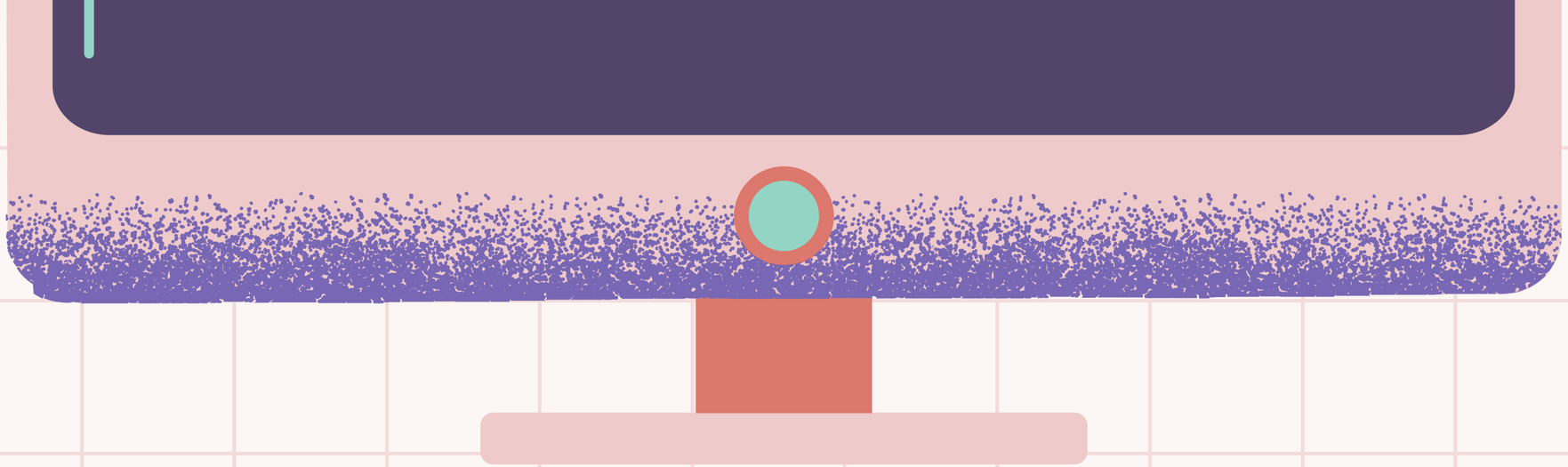


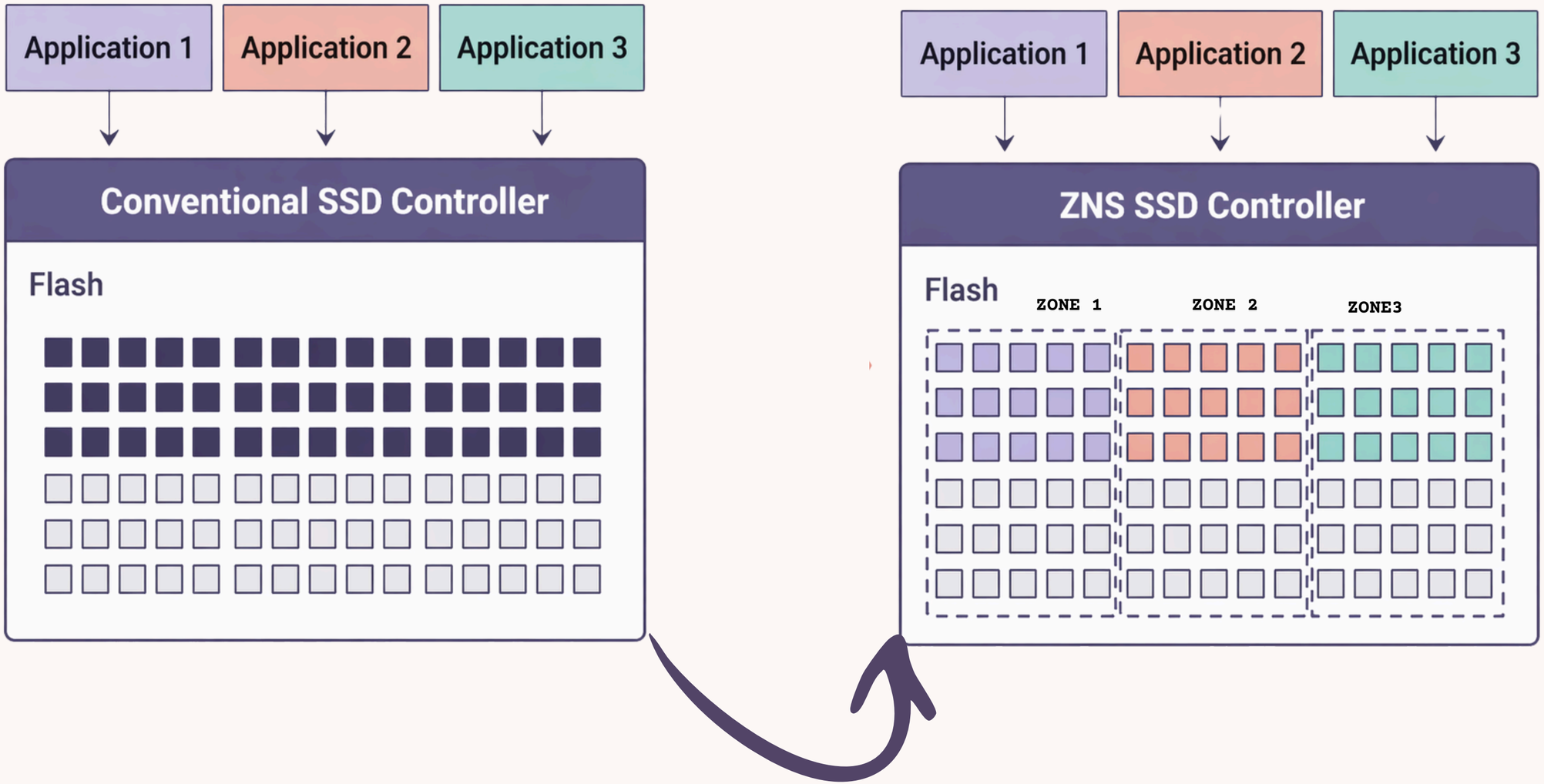
Analyzing Zone Sizes in ZNS SSDs

Chris, Nandana, Vishakha

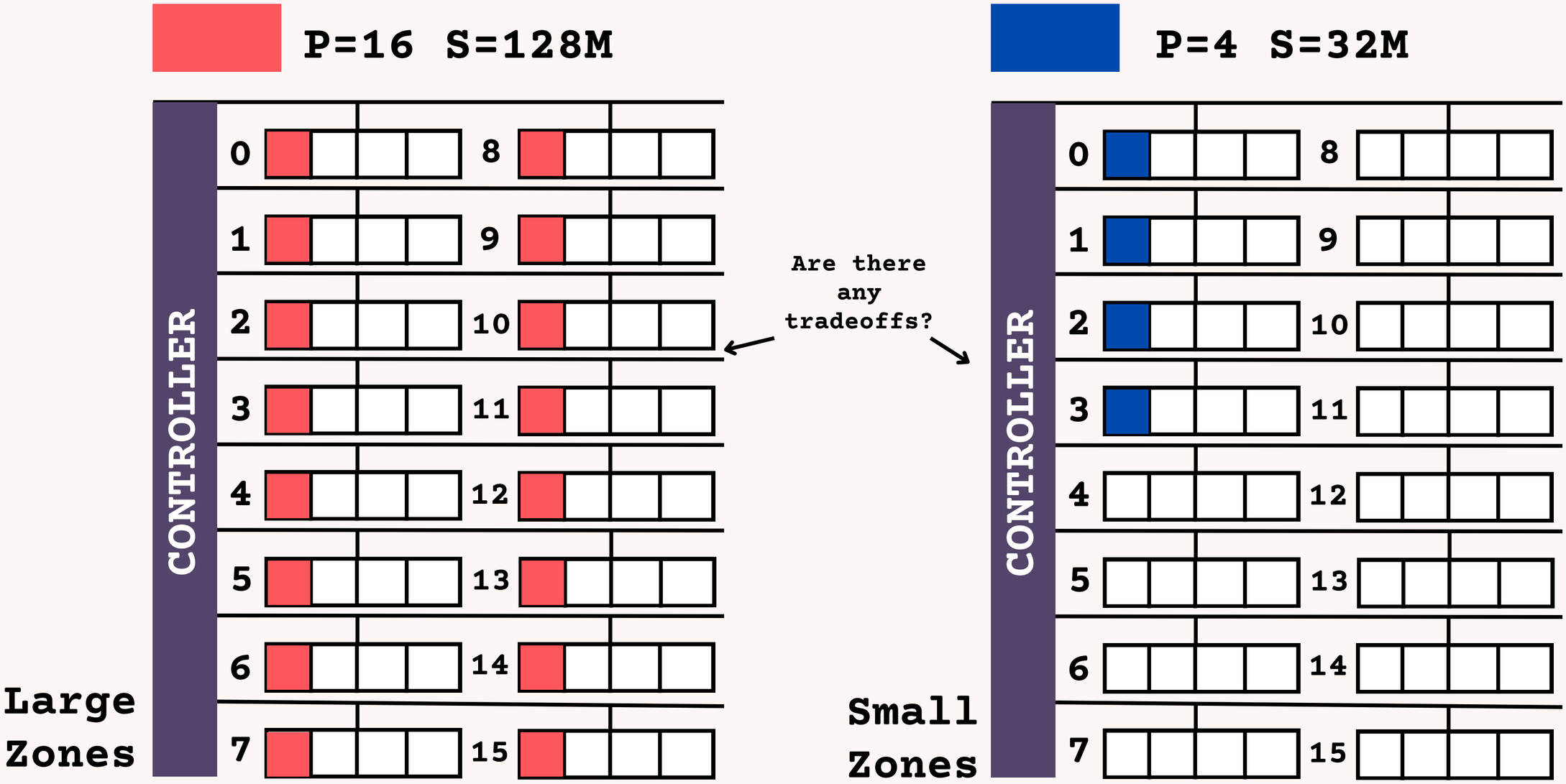
CS561

What is ZNS?

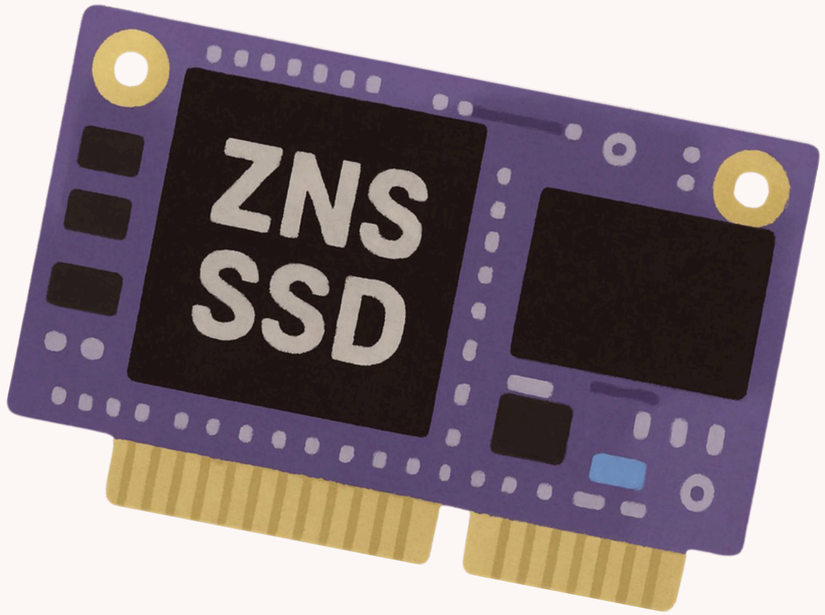
How it differs from traditional SSDs?



MOTIVATION


P=16 S=128M
P=4 S=32M
CONTROLLER
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Are there any tradeoffs?
Large Zones
Small Zones

System Description: ConfZNS++



ConfZNS++ builds on a QEMU-based emulator, acting as a drop-in replacement for a real SSD while giving the host full control over I/O management—making its behavior both realistic and trustworthy for evaluation.

Experiments

What we evaluated?

Preliminary Experiments:
Do initial experiments for intra zone, inter zone and impact of FINISH and RESET operations

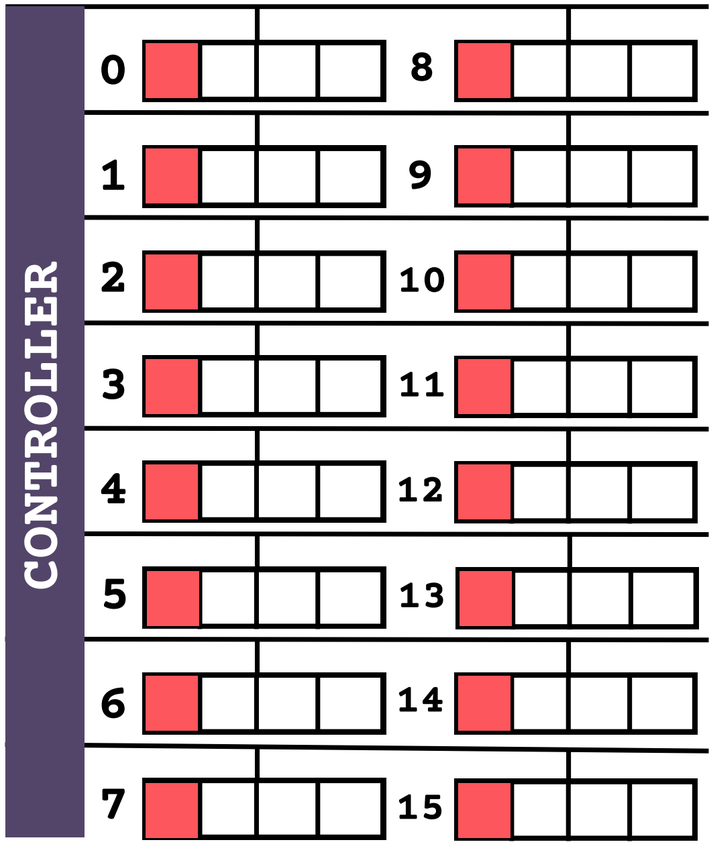
Device Level Benchmarking:
Inter Zone, Intra Zone, Zone Reset Cost under Prior Write History

Application Level Benchmarking:
Evaluate latency, space amplification and write amplification for different zone size configurations using RocksDB and KVbench with ZenFS

Configurations

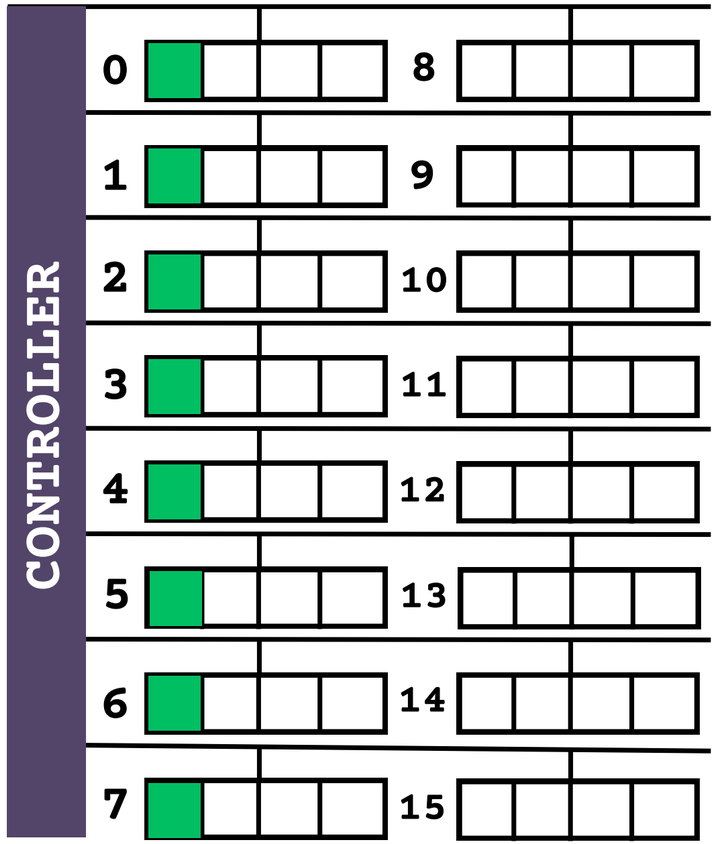
zns_channels_per_zone=8
zns_ways_per_zone=2

P=16 S=128M



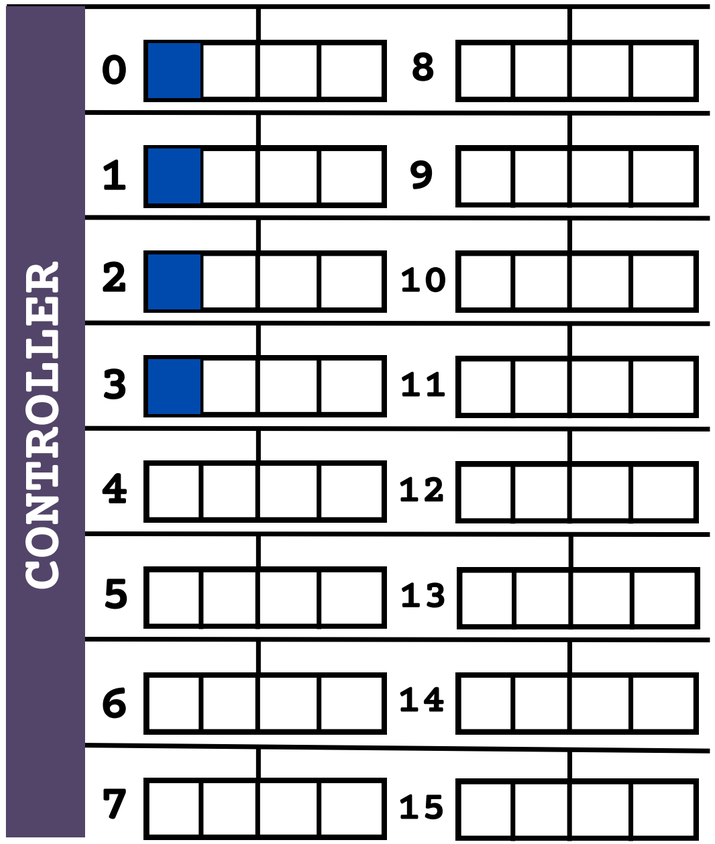
zns_channels_per_zone=8
zns_ways_per_zone=1

P=8 S=64M



zns_channels_per_zone=4
zns_ways_per_zone=1

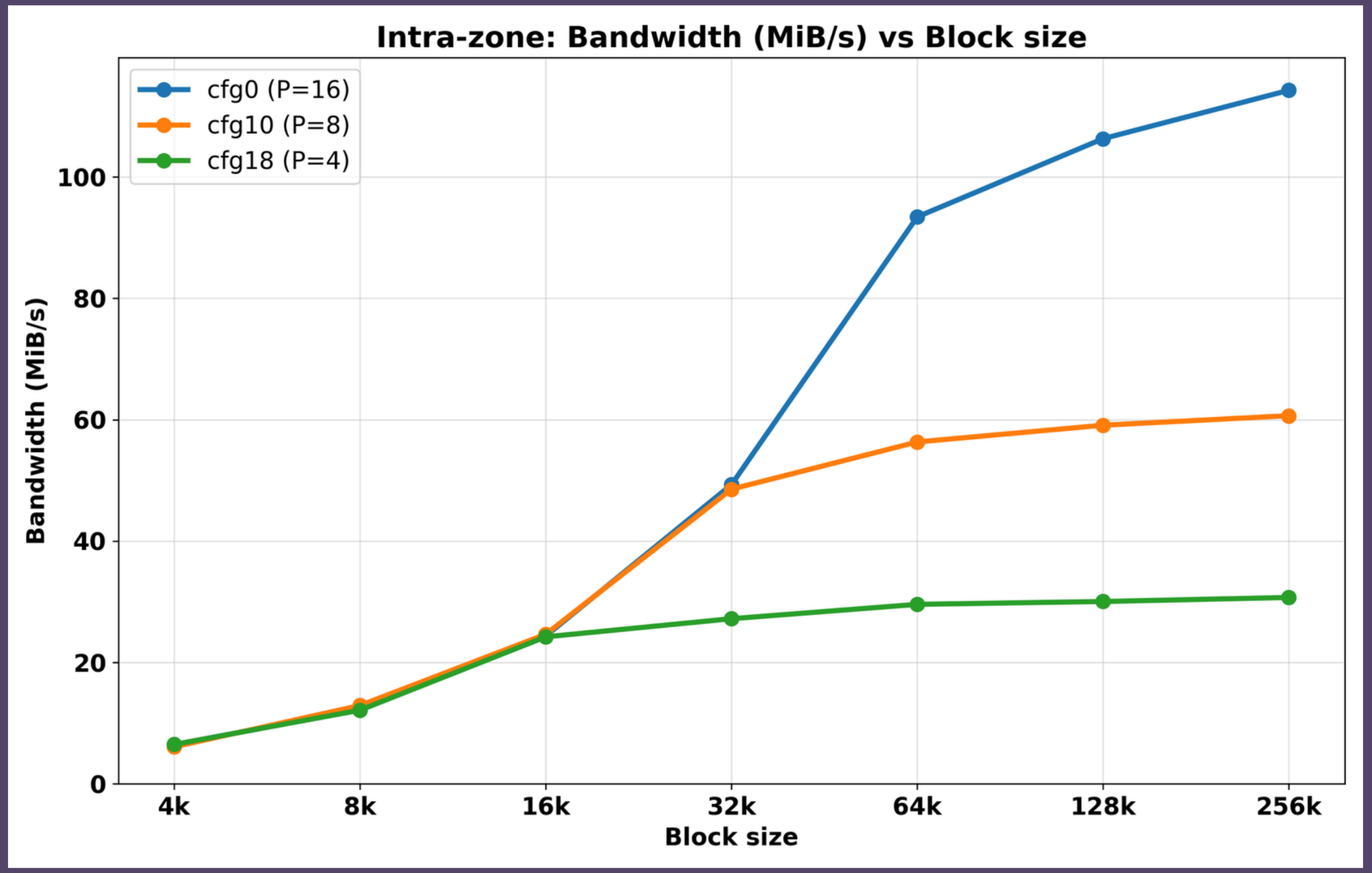
P=4 S=32M



Device Level Benchmarking : FIO & Resets

Intra Zone Experiment

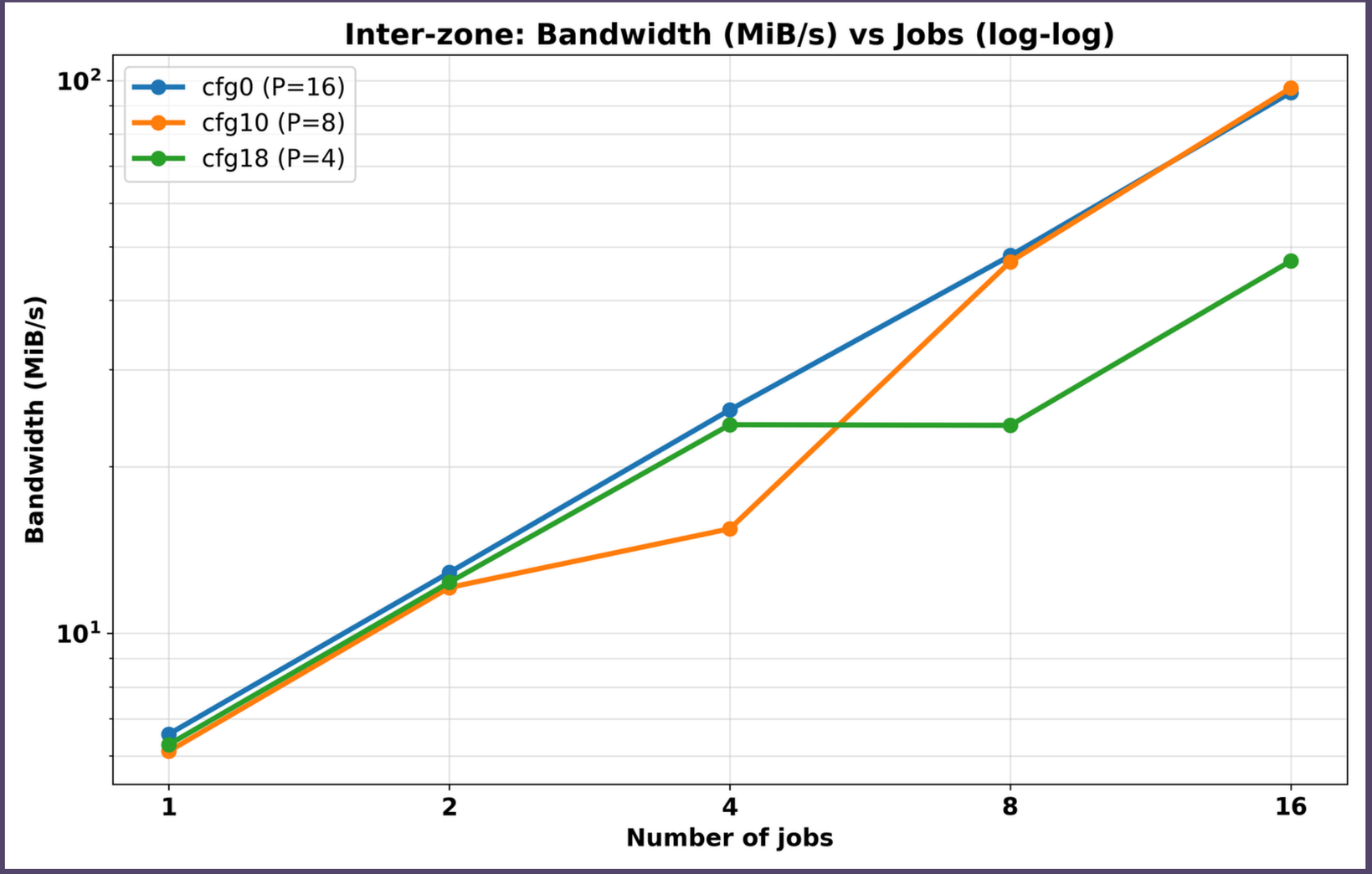
stripe size = channels × ways × page size



Performance scales with the amount of internal parallelism, and shows clear jumps when the stripe size is fully utilized.

Inter Zone Experiment

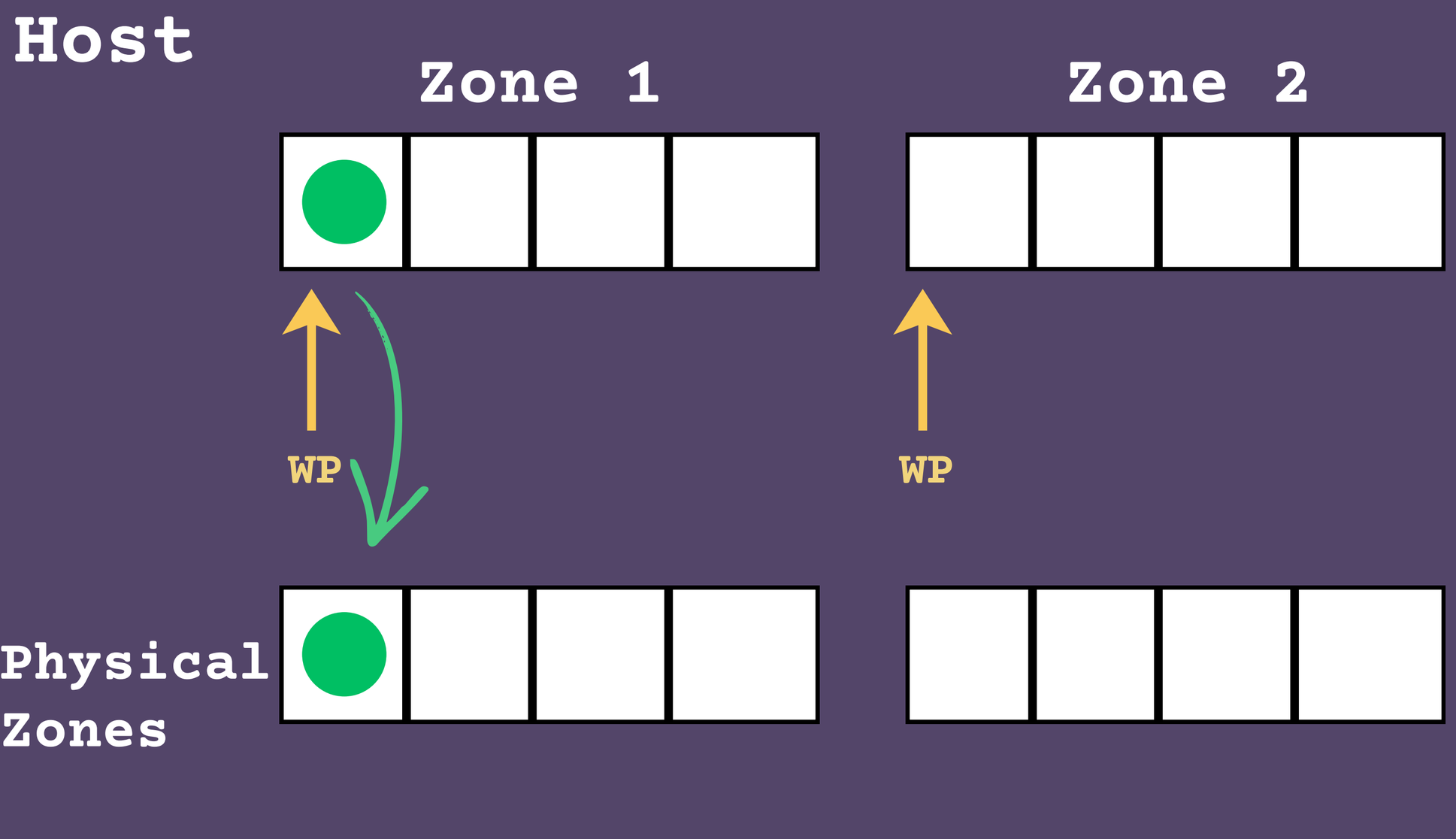
```
zone_concurrency = (nchnls / chnls_per_zone) * (ways / ways_per_zone);
```



Inter-zone parallelism is not unlimited, it is strictly bounded by zone_concurrency, which is determined by SSD geometry

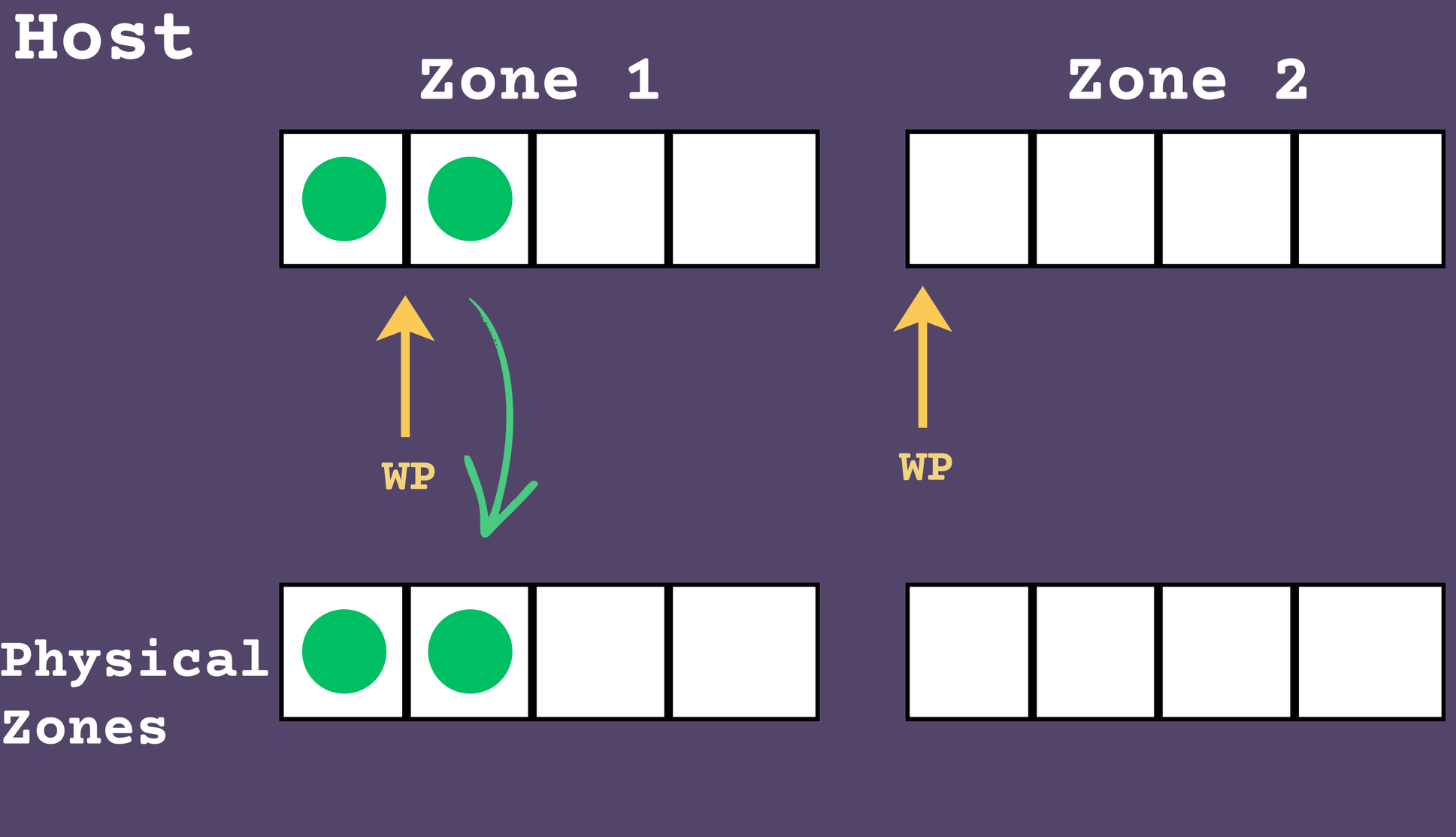
Zone Resets

WP → write pointer



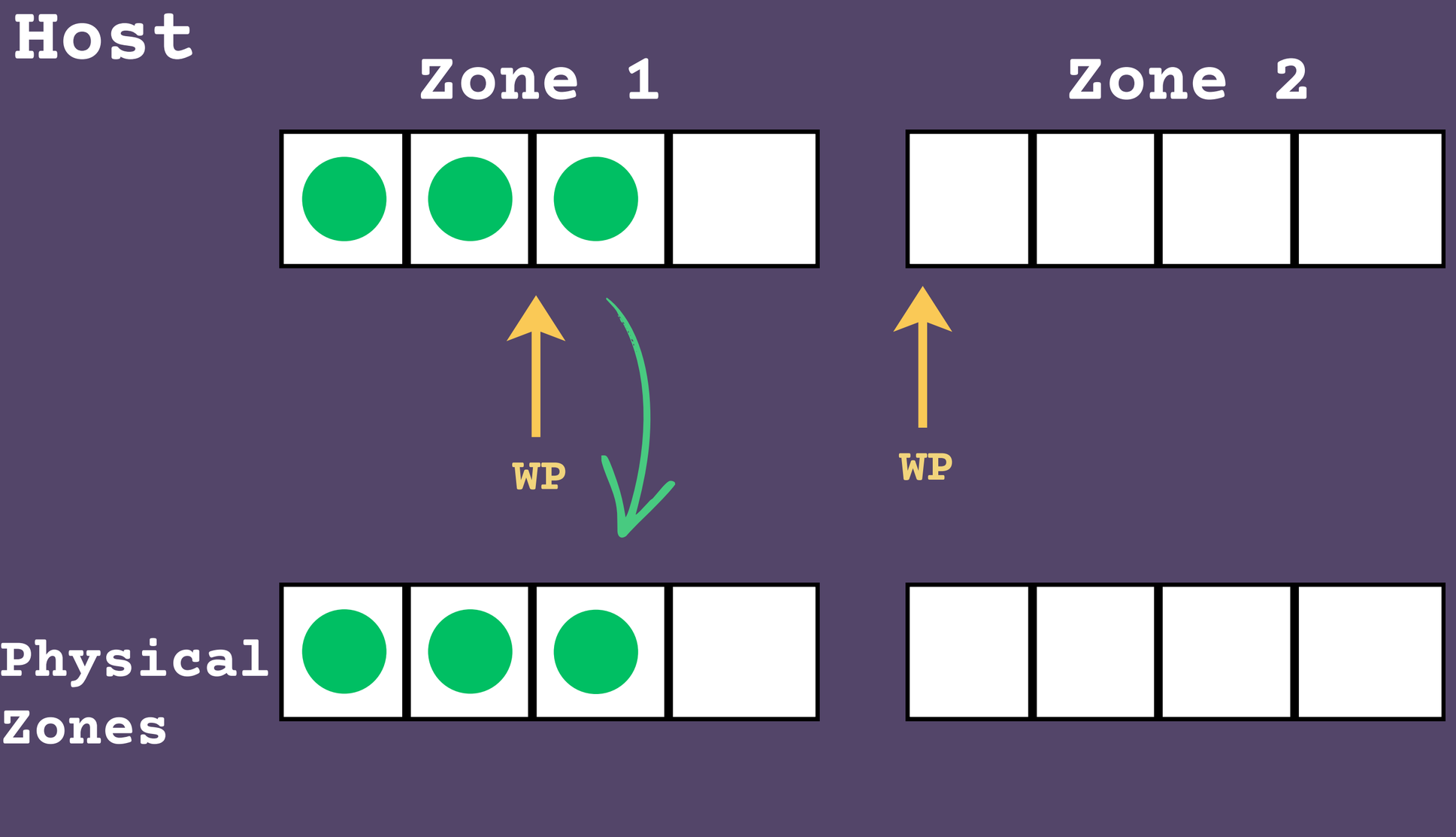
Zone Resets

WP → write pointer



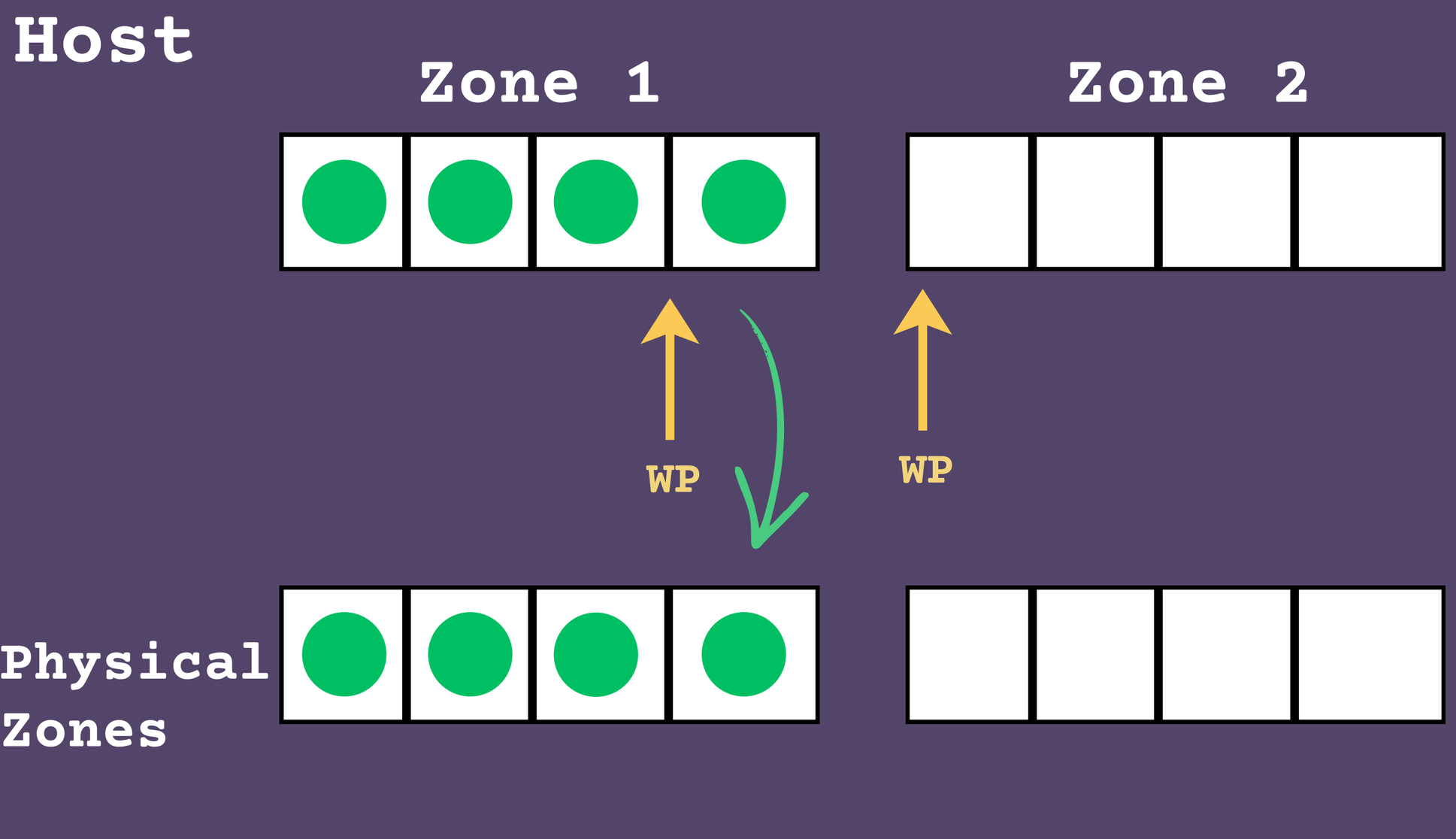
Zone Resets

WP → write pointer



Zone Resets

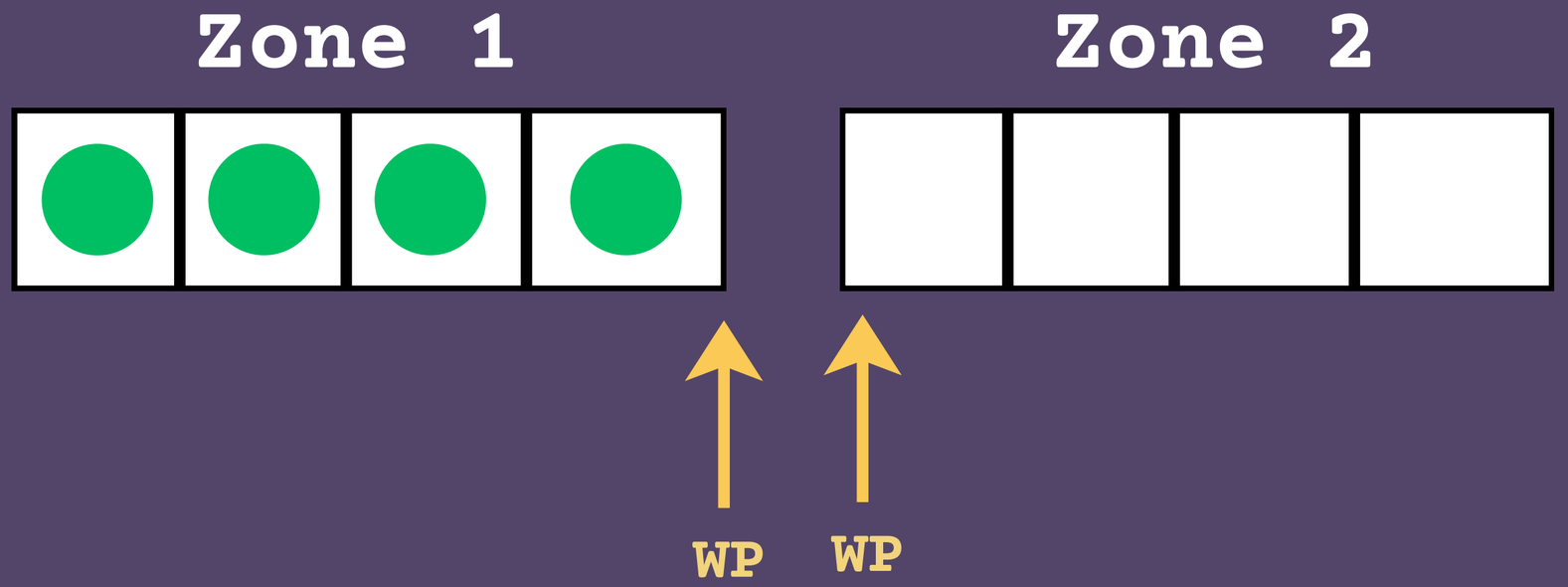
WP → write pointer



Zone Resets

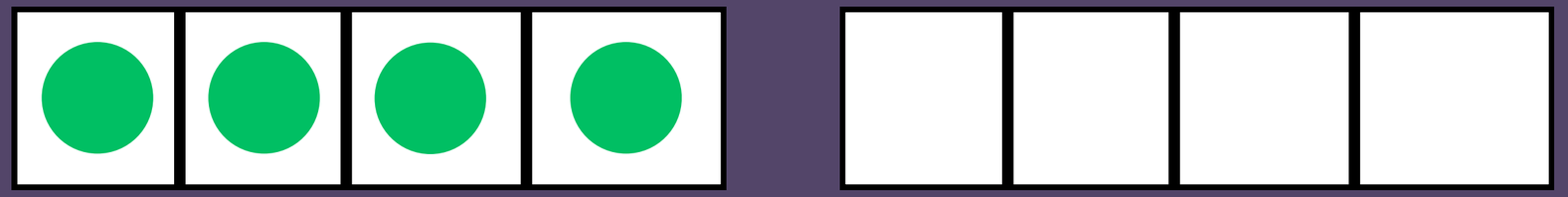
WP → write pointer

Host



ZONE RESET!

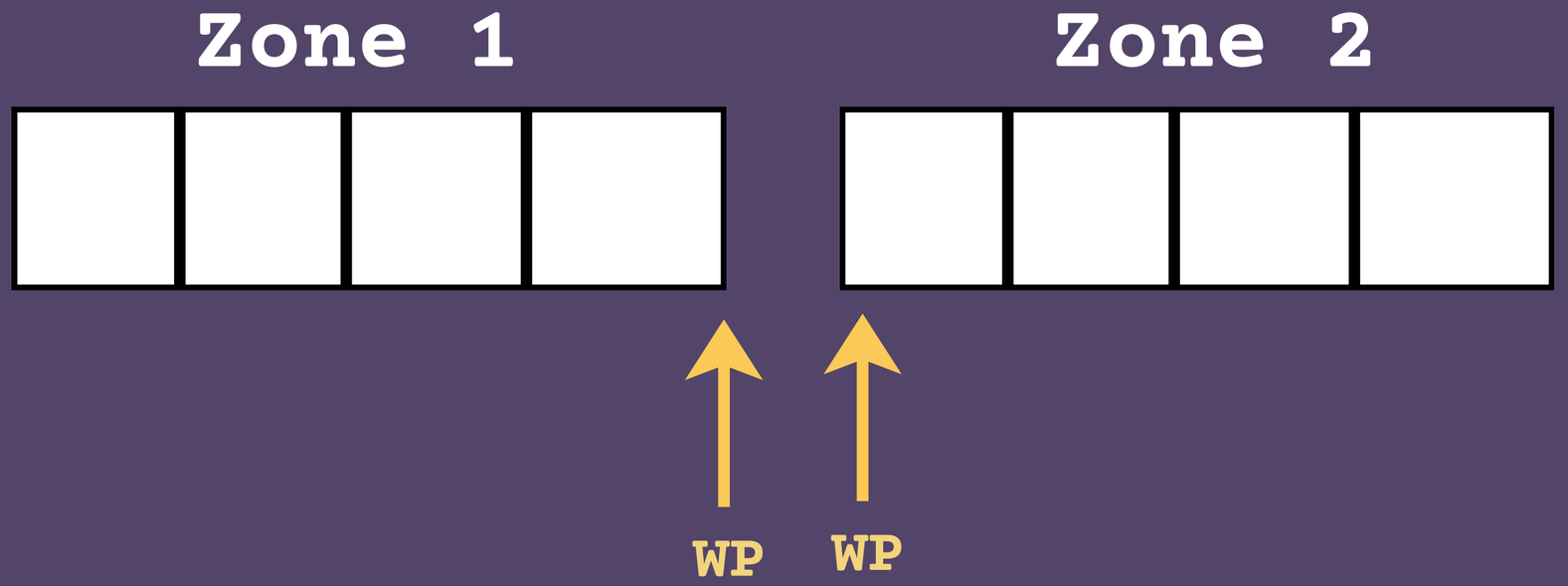
Physical Zones



Zone Resets

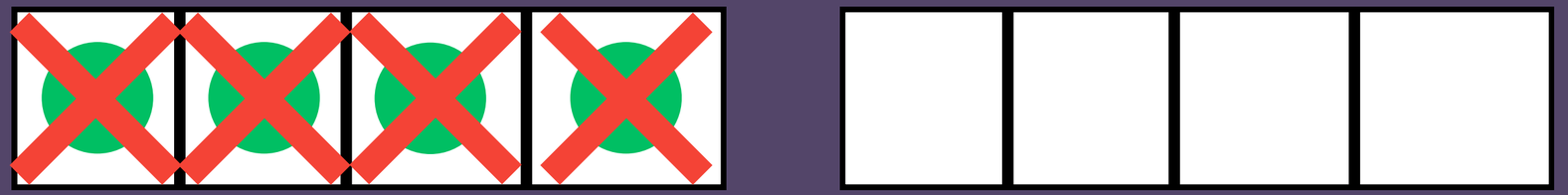
WP → write pointer

Host



ZONE RESET!

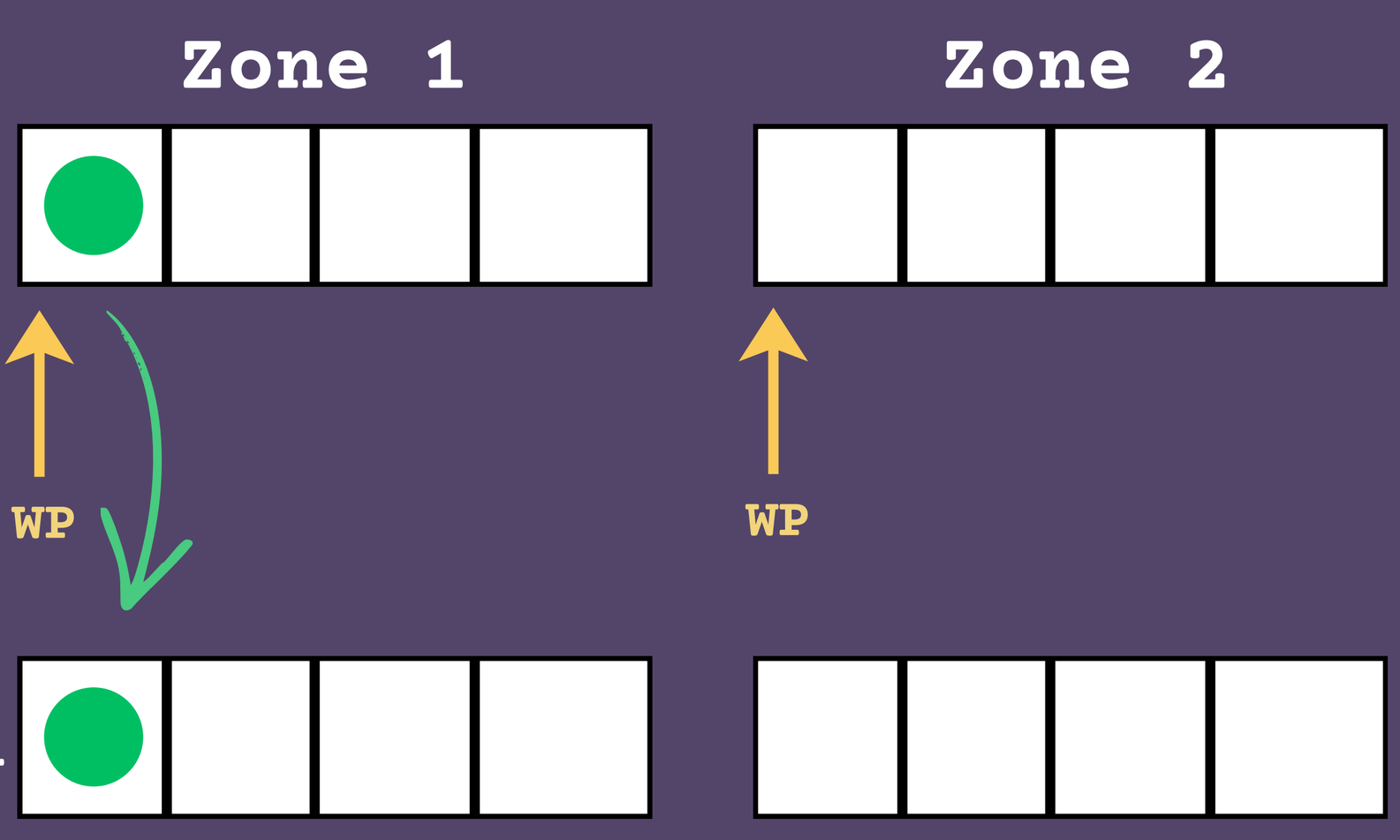
Physical Zones



Zone Resets

CASE 1

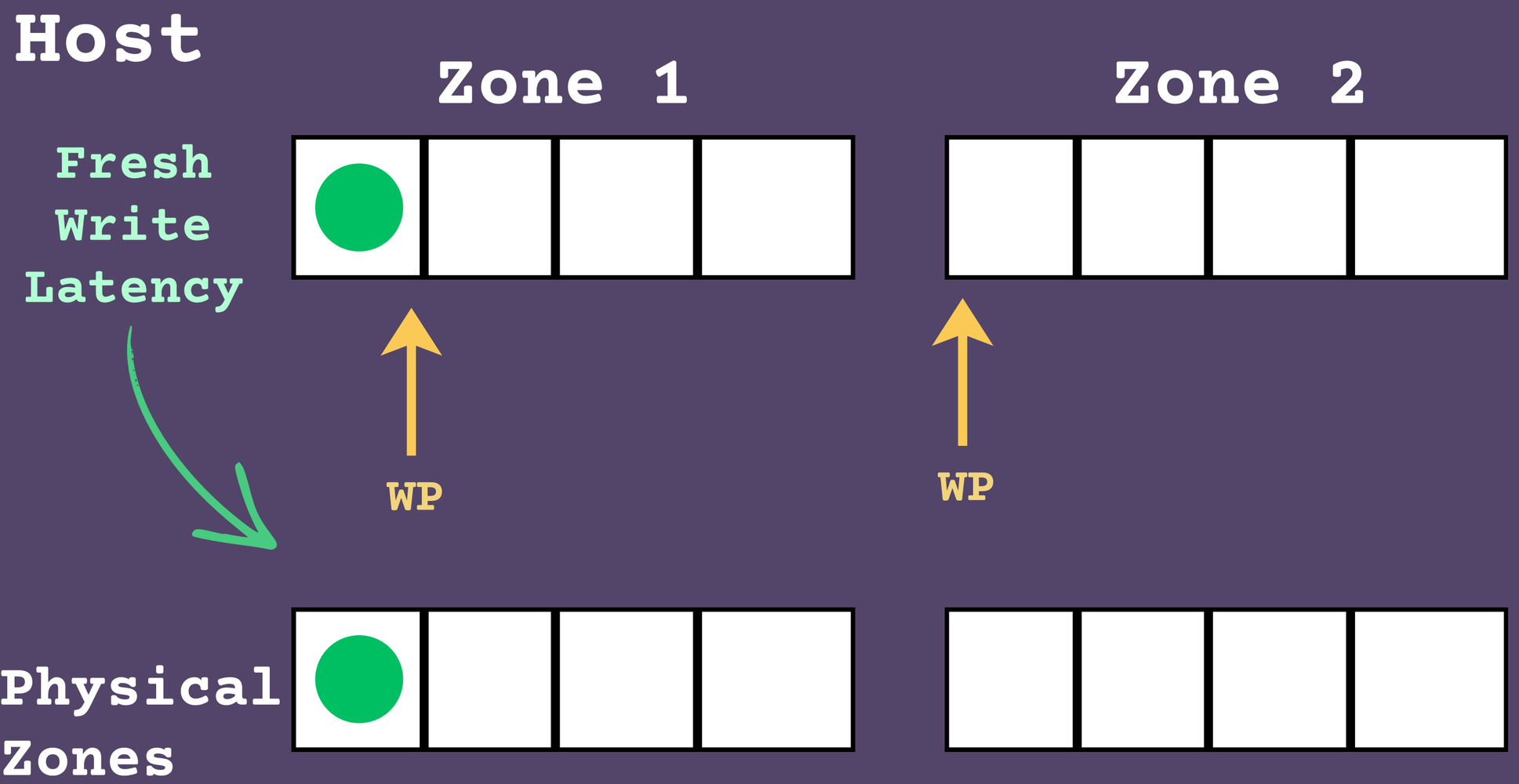
WP → write pointer



Zone Resets

CASE 1

WP → write pointer

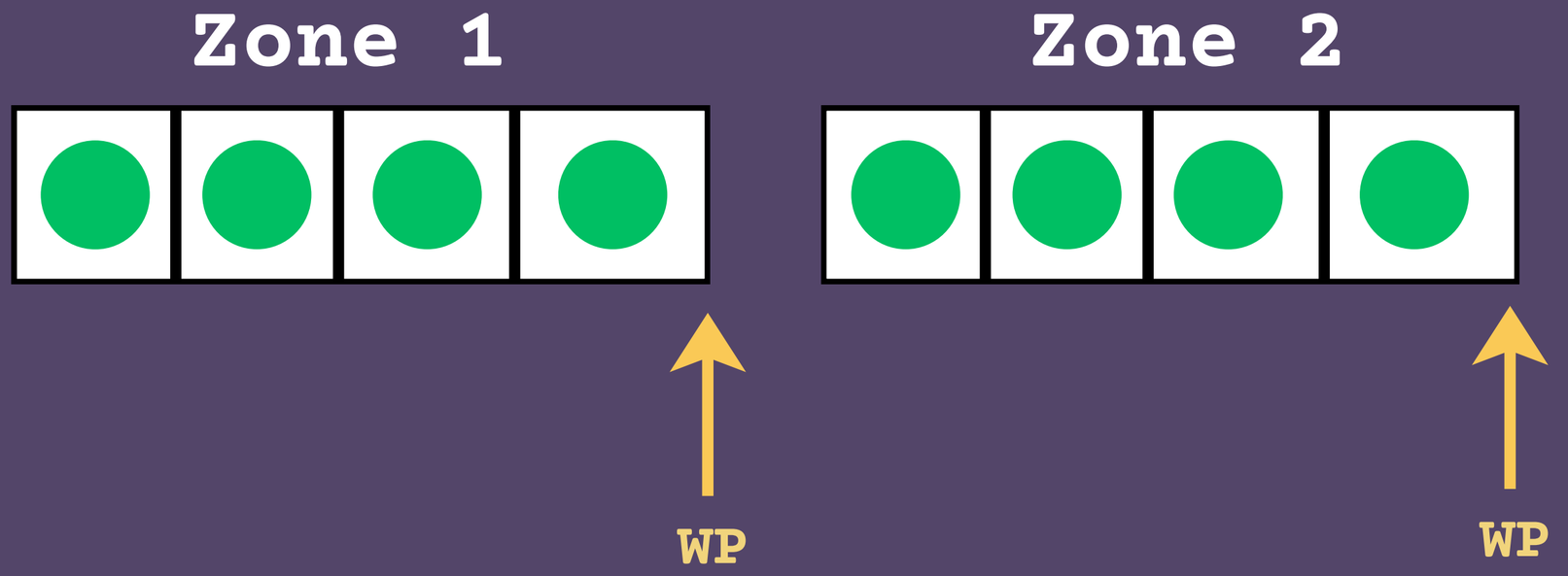


Zone Resets

CASE 2

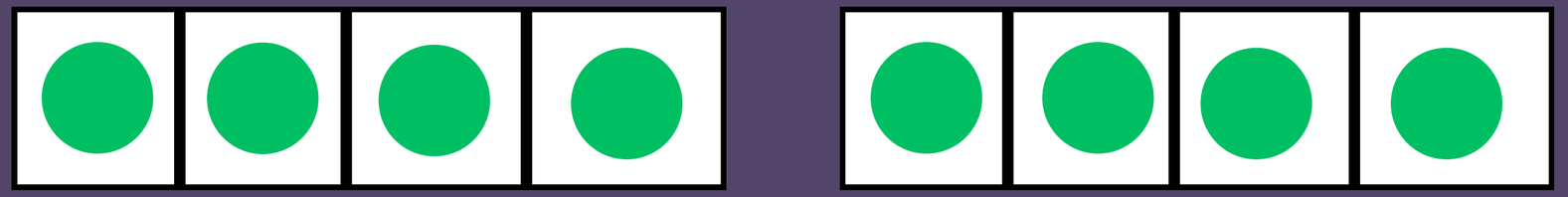
WP → write pointer

Host



ZONE RESET ZONE 1

Physical Zones

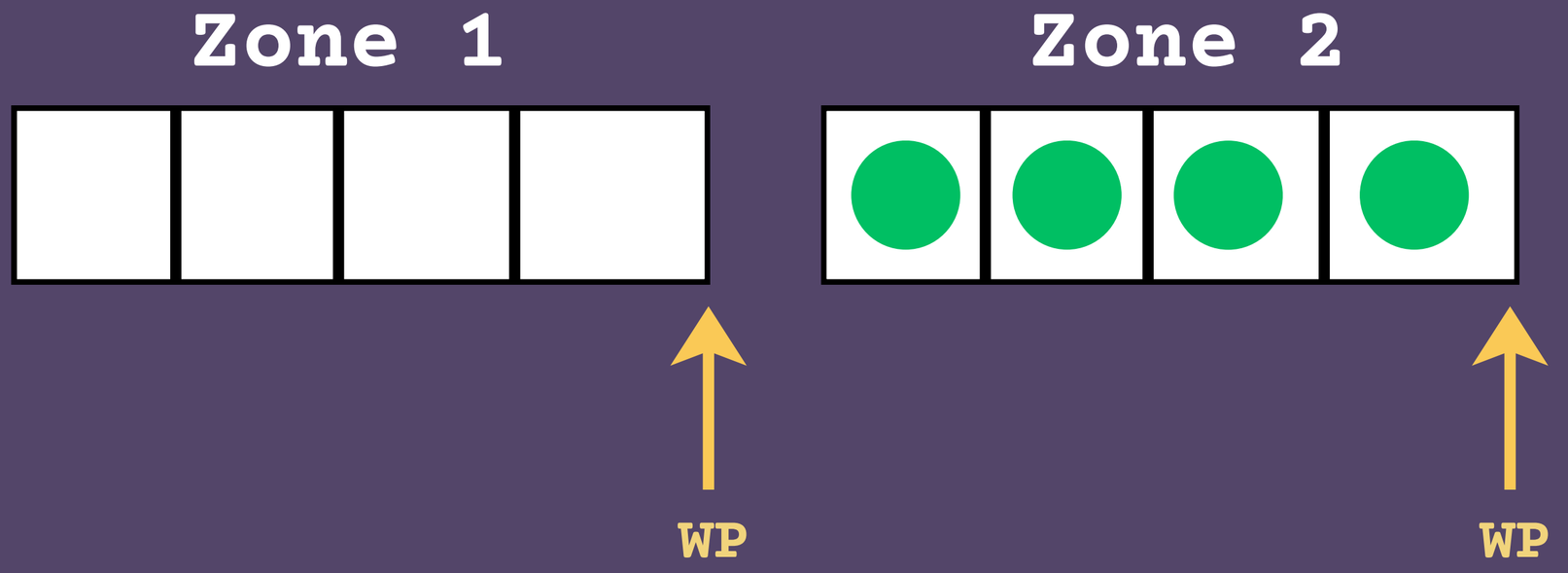


Zone Resets

CASE 2

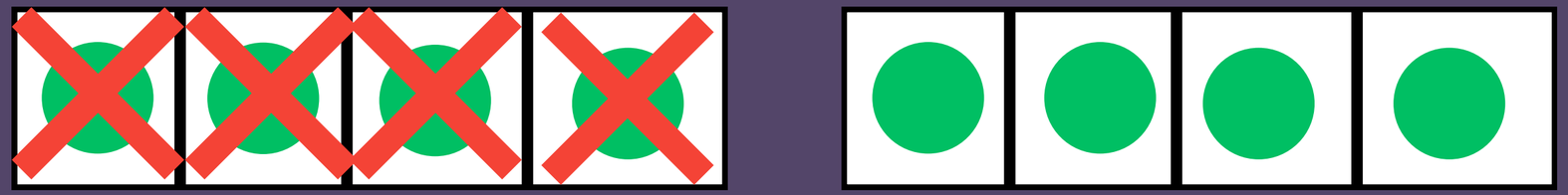
WP → write pointer

Host



ZONE RESET ZONE 1

Physical Zones

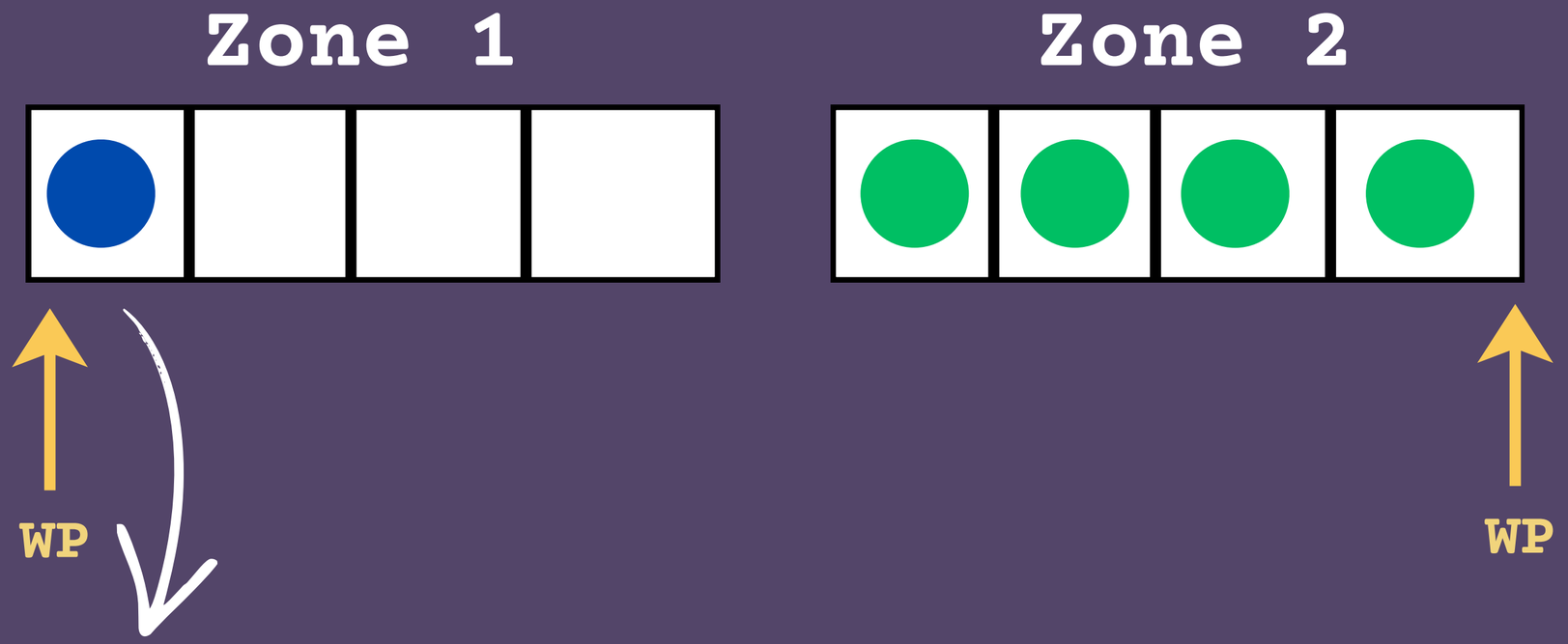


Zone Resets

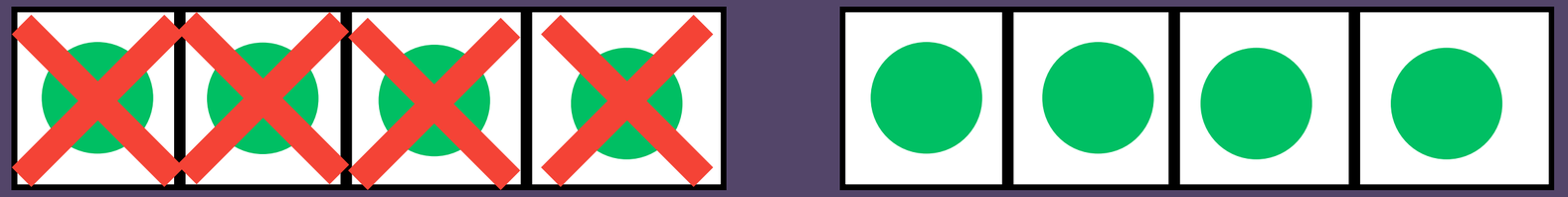
CASE 2

WP → write pointer

Host



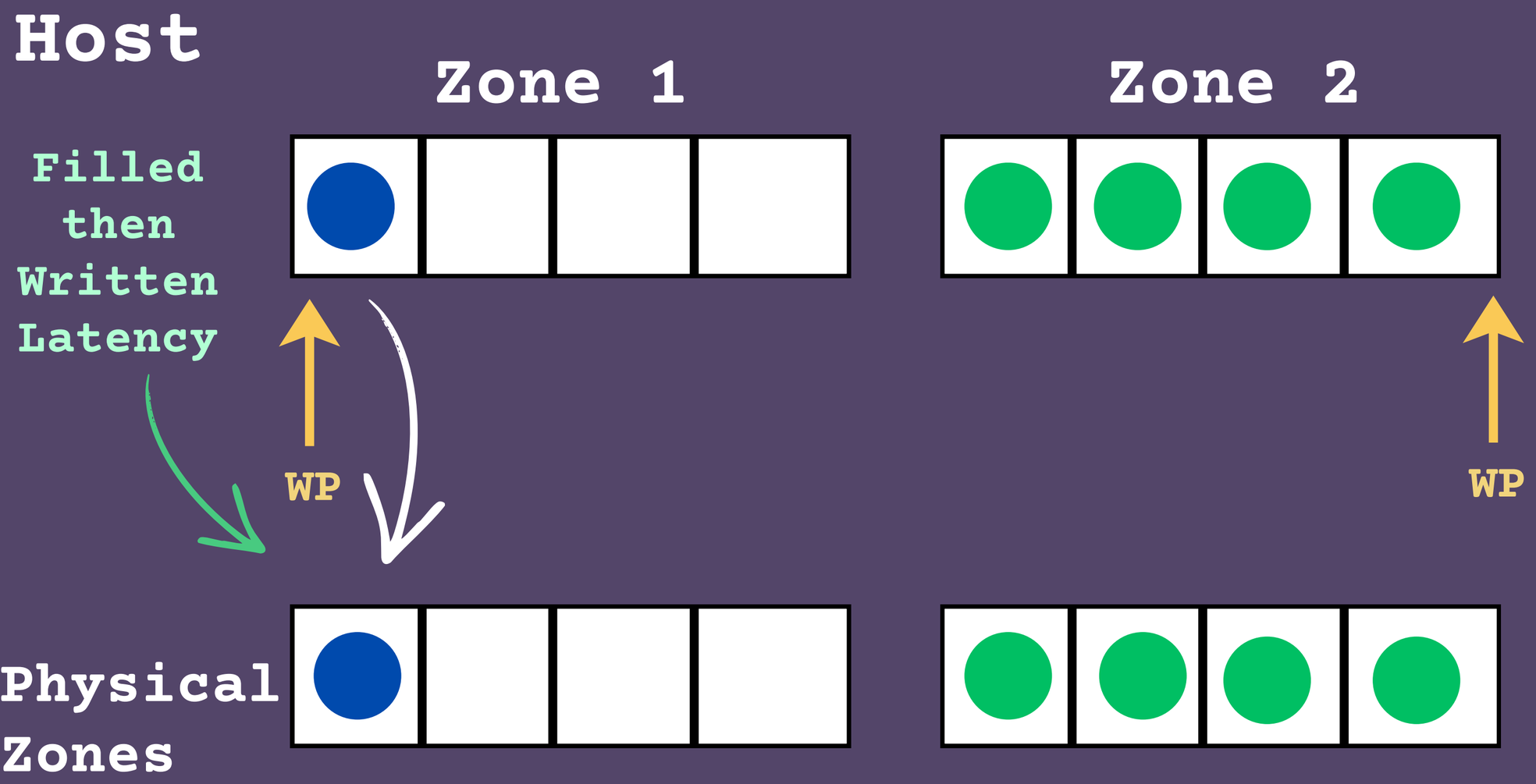
Physical Zones



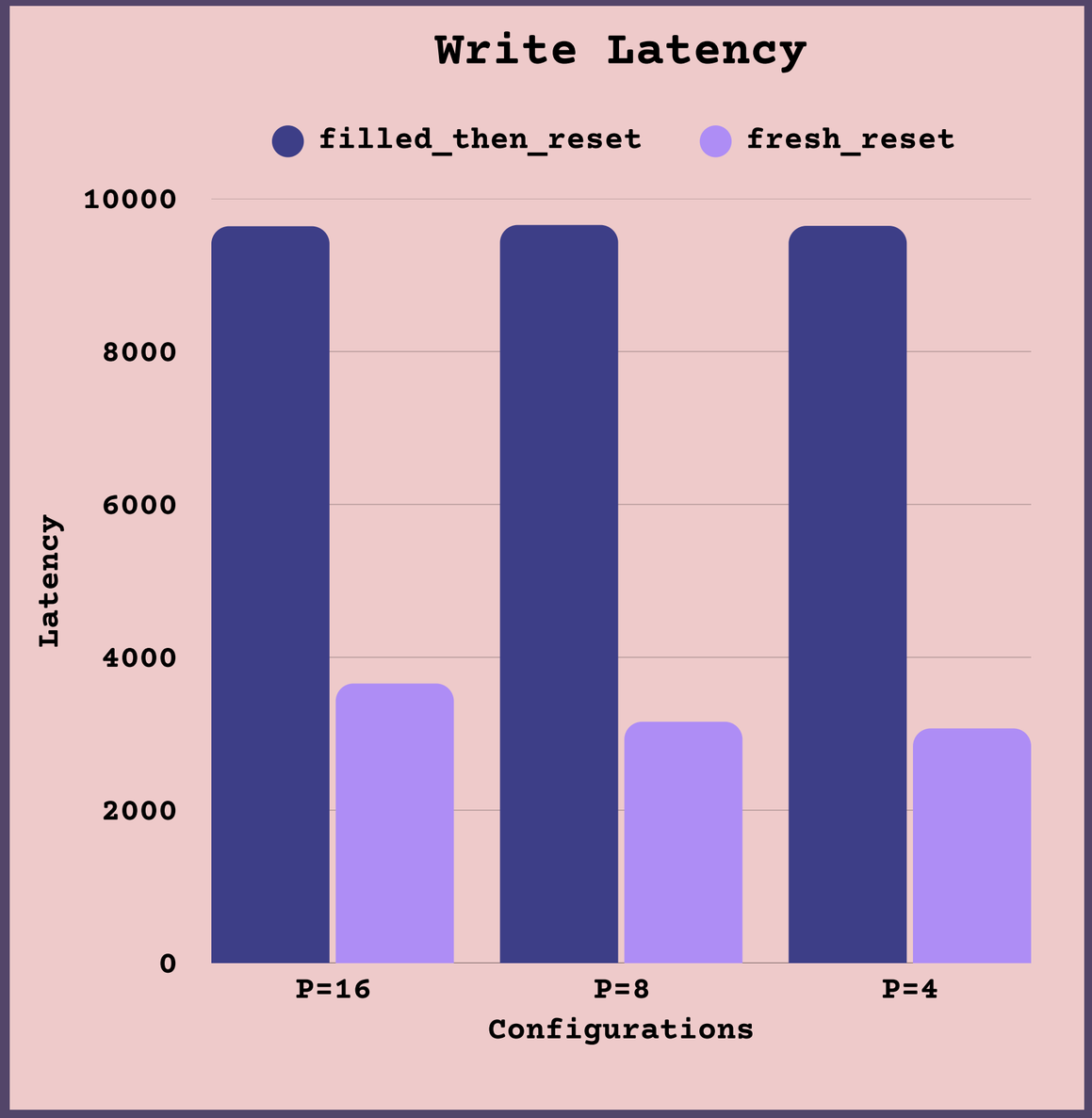
Zone Resets

CASE 2

WP → write pointer



Zone Reset Cost under Prior Write History



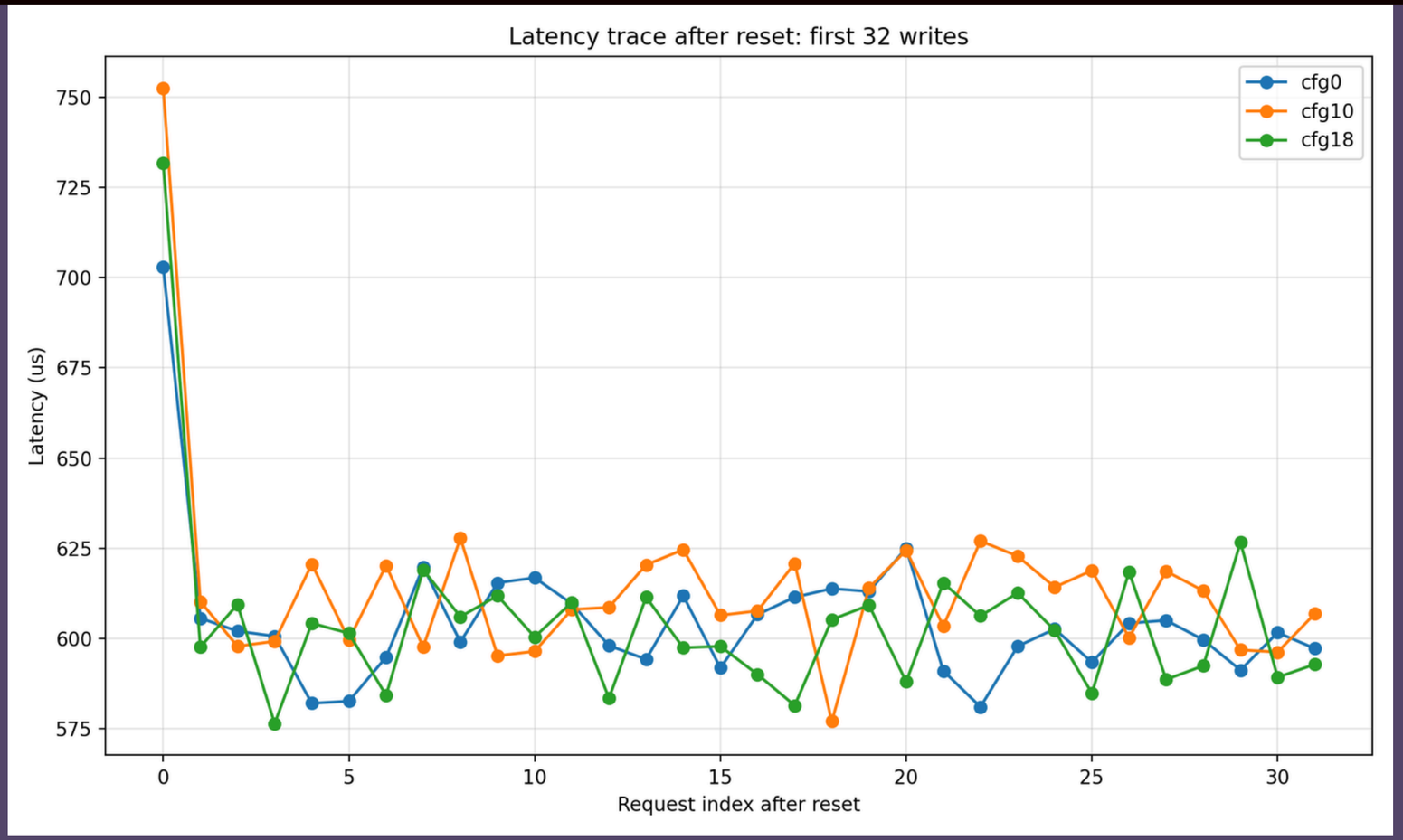
penalty_ratio = filled_then_reset / fresh_reset

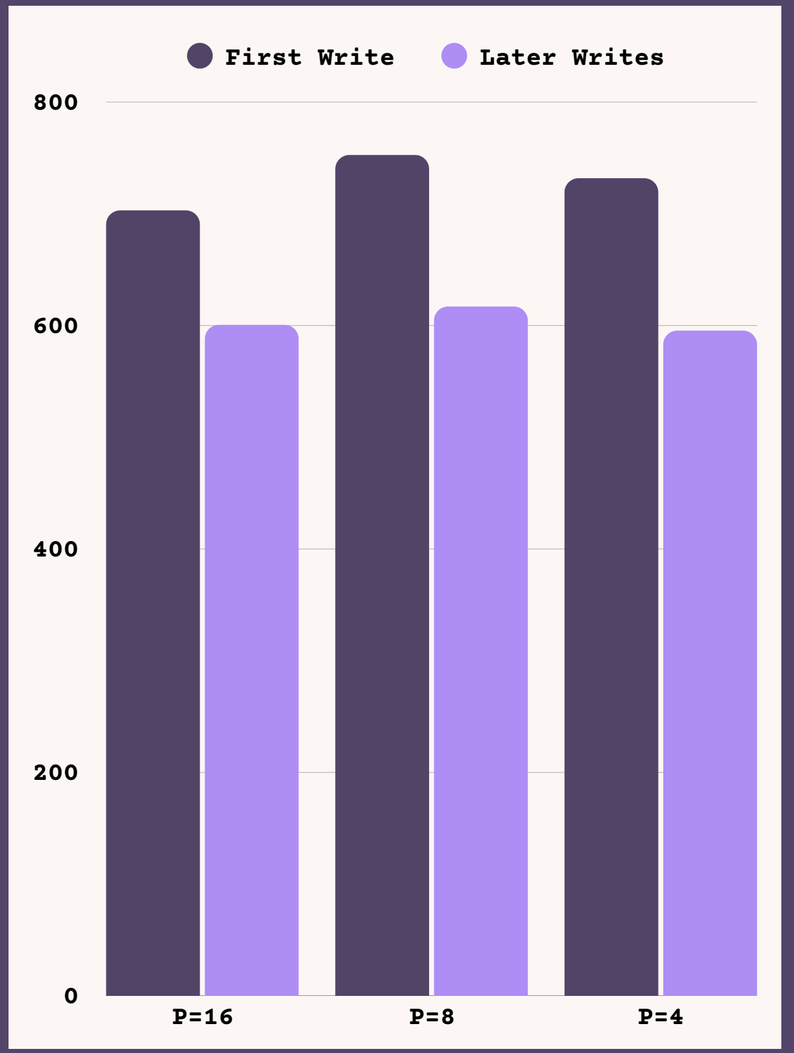
config	filled_then_reset	fresh_reset	penalty_ratio
P=16	9639.333	3657	2.635858
P=8	9655.8	3157	3.058537
P=4	9644.2	3069.6	3.141843

Zone reset cost is history-dependent. Reset latency increases ~3× when the zone has prior write history.

Does this persist through the next zone writes as well?

Transient First-Write Latency after Zone Reset



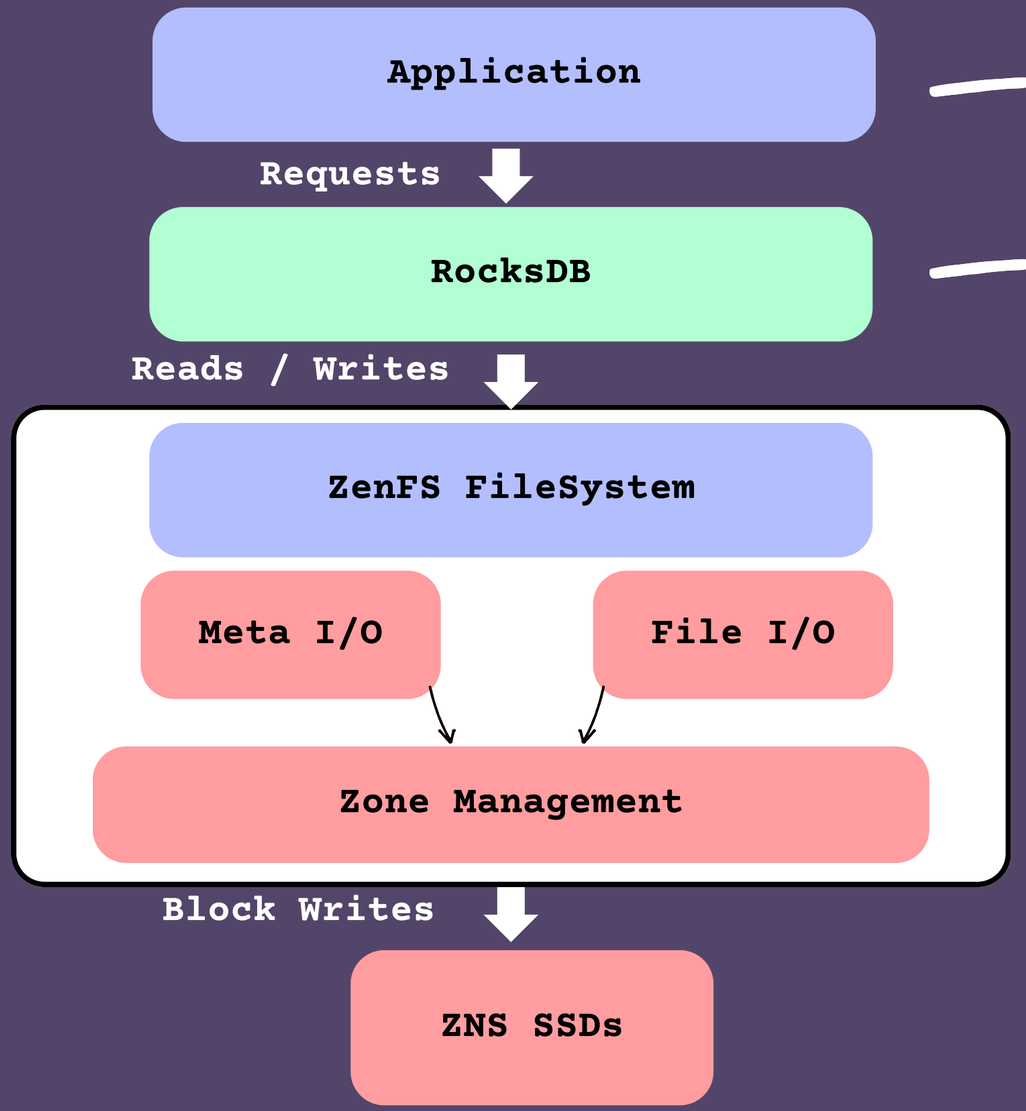


The latency penalty is mostly a one-time remap/erase cost on the first write after reset, not a permanent slowdown for the whole zone.

Application Level Benchmarking : RocksDB + ZenFS

×

RocksDB + ZenFS

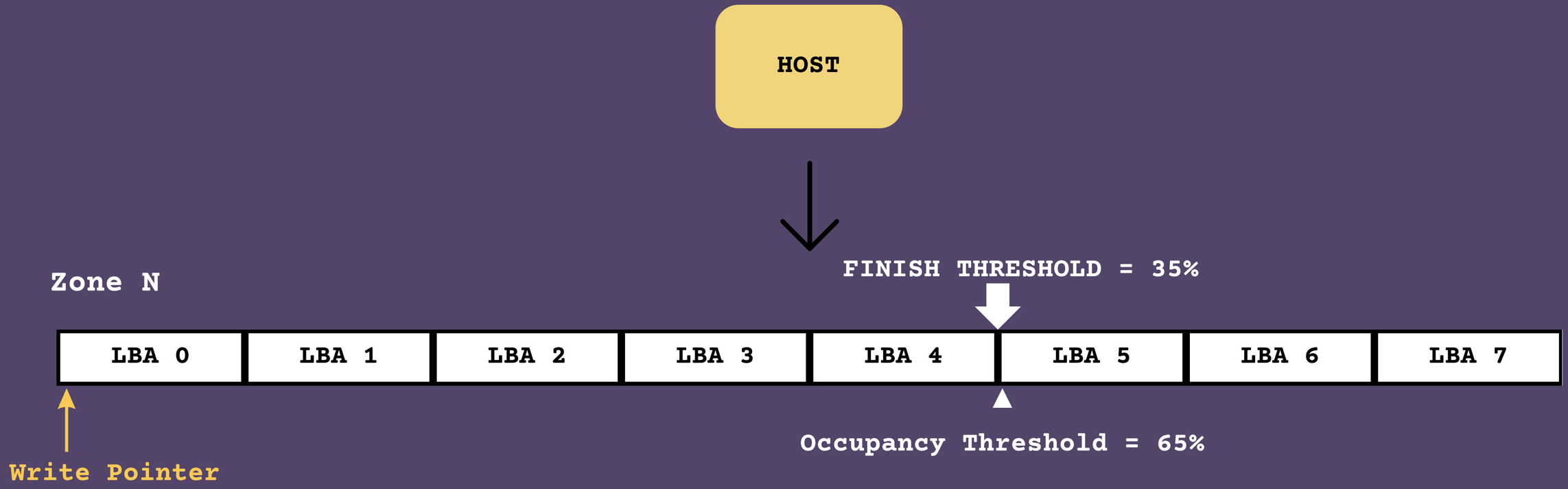


Our Workload generators: dbbench and KVbench!

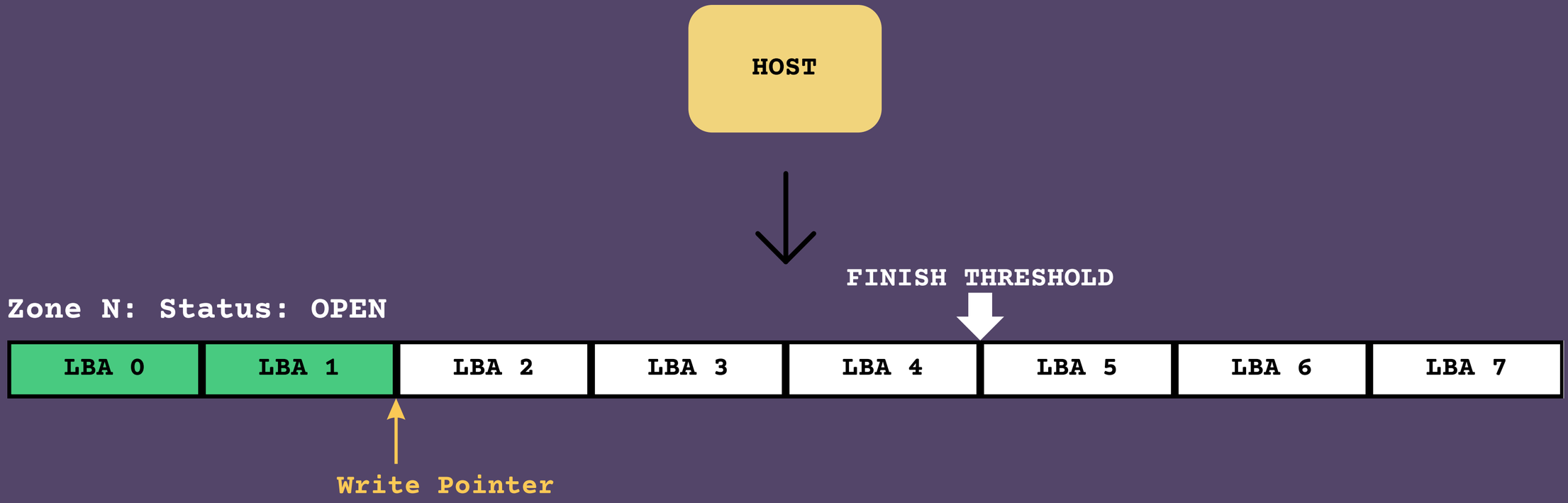
Compactions, flushes, and SST file merging, generating data with varying lifetimes across LSM tree levels

ZenFS is a zone-aware file system that enables RocksDB to work with ZNS SSDs.

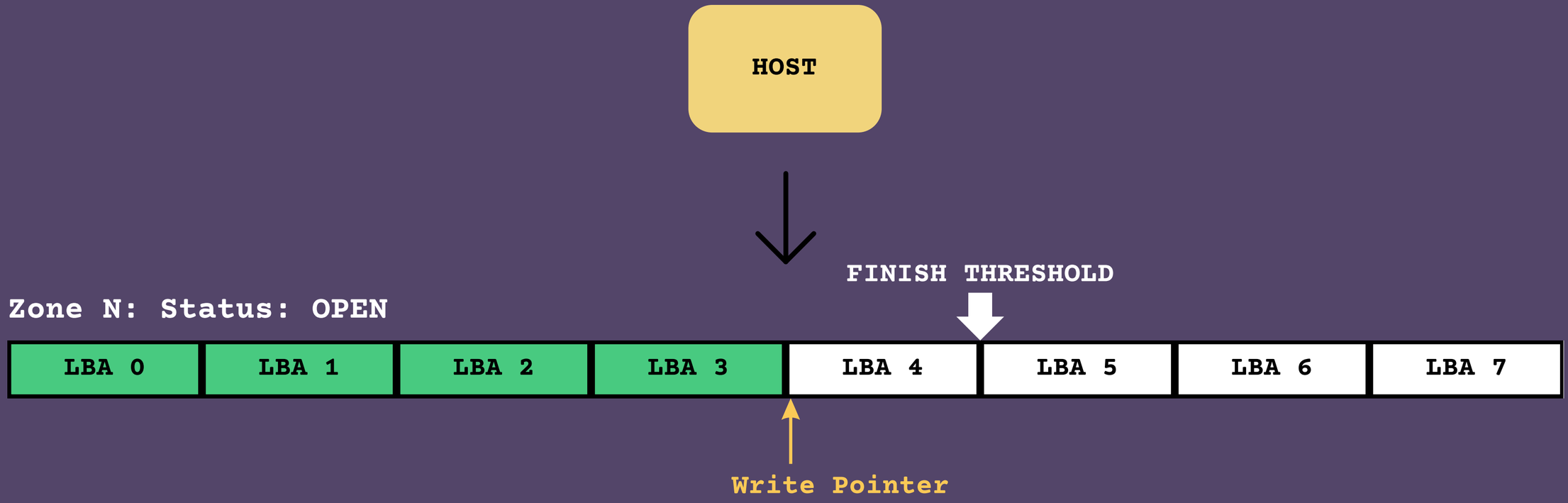
What is the FINISH Threshold?



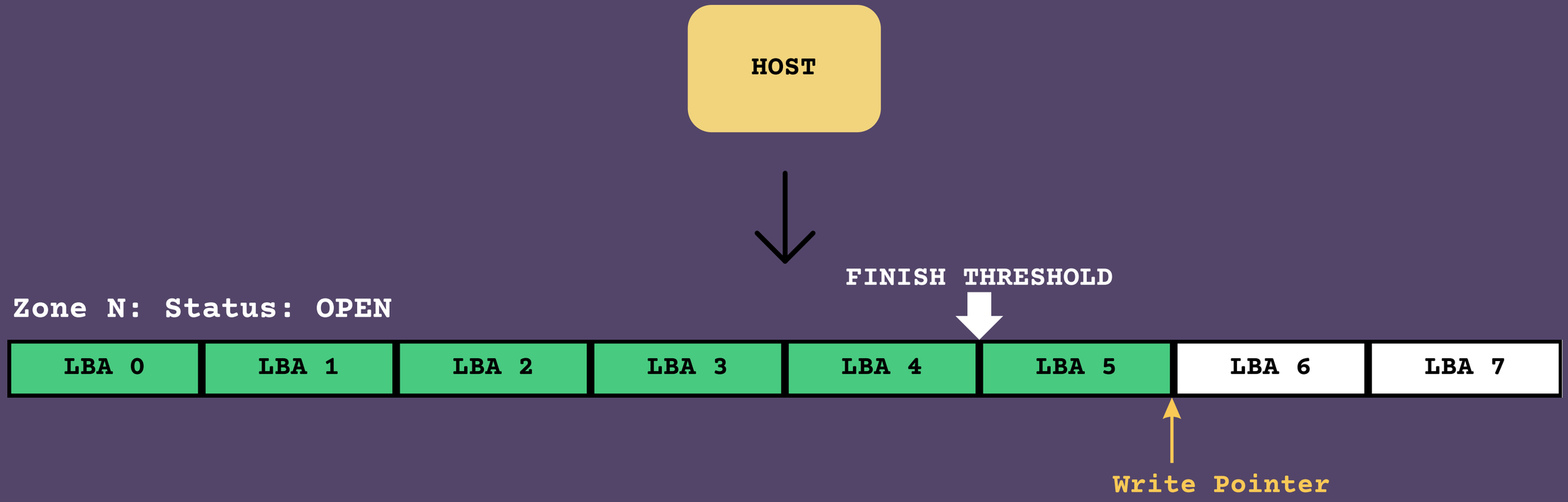
FINISH Threshold



FINISH Threshold



FINISH Threshold



FINISH Threshold

ZONE FINISH!

HOST

FINISH THRESHOLD

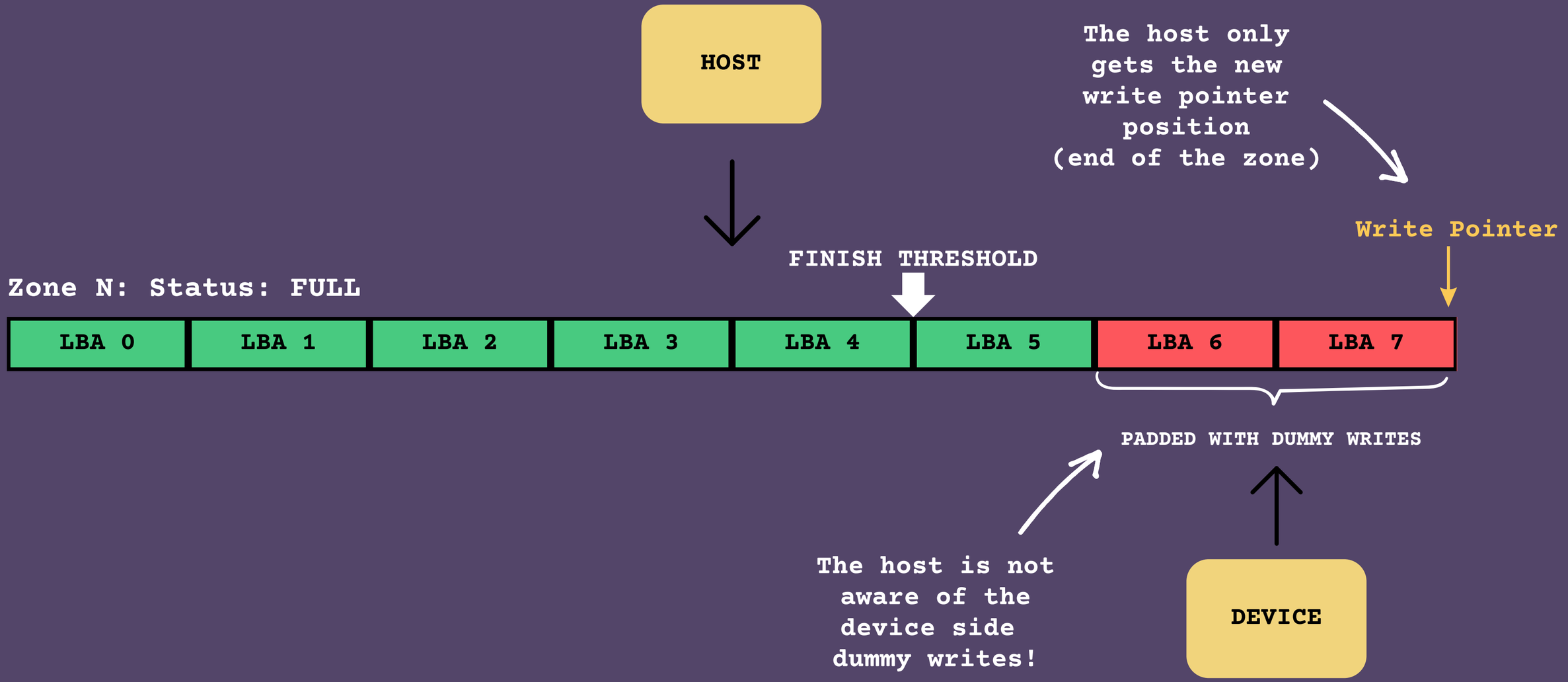
Zone N: Status: OPEN

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7

Write Pointer

DEVICE

FINISH Threshold



Experiments

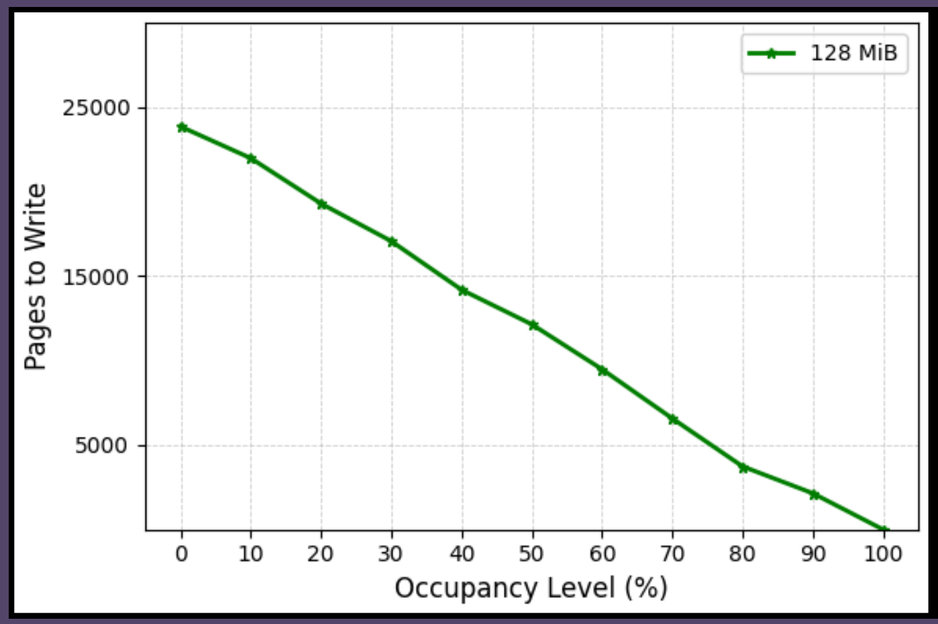
- Vary the occupancy threshold (0% - 100%) across different zone sizes and measure the impact on:-

1. Write Amplification w.r.t Dummy Writes
2. Space Amplification w.r.t Invalid Data
3. Latency and Bandwidth

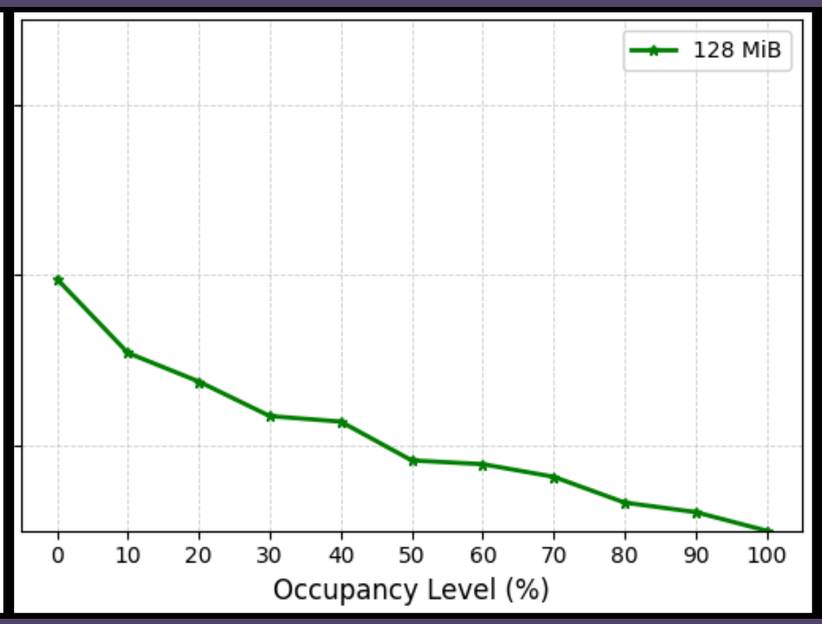
KVbench: using a more realistic workload size: 50% insert, 10% delete, 15% empty PQ, 25% updates

1. Dummy Writes VS varying Occupancy Thresholds

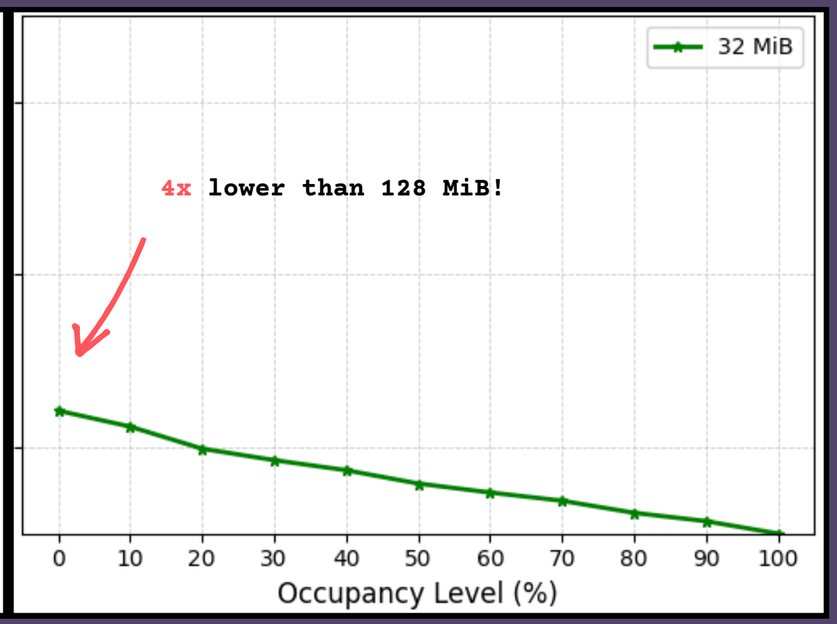
Zone Size = 128 MiB



Zone Size = 64 MiB



Zone Size = 32 MiB



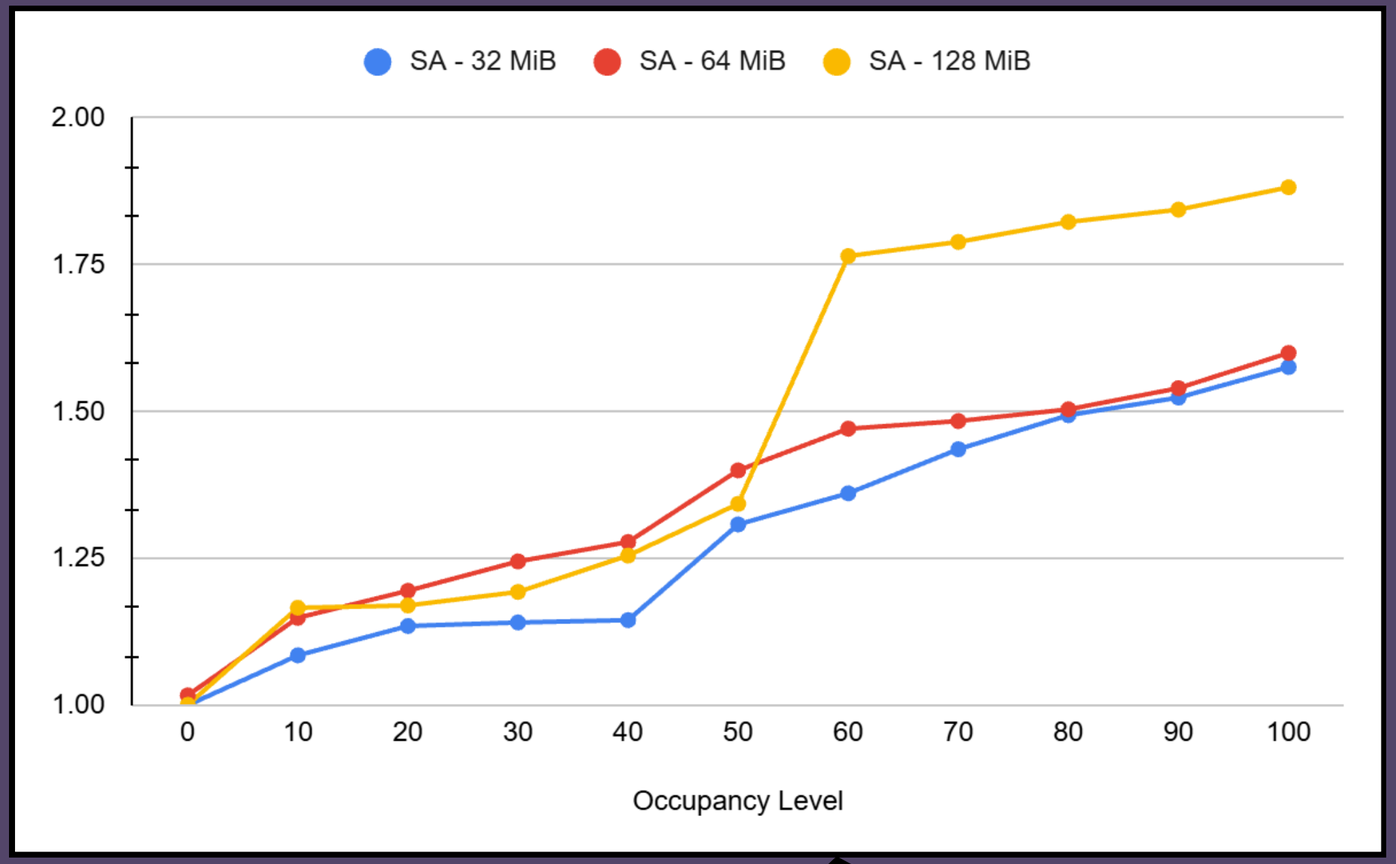
1. **Within a zone size: Higher finish threshold → fewer dummy writes**

2. **Across zone sizes: Smaller zones → fewer dummy writes overall**

2. Comparing Space Amplification across various Zone Sizes

$$SA = \frac{HostWrites + GarbageBytes}{HostWrites}$$

Larger zones show higher space amplification at higher occupancy levels compared to smaller zones



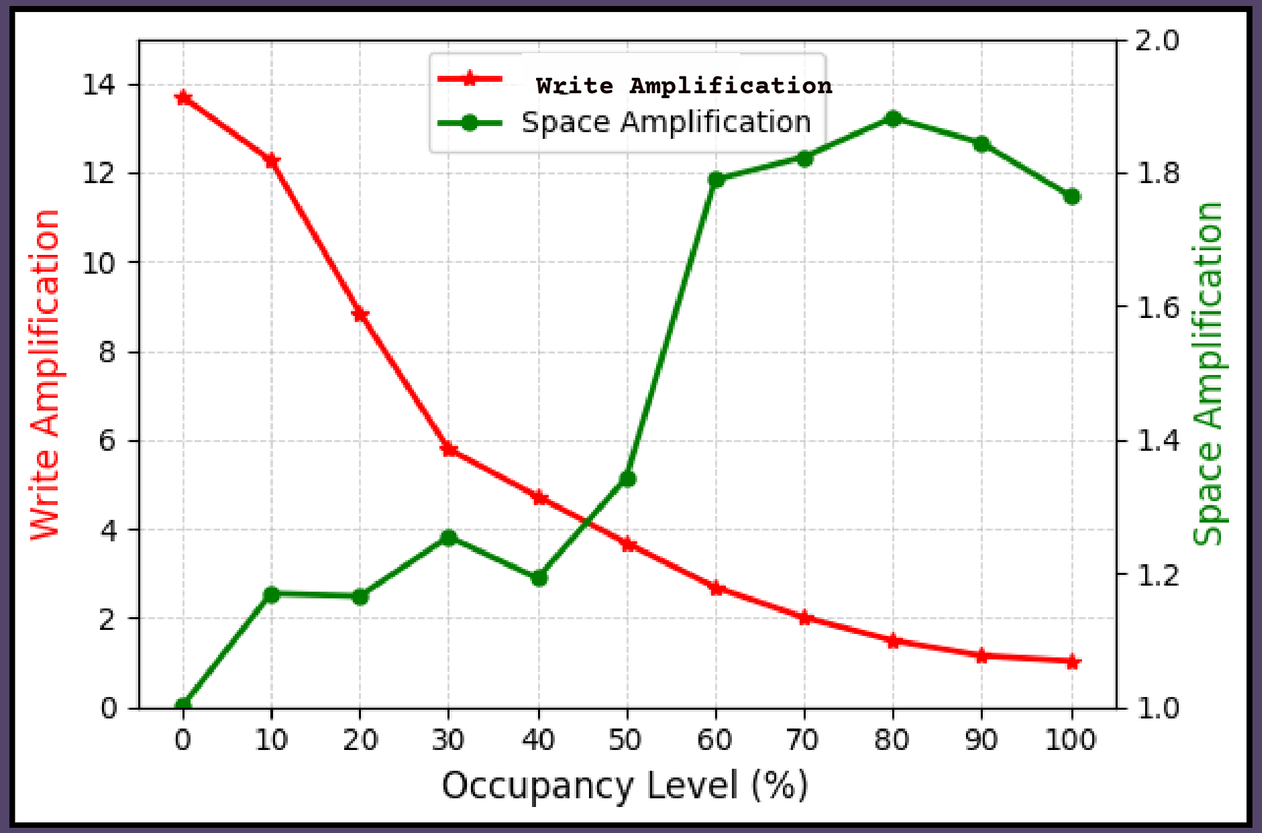
Increasing SA with increasing occupancy

Write Amplification VS Space Amplification

$$WA = \frac{HostWrites + DeviceWrites}{HostWrites}$$

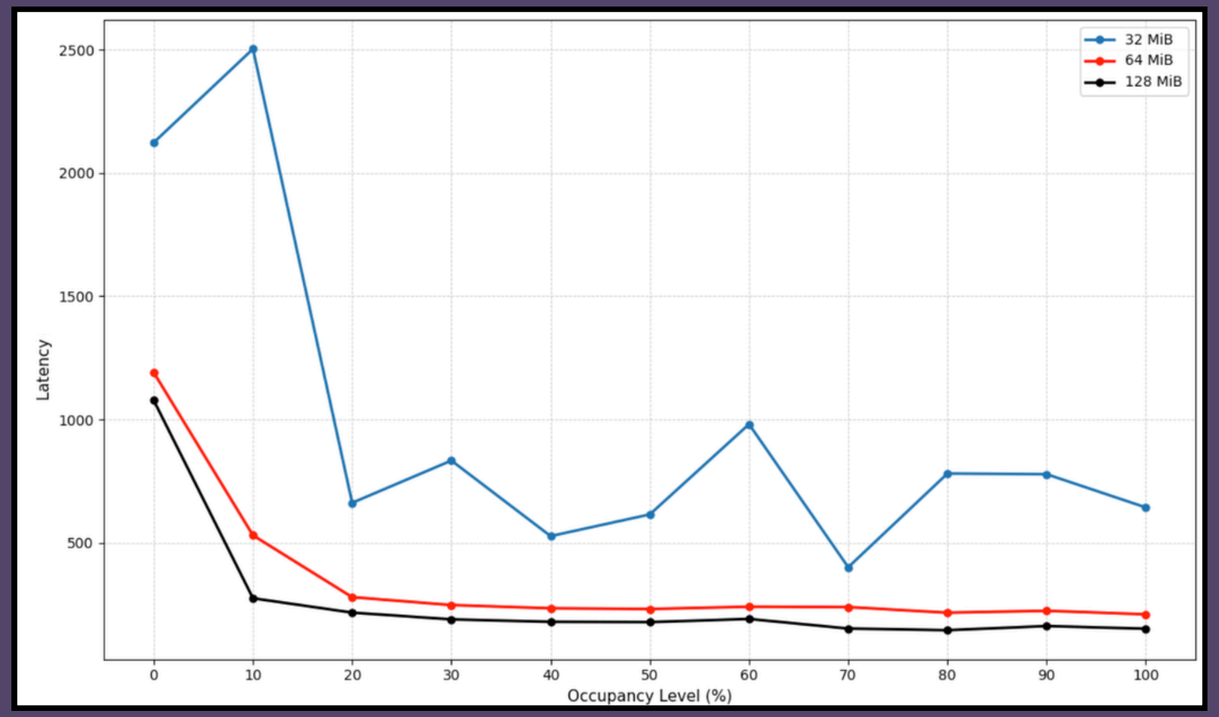
As occupancy increases, Write Amplification drops but Space Amplification rises!

$$WA \propto \frac{1}{SA}$$

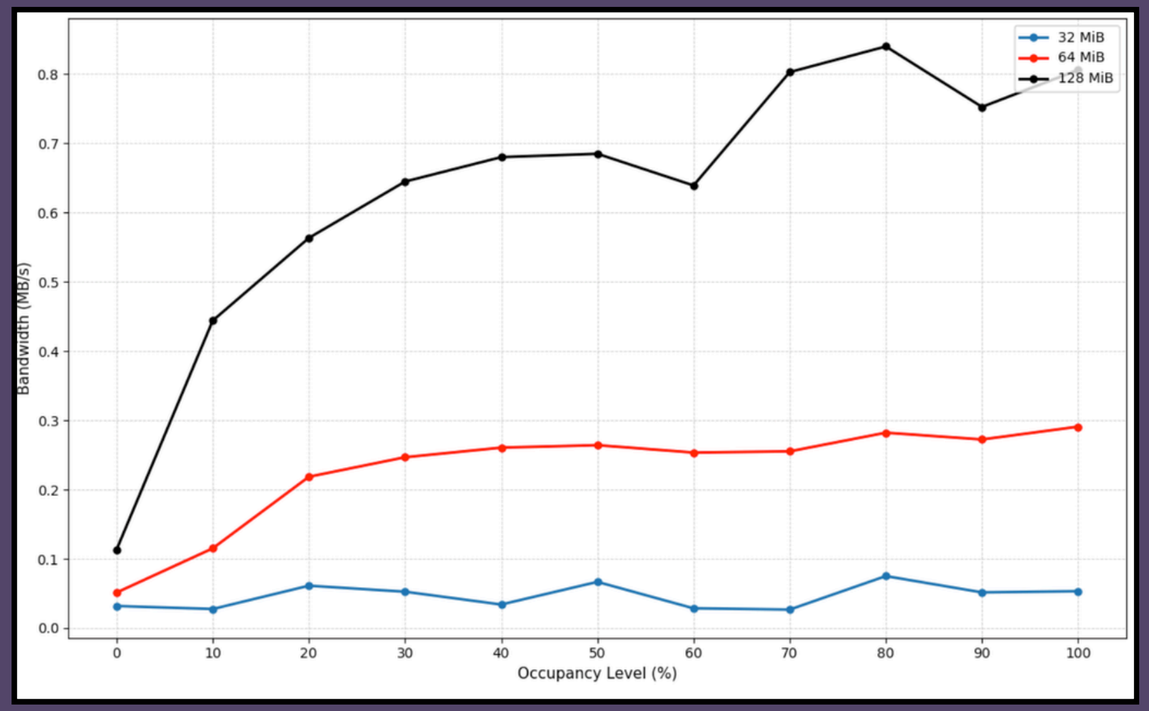


Zone Size: 128 MiB

3. Latency and Bandwidth



Latency vs Occupancy Level (varying zone sizes)

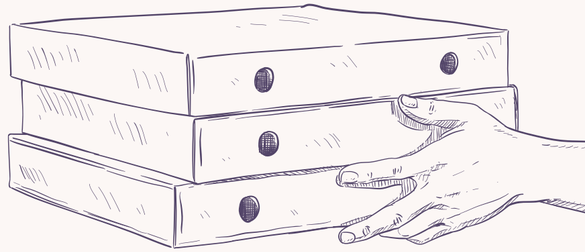


Bandwidth vs Occupancy Level (varying zone sizes)

As occupancy increases, latency drops; 128 and 64 stay stable, but 32 is unstable

Bigger zones + higher occupancy = better bandwidth utilization

Conclusion



Sequential / High Throughput

Large zones maximize parallelism and stabilize latency, but increase space and reset costs.

Partial Writes / Efficiency

Smaller zones reduce wasted work and extra writes, but they don't fully utilize performance and can become unstable under load.

There's no single 'best' choice, it really comes down to the type of workload (read vs write heavy)!

FUTURE WORK

- Measure wear leveling distribution across zones under different workload patterns and zone sizes
- Benchmark GC (Garbage Collection) behavior in ZNS SSDs, analyzing its impact on latency, space amplification and write amplification
- Try more configurations with different vtable values.

Learn More →

Thank you!