eZNS: An Elastic Zoned Namespace for Commodity ZNS SSDs

Jaehong Min, Chenxingyu Zhao, Ming Liu, Arvind Krishnamurthy

USENIX OSDI'23.

2024. 11. 27

Presented by Jeyeon Lee

jeyeonlee@dankook.ac.kr

ZNS SSD

- Divides the LBA space into fixed-sized zones
  - To provide 3 benefits
    - Smaller internal DRAM by maintaining coarse-grained mapping
    - Lower WAF and OP overhead by eliminating the device-side GC
    - I/O bandwidth isolation
  - But the zoned interface is static and inflexible

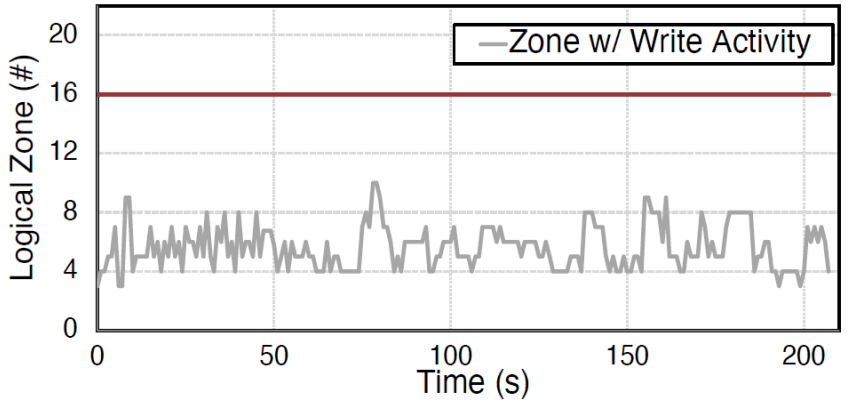


Figure 2: The number of zone with actual write activity when running the *fill-random* workload over the RocksDB. The storage backend is ZenFS. The maximum number of active zones is 16 (red line).

Performance Characterization

- Experimental Setup & System model
  - Logical zone: a group of physical zones

Device HW Parameters	Specification
Capacity	3,816 GB
Channels #	16 Channels
NAND Dies #	128 Dies
NAND Page Size	16 KB
NAND Channel B/W	~600 MB/s
Physical Zone Size	96 MB
Read B/W per Physical Zone	~200 MB/s
Write B/W per Physical Zone	~ 40 MB/s
Maximum Active Zones #	256

Table 1: The commodity ZNS SSD specification.

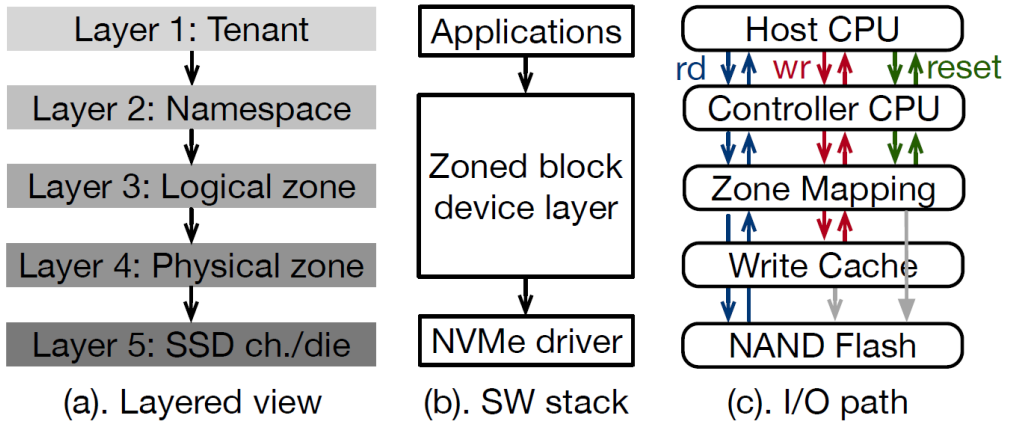


Figure 3: System model, SW stack, and I/O path of a multi-tenant ZNS SSD deployment. RD/WR=Read/Write. The write cache flushes data to the NAND flash asynchronously. Zone resets are completed after invalidating the mapping layer, where NAND blocks are erased lazily.

Performance Characterization: Zone Striping

- Is applied to achieve higher throughput
  - Two configuration parameters
    - Stripe size: the size of unit in a stripe
    - Stripe width: the number of stripes(zones)

→ Small ← Large

Hurts the device I/O efficiency

Underutilizes zones

Stripe Size	Avg. Lat(us)	P99.9 Lat. (us)	B/W (MB/s)
4KB	64	76	59
8KB	71	84	108
16KB	88	103	175
32KB	163	269	190
64KB	314	619	198

Table 2: Read I/O average/P99.9 latency and bandwidth varying the stripe size on a physical zone.

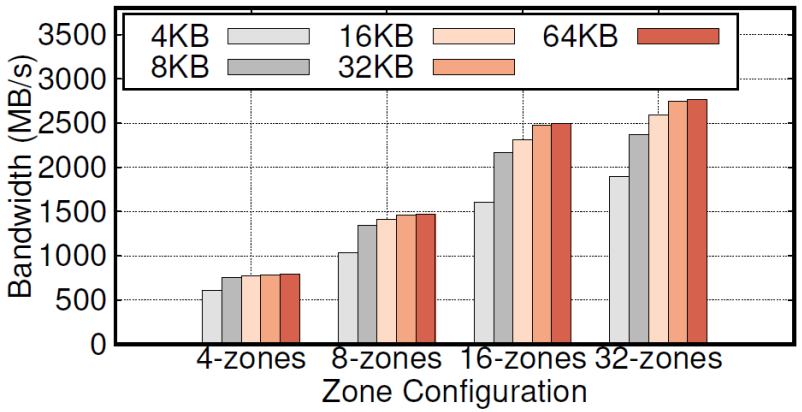


Figure 4: Read bandwidth varying the stripe size for different types of zones.

Performance Characterization: Zone Allocation

- Should be locality-aware and parallelism-aware
  - Two types of inefficient placements
    - Channel-overlapped placement
    - Die-overlapped placement

→ It's challenging to infer the zone's physical location

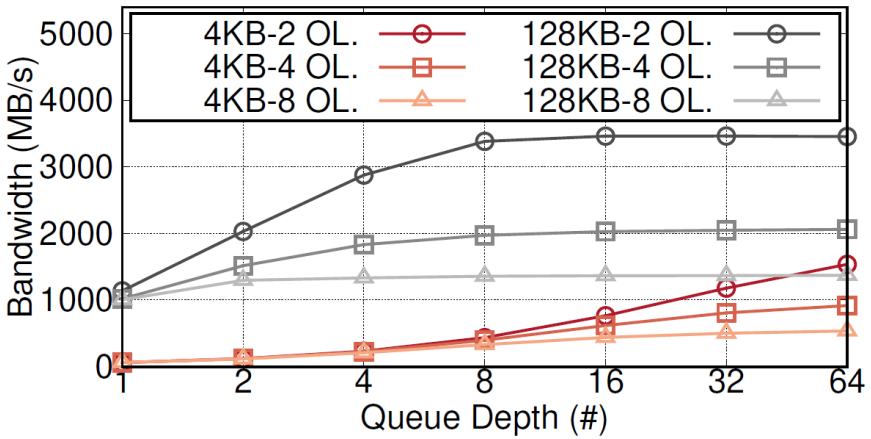


Figure 6: Read bandwidth under three channel overlapping (OL) allocations.

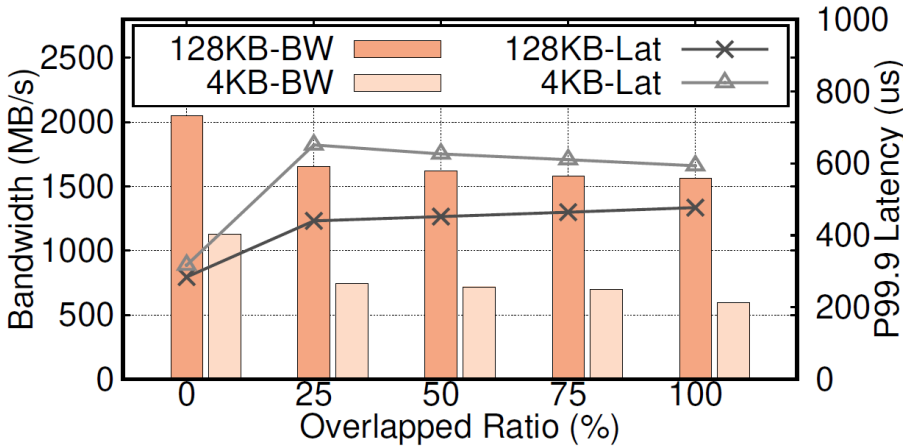


Figure 7: Bandwidth and tail latency varying with the die overlapping ratio.

Performance Characterization: I/O Execution

- of Multi-tenants
  - Frag: a conv-SSD preconditioned by filling 70% with 128KB random writes
  - Zone A(qd 8, 2-zone), Zone B(qd 2, 8-zone)
  - Tenants: reader(128KB random), writer(sequential)

→ Should employ a global congestion avoidance scheme

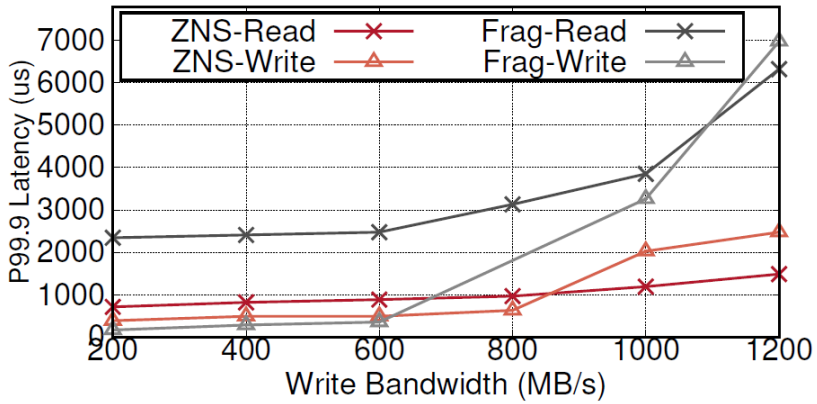


Figure 8: Read tail latency varying the write bandwidth (ZNS vs Conventional SSD)

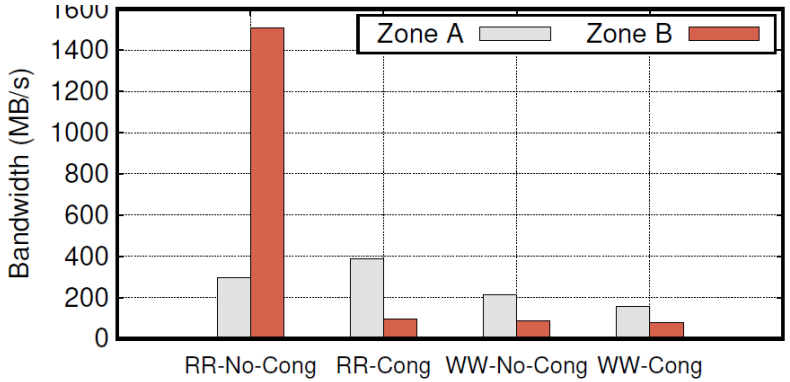


Figure 9: Bandwidth under RD-RD and WR-WR congestion due to the die-collision.

eZNS: Enabling an Adaptive Zoned NS

- Overall system architecture
  - Zone Arbiter
    - HAL
    - Serial zone allocator
    - Zone ballooning
  - Zone I/O Scheduler
  - v-zone(a specialized logical zone)
    - Runtime hardware adaptiveness
    - Application elasticity
    - Tenant awareness

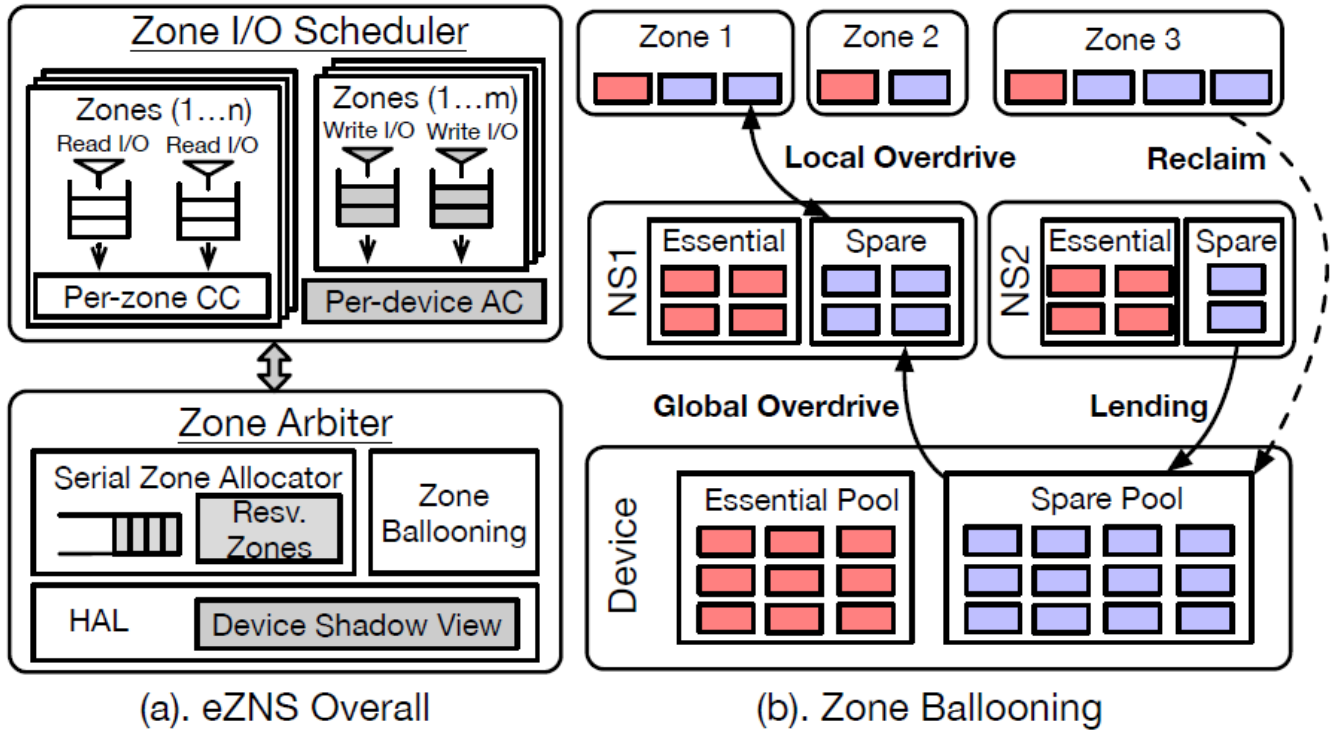


Figure 10: eZNS System Architecture.

Zone Ballooning

- Automatically scale the I/O striping configuration of v-zones
  - Overprovisioning
    - Essential group: a minimal number of physical zones for maximize SSD bandwidth
    - Spare group: max active zones – # of zone in the essential group
  - Expanding
    - Local overdrive: expand stripe width through active zone history
    - Global overdrive: Reallocates spare zones across namespaces
  - Reclaiming
    - Migrate to a new stripe group with shrunk width
    - If a namespace presents no write I/Os

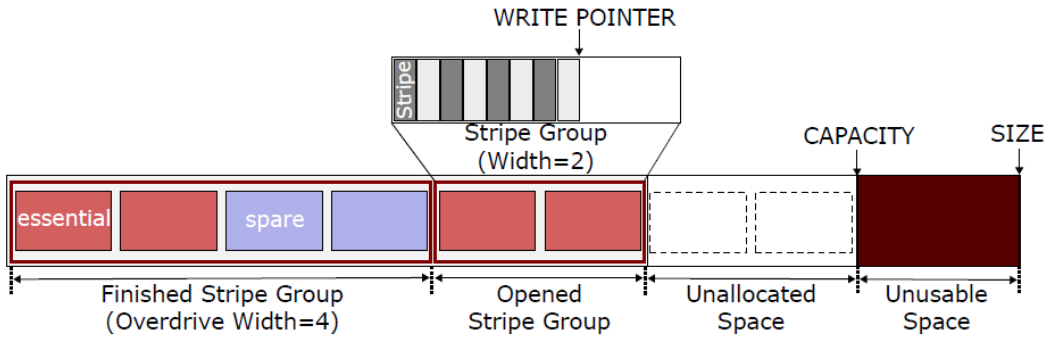


Figure 11: Example of eZNS *v-zone* structure.

Zone I/O Scheduler

- Maximize the overall device utilization
  - Congestion-avoid read scheduler(for read)
    - Detect congestion via latency
    - Congestion window (1 to 4 x stripe width)
  - Cache-aware write admission control(for write)
    - Write congestion happens globally
    - Token-based write admission

Algorithm 1 Zone I/O Scheduler

1: **procedure** READ COMPLETION()
2: $lat\_thresh \leftarrow 500us$
3: **if** $io\_lat > lat\_thresh$ **then**
4: $cwnd = \max(1, cwnd \times \frac{lat\_threash}{2 \times io\_lat})$
5: **else** ▷ α = additive factor
6: $cwnd = \min(stripe\_width \times 4, cwnd + \alpha \times \frac{io\_count}{cwnd})$
7: **procedure** WRITE LATENCY MONITOR()
8: On t every 10ms
9: $total\_lat = \sum_{active\_zone} per\_block\_lat$
10: $total\_ios = \sum_{active\_zone} num\_ios$
11: $avg\_lat(t) = \frac{total\_lat}{total\_ios}$
12: $block\_admission\_rate = \frac{avg\_lat(t-1)+avg\_lat(t)}{2}$
13: **procedure** WRITE TOKEN GENERATOR()
14: On every 1ms
15: **for** pending write zones **do**
16: $token \mathrel{+}= \frac{now-last\_refill}{block\_admission\_rate} \times stripe\_width$

Evaluation

- Zone Ballooning
  - Configuration
    - 4 namespaces, each of which is allocated 32 essential and 32 spare zones (16 active zones)
    - NS 1,2,3: stop issuing writes from t=30s to t=80s

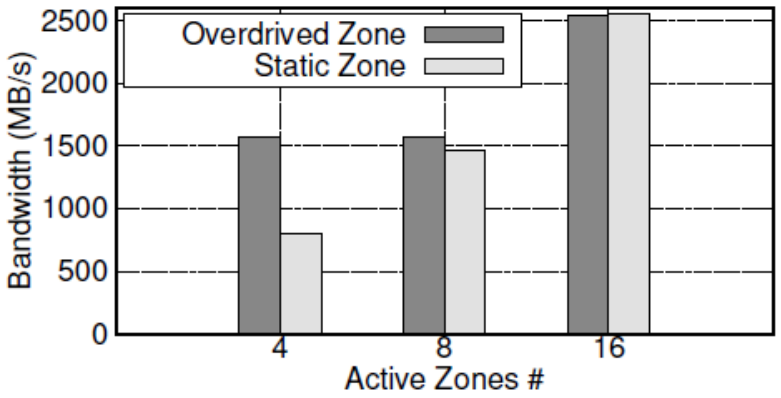


Figure 12: B/W comparison between an over-drived and three statically configured zones.

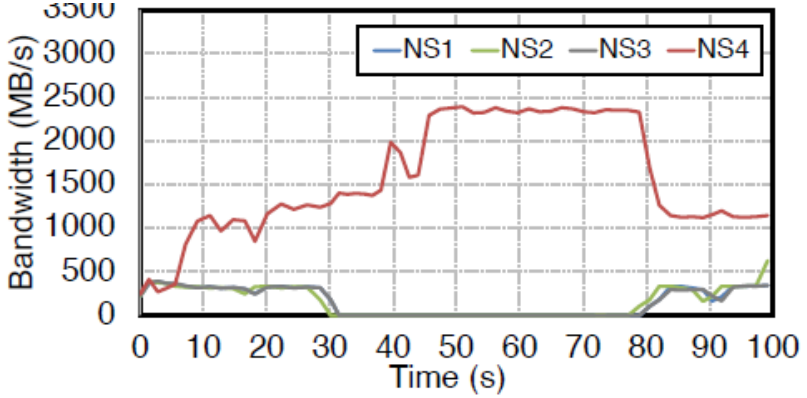


Figure 13: Performance variation of four namespaces with global overdrive under 100s.

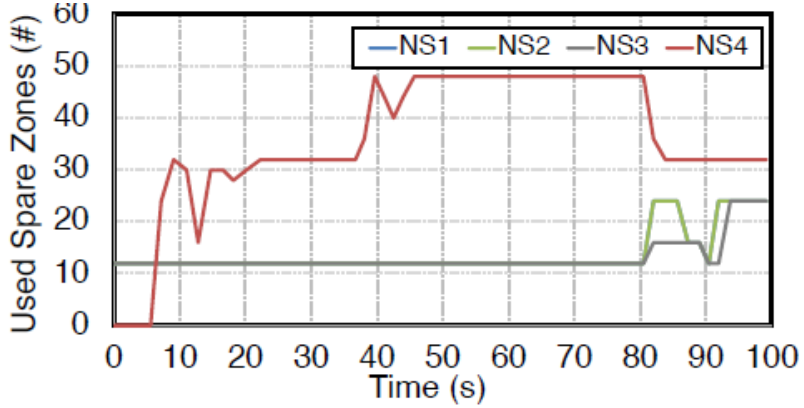
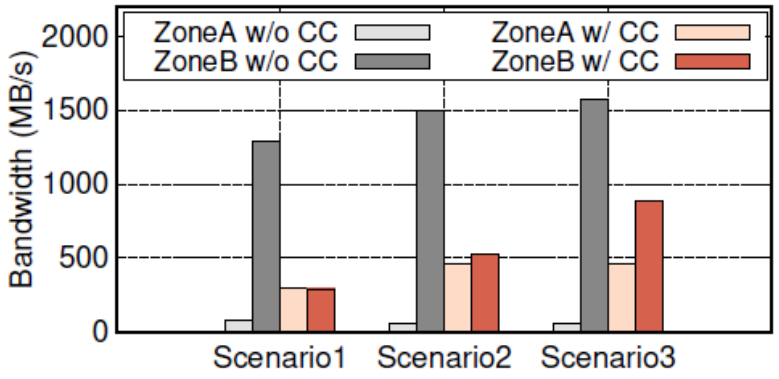
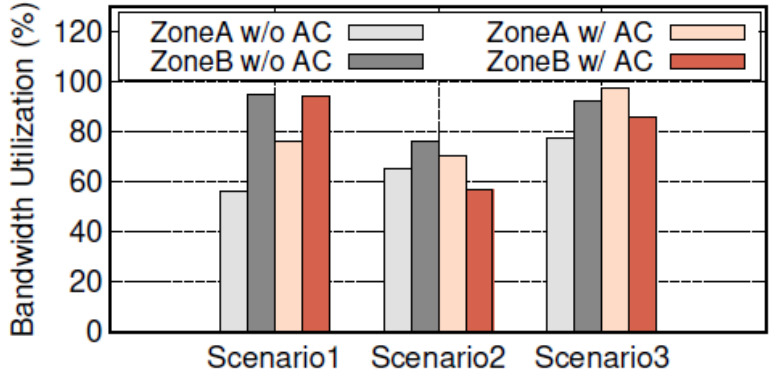


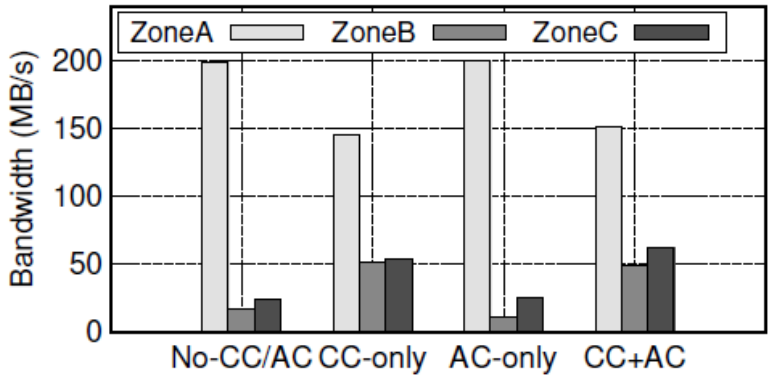
Figure 14: The number of used spare zones of four namespaces under 100s.

Evaluation

- Zone I/O Fairness







(a) Read-Read Fairness. (128KB Read. Zone A with QD-1, and Zone B with QD-32)

(b) Write-Write Fairness. (Zone A for regular writers, and Zone B for the busy writer)

(c) Read-Write Fairness. (Zone A for readers, Zone B for the busy writer, and Zone C for regular writers)

Figure 15: Efficiency of eZNS on handling read-read, write-write, and read-write congestion. (CC=Congestion Control, AC=Admission Control)

1) Zone A : (Stripe width : 2, QD 1, 32KB)
   Zone B : (Stripe width : 8, QD 32, 8KB)
2) Zone A : (Stripe width : 4, QD 1, 16KB)
   Zone B : (Stripe width : 8, QD 1, 8KB)
3) Zone A : (Stripe width : 1, QD 1, 128KB)
   Zone B : (Stripe width : 8, QD 1, 8KB)

1) Zone A : (Stripe width : 8, QD 1, 8KB, 5ms intervals)
   Zone B : (Stripe width : 2, QD 1, 32KB)
2) Zone A : (Stripe width : 8, QD 1, 8KB)
   Zone B : (Stripe width : 2, QD 1, 32KB)
3) Zone A : (Stripe width : 8, QD 1, 8KB)
   Zone B : (Stripe width : 2, QD 1, 32KB)
   No overlap die

Zone A : Read, Stripe width : 2, 128 KB
Zone B : Write, Stripe width : 2, 32KB
Zone C : Write, Stripe width : 8, 32KB, 5ms interval

Evaluation

- Application: RocksDB

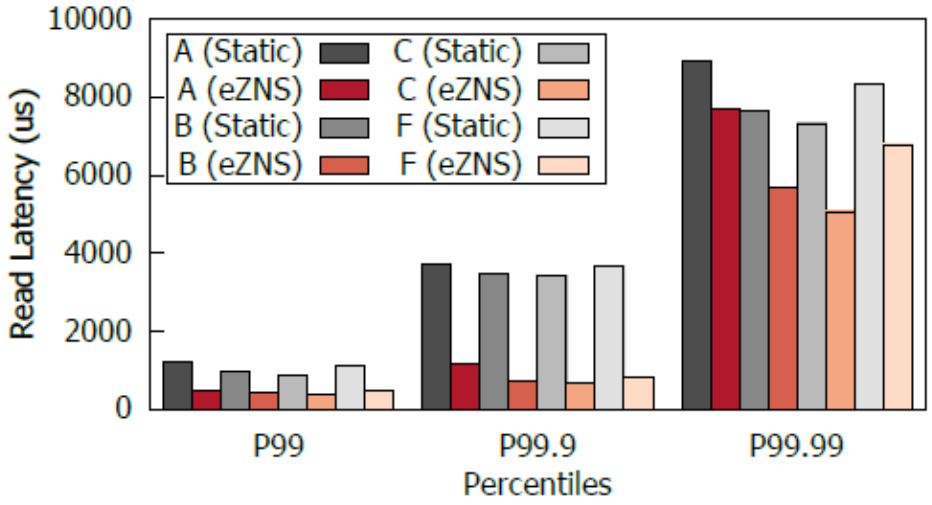


Figure 19: Read latency of YCSB workloads (A/B/C/F) on different namespaces over eZNS and static zone.

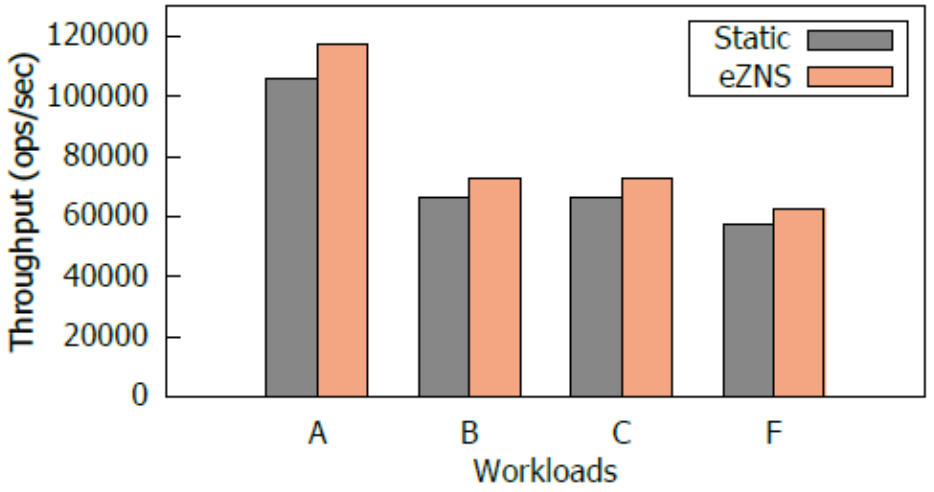


Figure 20: Throughput of YCSB workloads (A/B/C/F) on different namespaces over eZNS and static zone.

eZNS: An Elastic Zoned Namespace for Commodity ZNS SSDs

Jaehong Min, Chenxingyu Zhao, Ming Liu, Arvind Krishnamurthy

USENIX OSDI'23.

Thank You !

2024. 11. 27

Presented by Jeyeon Lee

jeyeonlee@dankook.ac.kr