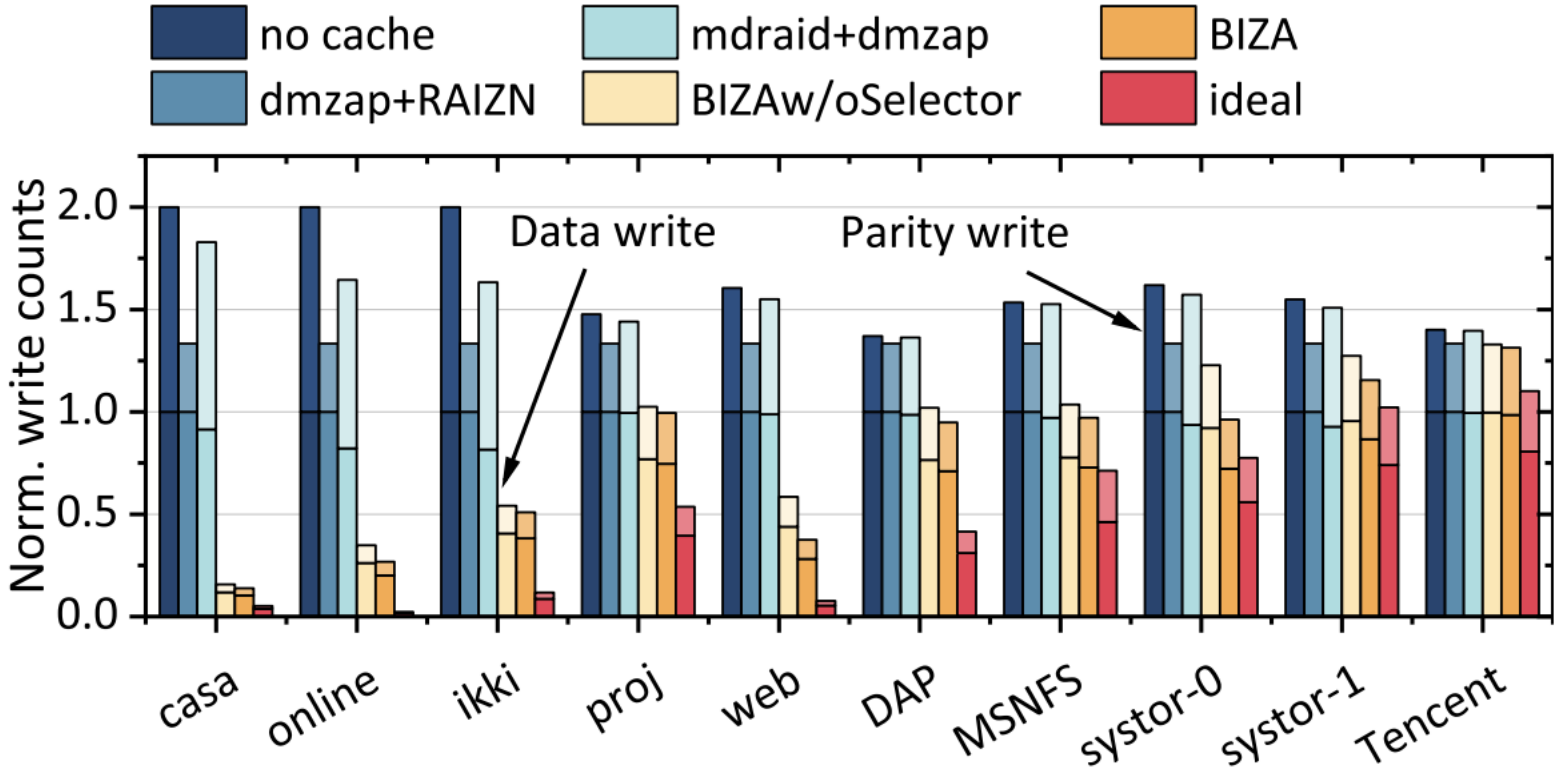
(a) F2FS and filebench. (b) RocksDB and db-bench.

Write amplification reduction

- BIZA setup
  - LRU cache 4 GB
  - HR cache 1 GB
  - HP cache 64 MB
- mdraid & RAIZN setup
  - Employ in-host-DRAM write buffer
  - 56MB (the same as sum of ZRWA size)
- Workloads
  - Real I/O traces

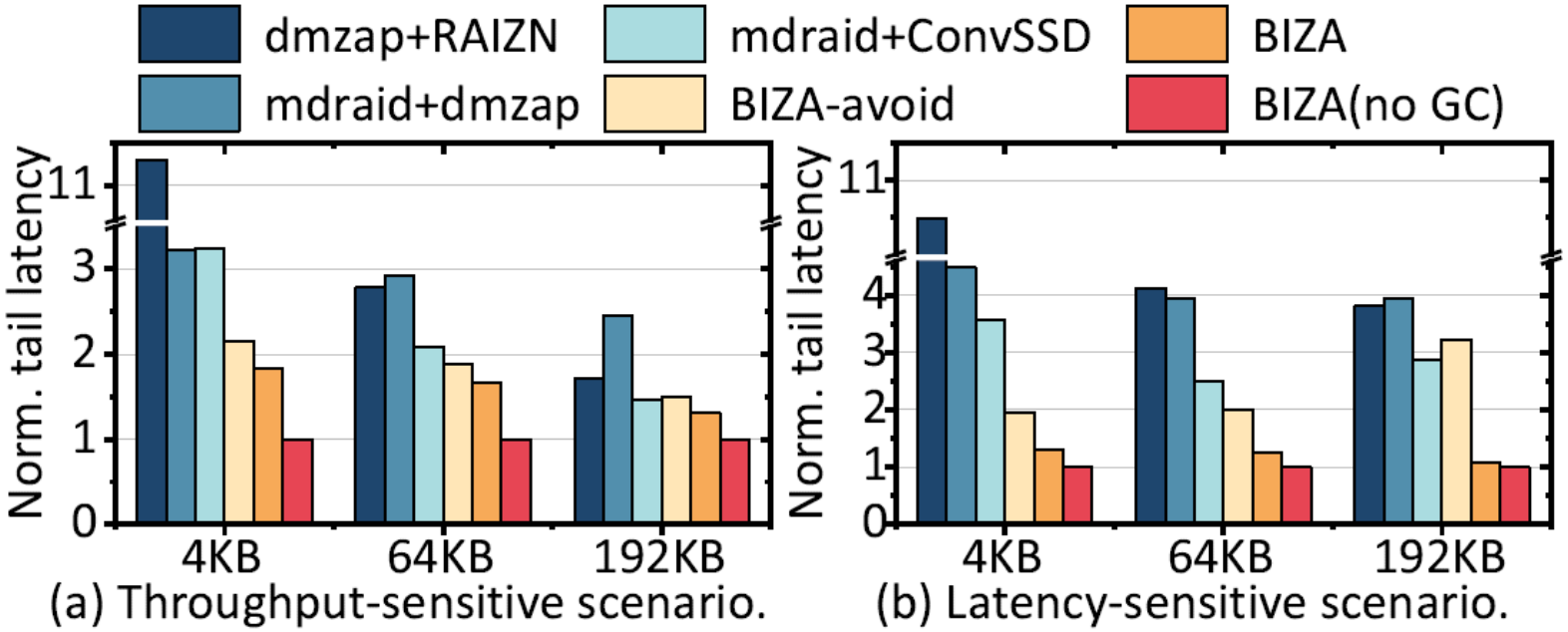
Write amplification reduction results

- Comparison of write amplification
  - No cache: no updates are absorbed in cache
  - RAIZN & mdraid benefits from its own write buffer
  - BIZA benefits from its ZRWA & zone group selector



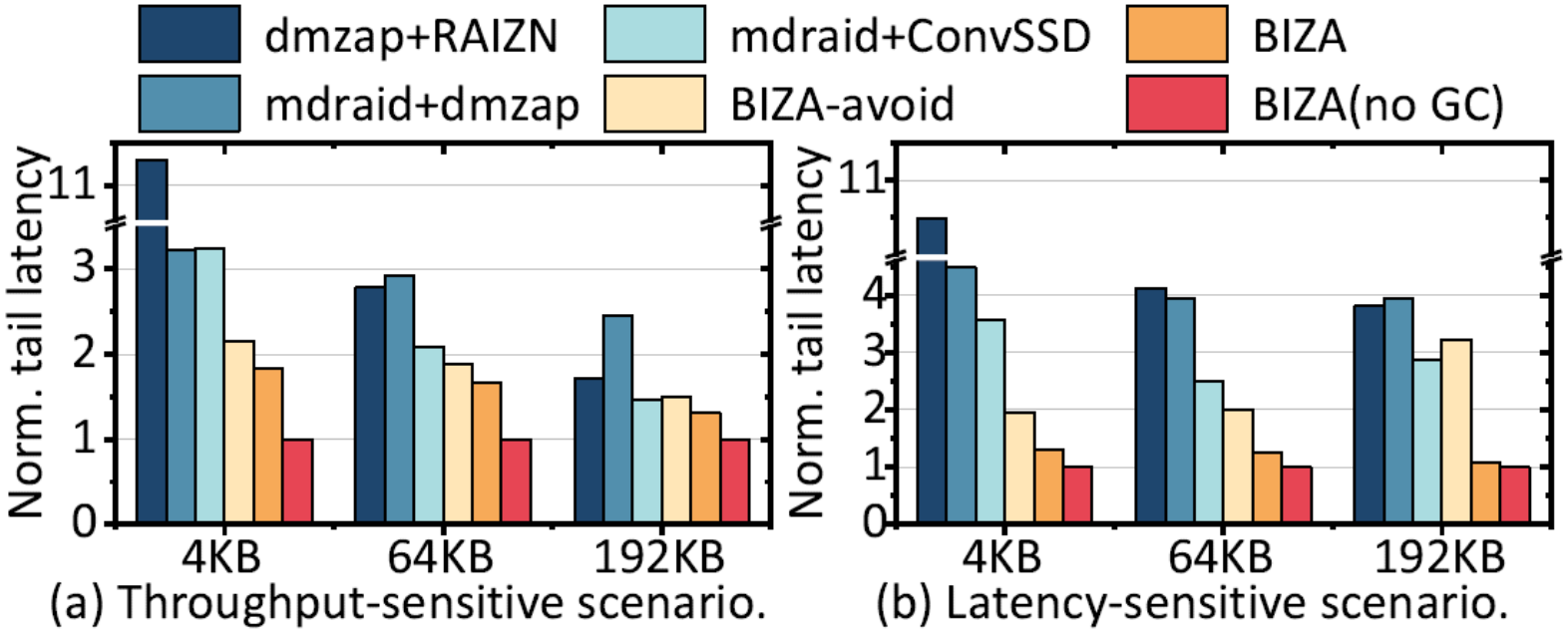
GC-induced latency spike results

- Comparison of tail latency
  - BIZA (no GC): idle
  - BIZA-avoid: turn off the GC avoiding mechanism
  - At most 80% improvement when turn on the avoiding mechanism



GC-induced latency spike results

- Comparison of tail latency
  - BIZA (no GC): idle
  - BIZA-avoid: turn off the GC avoiding mechanism
  - At most 80% improvement when turn on the avoiding mechanism



Conclusion

- BIZA
  - First paper that uses ZRWA
  - Focus on AFA
  - Designed with ZNS characteristics
  - Interesting zone group selector design
  - Channel isolation on large zone ZNS SSD
  - Solid evaluation

  - Cache + storage!

Thanks!
Q & A