Concurrency Aware Algorithm Design For ZNS SSD's

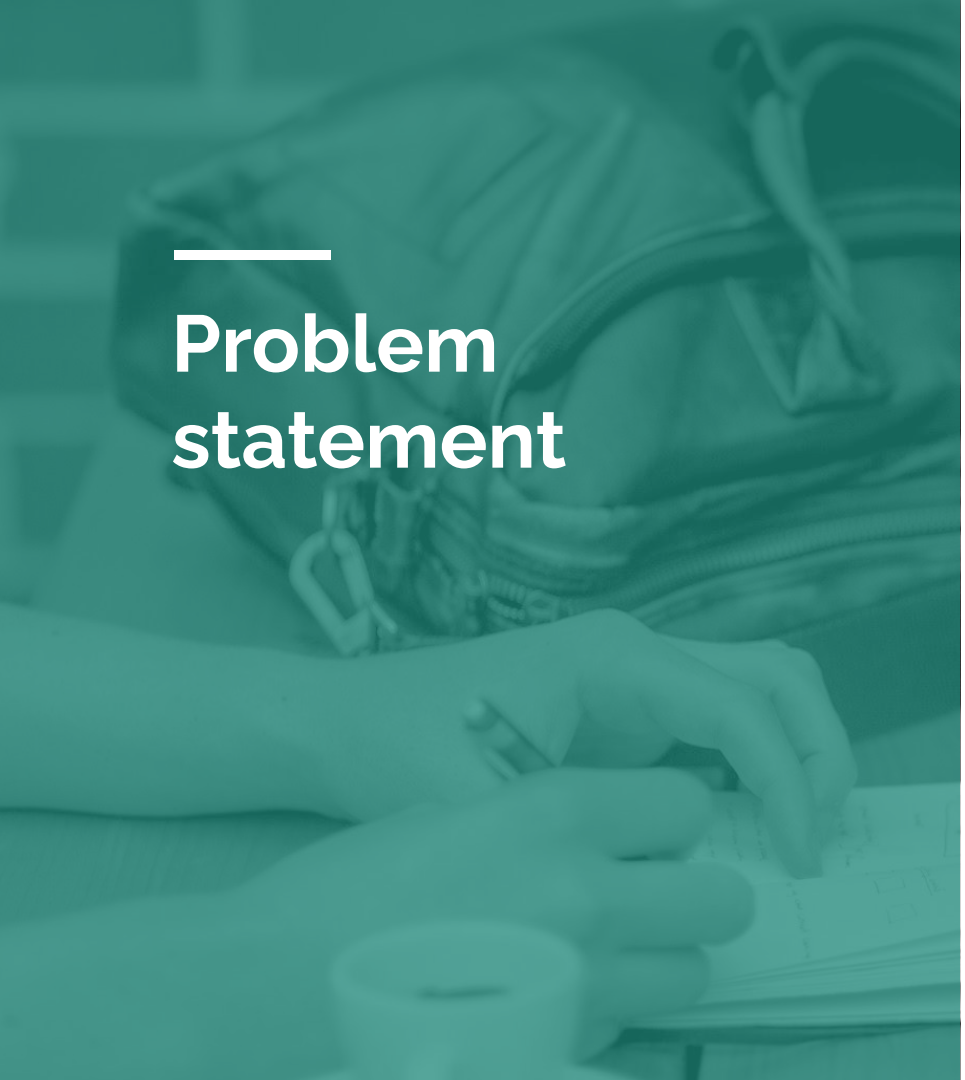
Anastas Kanaris, Bruce Casillas, Ross Mikulskis

Motivation

Why is Concurrency Aware Algorithm Design Important?

To Maximize Modern Hardware Capabilities

- Enable Parallelism
- Maximize Throughput
- Mitigate Write Amplification
- Without this, we won't know how to fully utilize hardware



Problem statement

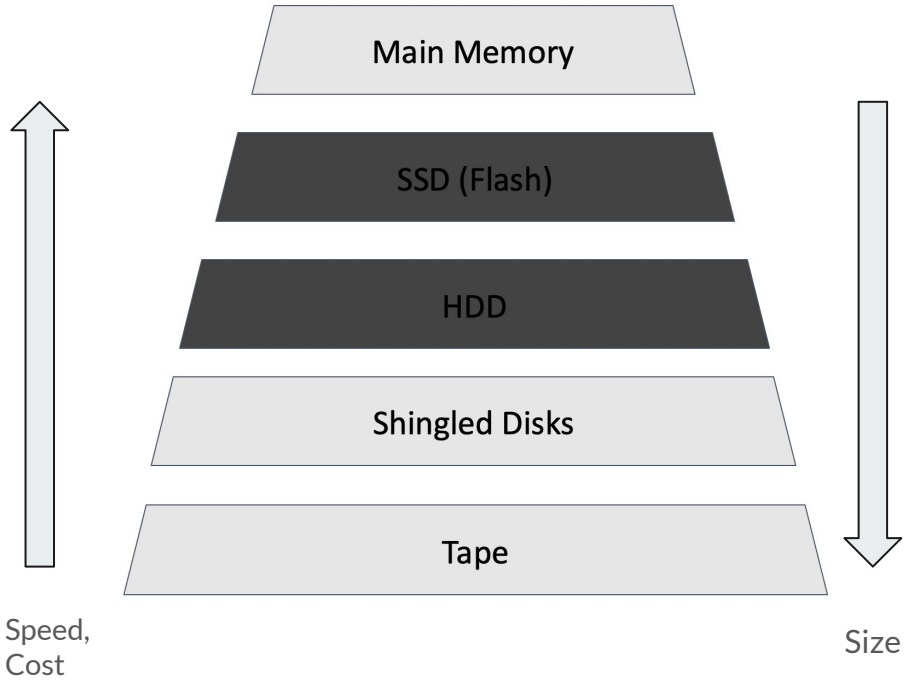
How can we quantify read/write asymmetry and concurrency in ZNS SSDs?

- How do we test for this, without buying a bunch of different physical SSD's.
- Why is it important to know this for ZNS?

Storage Hierarchy

01

Flash SSD's are taking the place of HDD's in storage pyramid because the speed increases are worth more than the cost increase

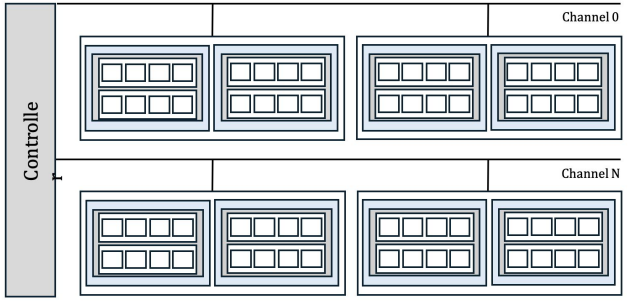


Recall: Storage Hierarchy Pyramid

Traditional SSD

02

- **Random writes** to any logical block address.
- **Device firmware** handles internal complexities like wear leveling and garbage collection.
- The host (OS/filesystem/database) has **no visibility** into the internal physical layout.



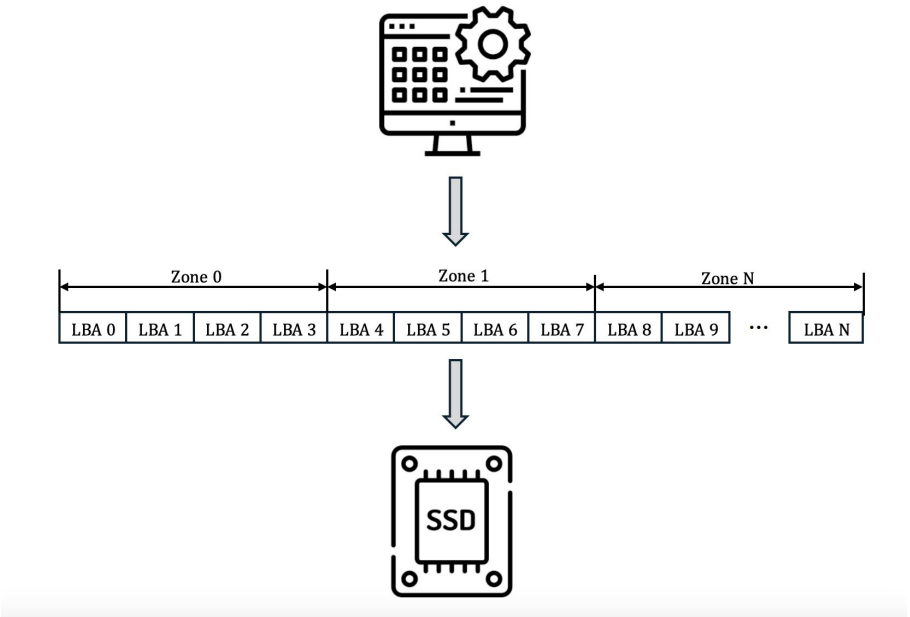
Flash Package -> Chip -> Die -> Plane -> Erase Block -> Page -> Nand Cells

Problem: Block Interface Tax

ZNS SSD

03

- Storage divided into **zones**, and **writes must be sequential within each zone**.
- The host is **responsible** for managing zone write pointers and handling resets.
- This **offloads complexity** from the SSD firmware to the host, enabling better **coordination and performance**.

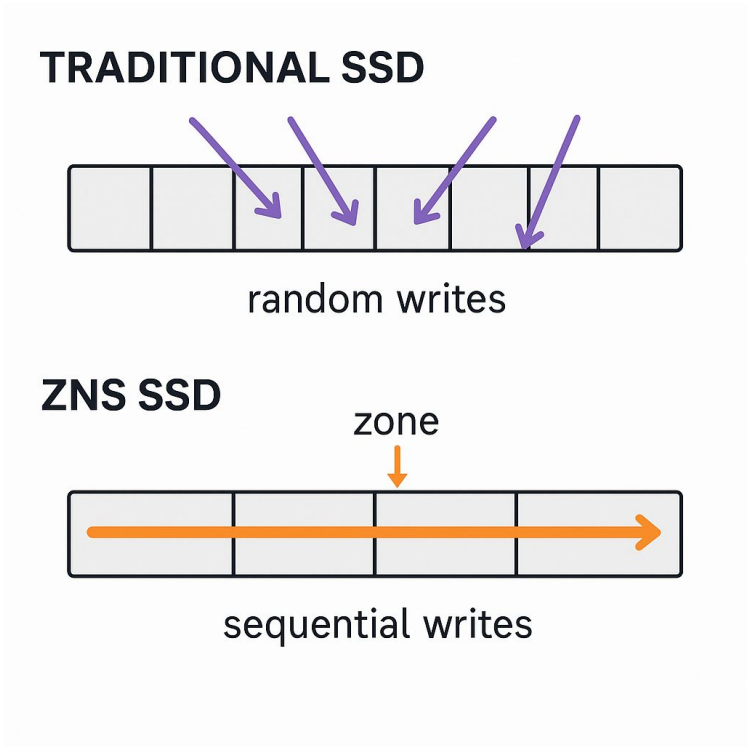


Comparing SSD Types

04

Traditional SSDs: Still maintain block interface design, simplifies host software. Operational cost of supporting this are growing prohibitively. Mismatch between allowed operations and flash media.

ZNS SSDs: Designed to better take advantage of the flash media they’re built on. More complex to use, but offer performance benefits

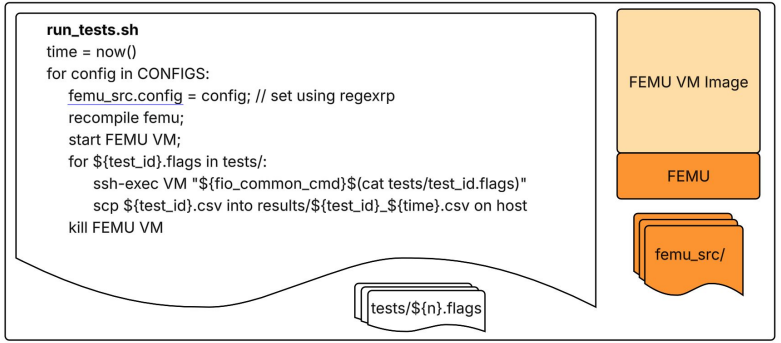


How do we test different configurations?

- ➔ ConfZNS++!
- ➔ Developed by Storage and Network research group - VU Amsterdam
- ➔ Allows users to configure zones, enabling workload-aware tuning.
- ➔ Passes this configuration into FEMU, allowing users to emulate ZNS behavior without the physical hardware
- ➔ Inside FEMU we can use FIO to simulate reads/writes to ZNS device

Methodology

- ➔ Install ConfZNS++ Code Artifact in Linux Environment
- ➔ Install FEMU so that ConfZNS++ can connect to it
- ➔ Use ConfZNS++ to load desired ZNS Configuration into VM
- ➔ Inside VM run FIO tests
- ➔ Send results back to host machine and check for overarching trends



Tested Parameters - ZNS Configurations

- ➔ SU Zone: 1:1 Zone-Parallel Unit Mapping
- ➔ MU^2 Zone: 1 Zone Maps to 2 Parallel Units
- ➔ MU^4 Zone: 1 Zone Maps to 4 Parallel Units
- ➔ FU Zone: 1 Zone Maps to all Parallel Units

Tested Parameters - FIO

- ➔ –numjobs
  - ◆ Simulates concurrent access from multiple sources
- ➔ –iodepth
  - ◆ Controls level of queuing and parallelism within each job.
- ➔ -blocksize
  - ◆ Controls how large each I/O request is

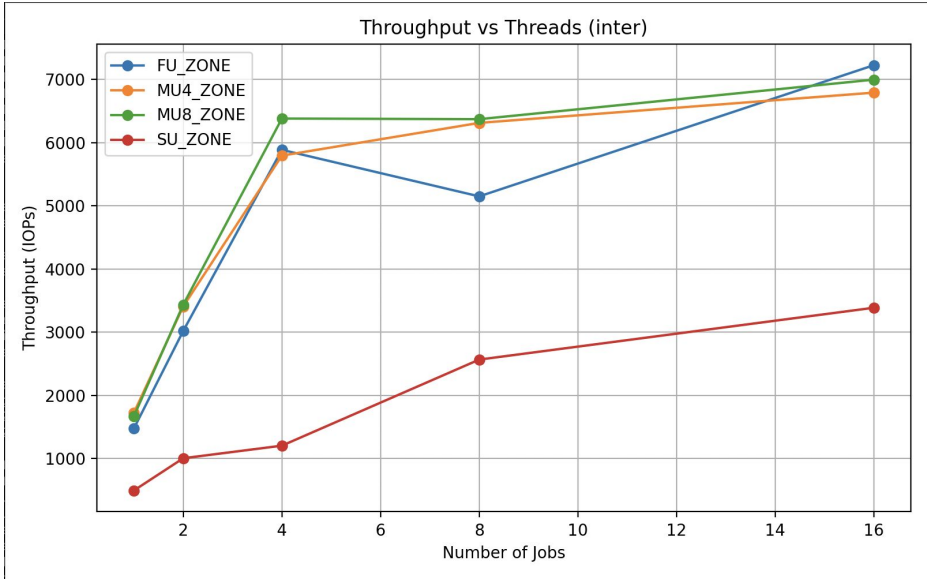
Arguments	Meaning
--numjobs=int	The number of threads spawned be the test.
--iodepth=int	Controls how many I/O requests it issues to the OS at any given time. A sequential job with --iodepth=2 will submit two sequential IO requests at a time.

Experiments

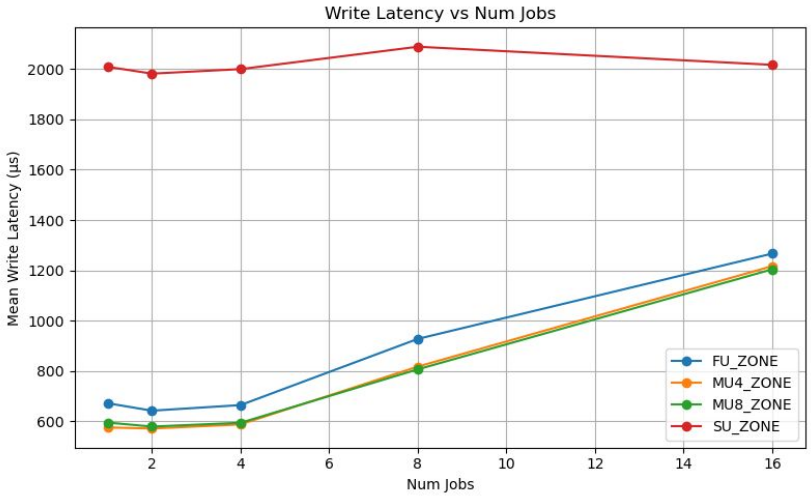
Test: Inter-Zone Parallelism

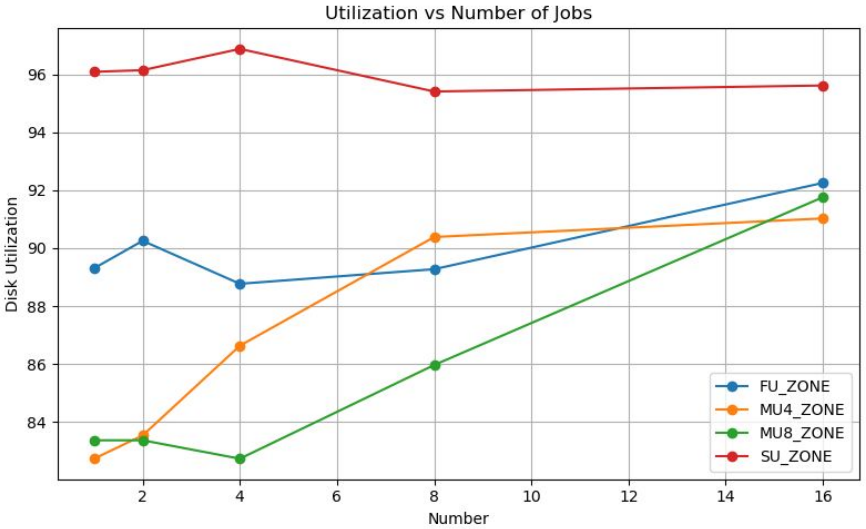
Idea: Increase the number of jobs (threads) spawned by the test

This simulates concurrent access from multiple sources

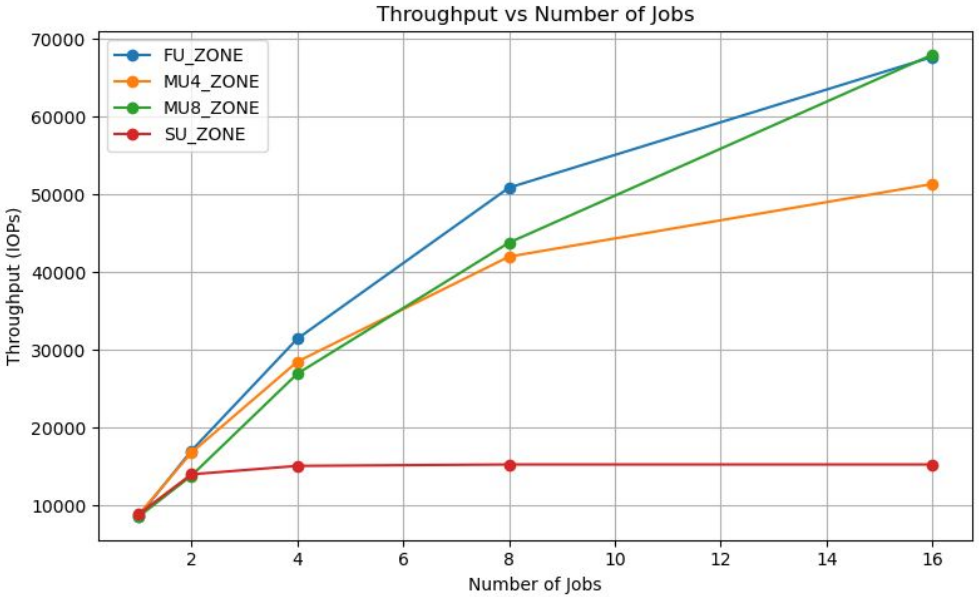


Test: Inter-Zone Parallelism (cont.)





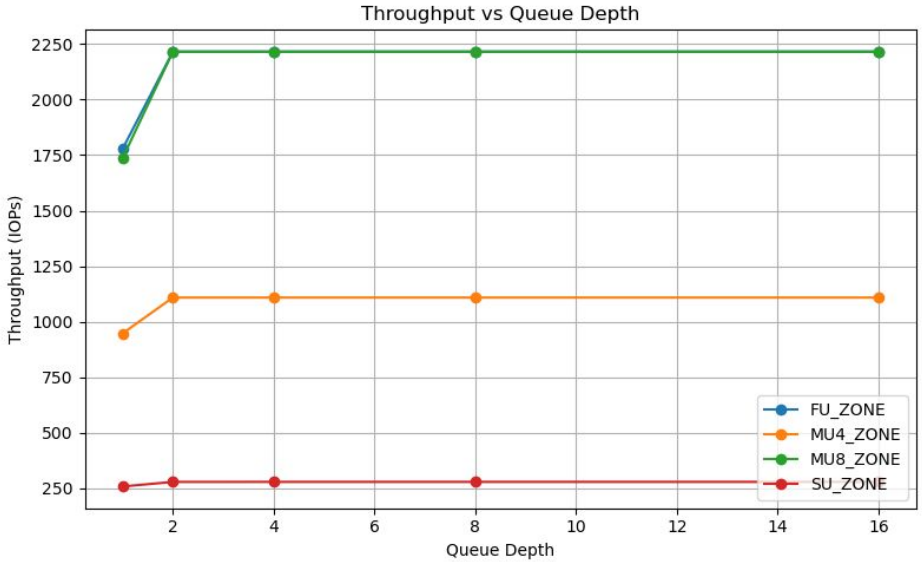
Test: Inter-Zone Parallelism (Reads)



Test: Intra-zone Parallelism

Idea: Increase the io-depth, increasing the number of sequential IO requests

Did not really gain meaningful information from this test, possible there's a better parameter to use



Tests: Mixed Workloads

Idea: test mix of read/writes instead of just writes

Have had issues configuring this test but hoping to have results soon!

Conclusions

What did we learn?

- ➔ How Storage Hardware has evolved over time
- ➔ How FIO and FEMU/ConfZNS++ can be used to experiment with different ZNS Emulations
- ➔ How Inter-Zone Parallelism can vary across the different configurations

Possible next directions

- ➔ Complete Mixed Workload tests to evaluate read-write asymmetry
- ➔ Measure impact these different configurations have on other zns specific commands (ie zone-finish)

References

[1]Papon, Tarikul Islam, and Manos Athanassoulis. "A parametric I/O model for modern storage devices." *Proceedings of the 17th International Workshop on Data Management on New Hardware*. 2021

[2]Doekemeijer, Krijn, et al. "Exploring I/O Management Performance in ZNS with ConfZNS++." *Proceedings of the 17th ACM International Systems and Storage Conference*. 2024.

[3] Im, Minwoo, Kyungsu Kang, and Heonyoung Yeom. "Accelerating RocksDB for small-zone ZNS SSDs by parallel I/O mechanism." *Proceedings of the 23rd International Middleware Conference Industrial Track*. 2022.