Exploring Storage Stack on Modern NVMe Hardware

(lots of graph and numbers coming up - apologies)

Animesh Trivedi
Assistant Professor (tenured)
https://animeshtrivedi.github.io/
April 22nd, 2024

Invited talk at the 4th Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems (CHEOPS'24)

Data is Essential to our Society

Financial

Warehouse

GPT - 4

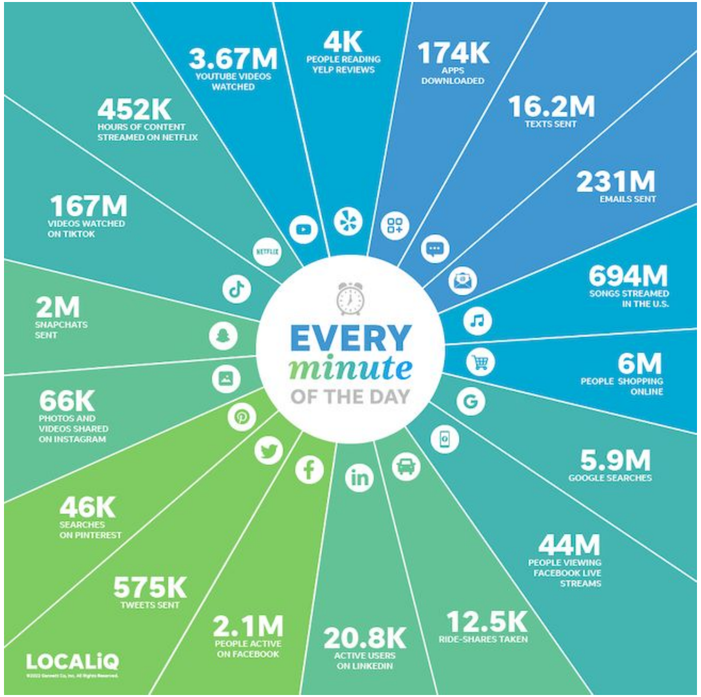
AI/ML

Health

Mobility

Science

A Minute on the Internet



https://localiq.com/blog/what-happens-in-an-internet-minute/

https://www.bondhighplus.com/2022/01/08/what-happen-in-an-internet-minute/

1 Yottabytes

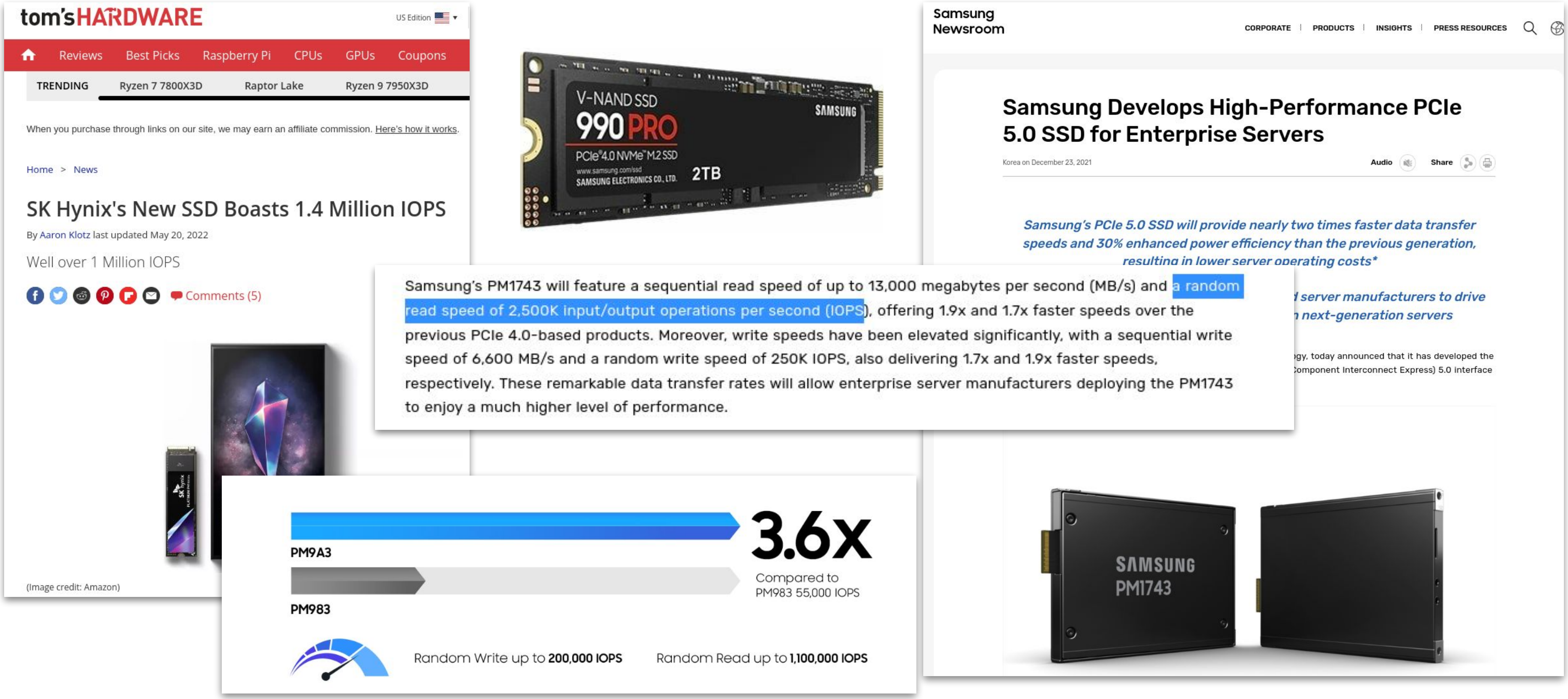
(per year by 2030)

1 byte = 1 grain

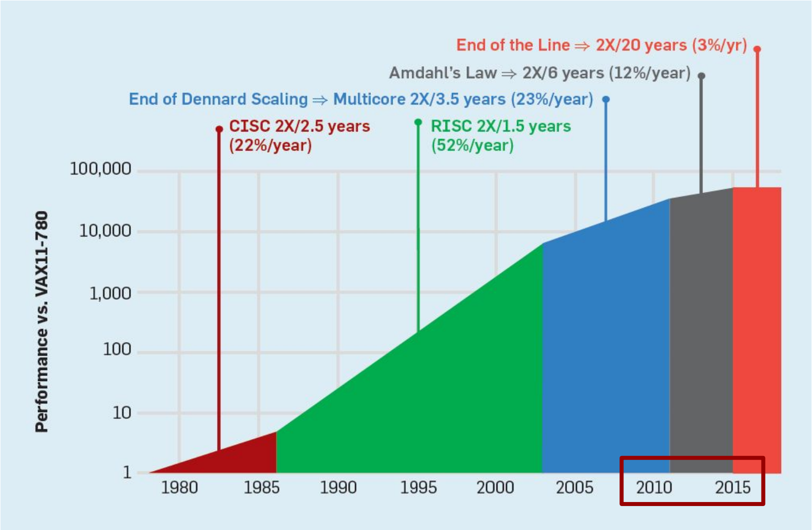
Computing 2030, https://www-file.huawei.com/-/media/corp2020/pdf/giv/industry-reports/computing_2030_en.pdf

What is big data?, David Wellman, https://www.slideshare.net/dwellman/what-is-big-data-24401517

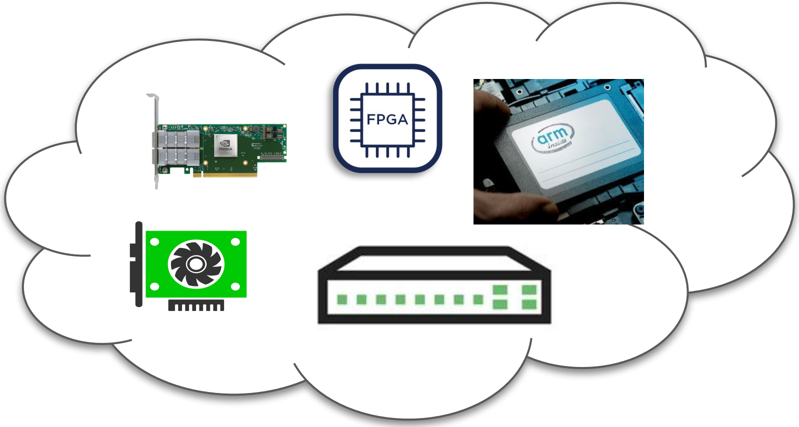
Non-Volatile Memory (NVM) Storage to the Rescue…



Rise of Domain-Specific Computing



Stalled CPU-centric computing scaling

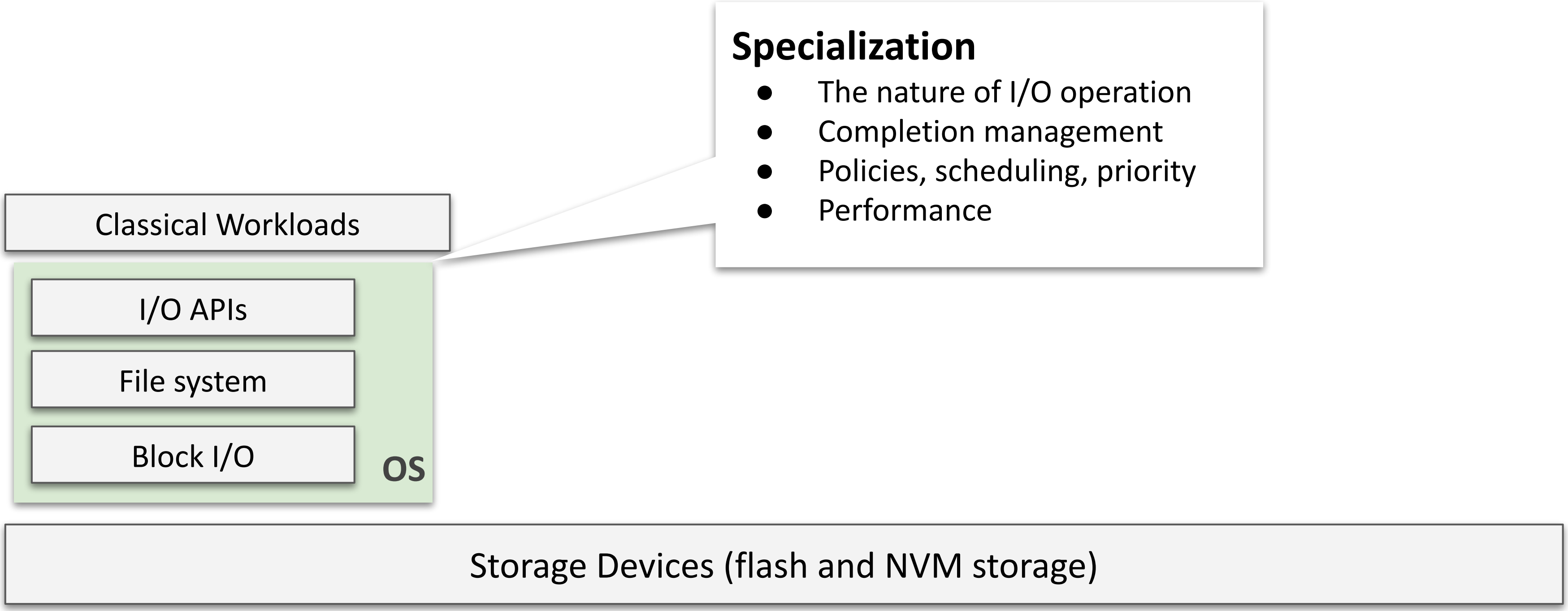


Rise of accelerator-centric computing

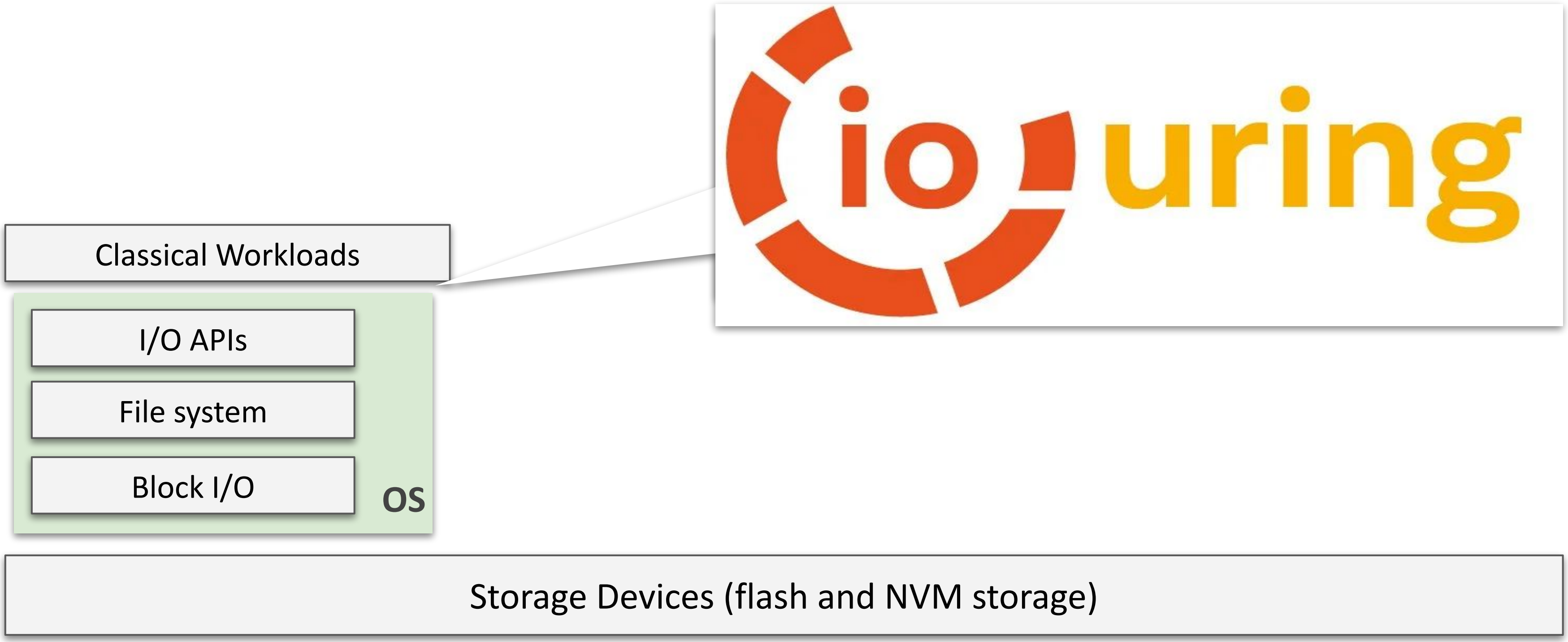
+ Specialized hardware
+ Energy/Perf. gains over the CPU

John L. Hennessy and David A. Patterson. 2019. A new golden age for computer architecture. Commun. ACM 62, 2 (February 2019), 48–60. DOI: https://doi.org/10.1145/3282307

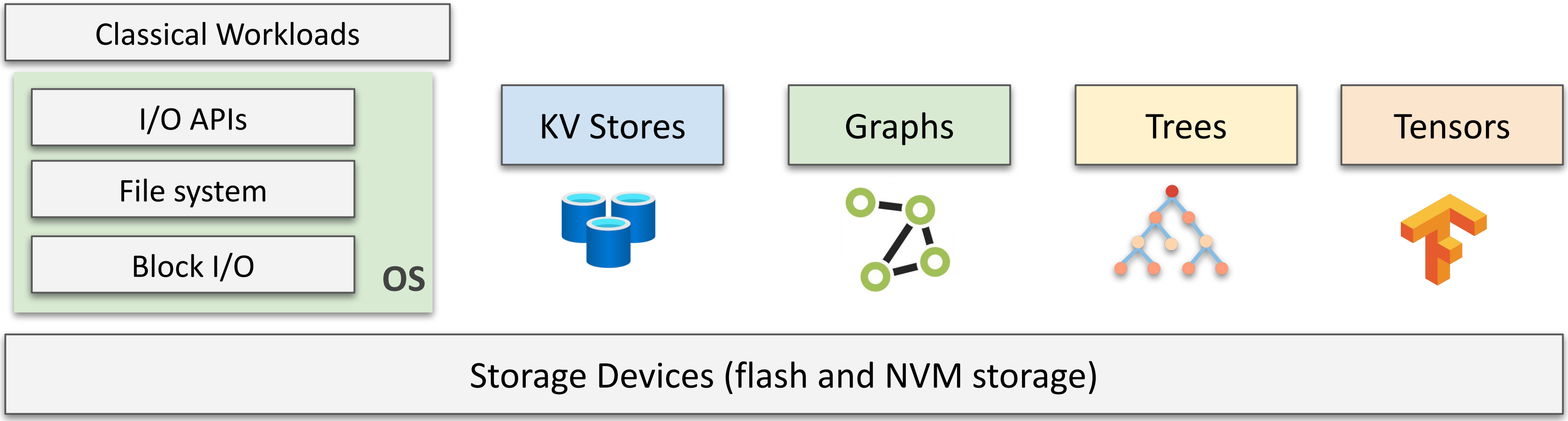
Position: Workload-Specialized Storage will Emerge



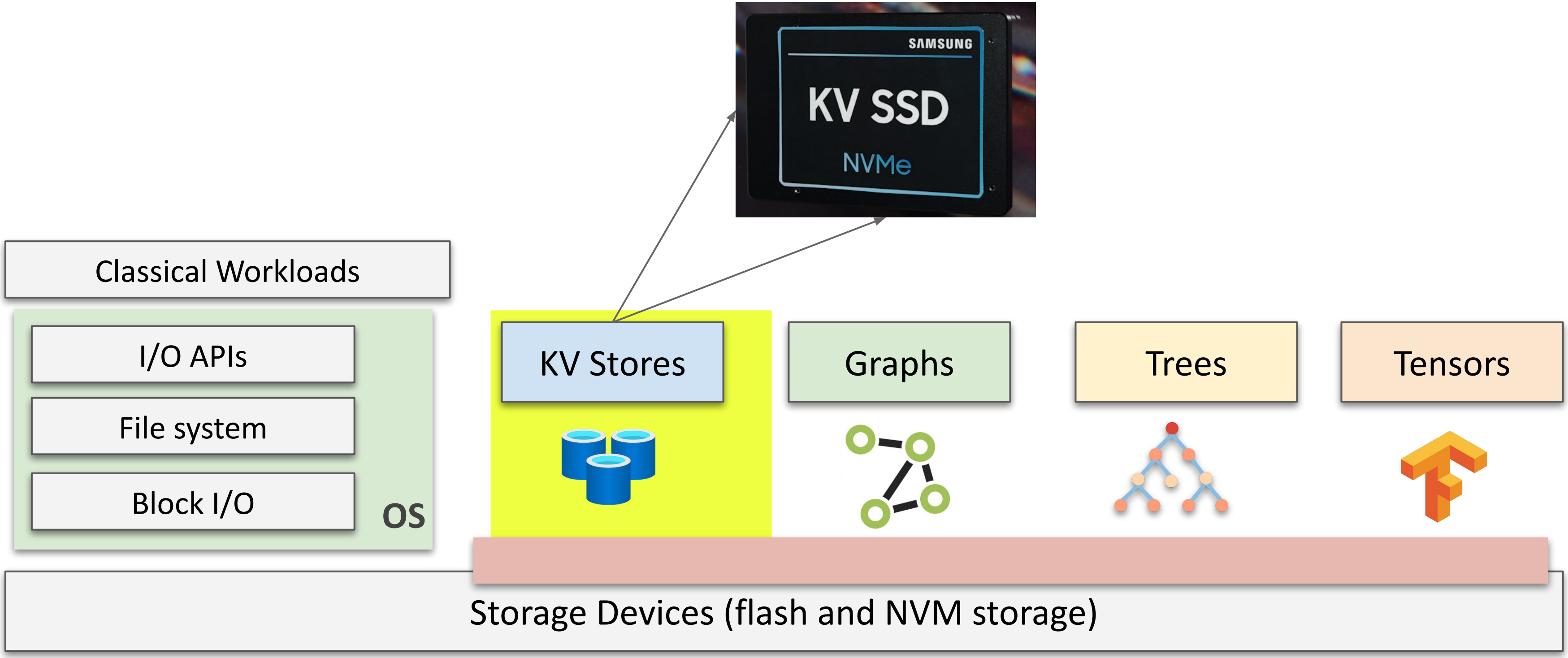
Position: Workload-Specialized Storage will Emerge



Position: Workload-Specialized Storage will Emerge



Position: Workload-Specialized Storage will Emerge



Position: Workload-Specialized Storage will Emerge

[Part - 1/2] : **Study:** I/O Performance and Scheduling

[Part - 2/2] : Zone Namespace Devices (ZNS)

Classical Workloads

I/O APIs

File system

Block I/O

OS

NVMe Zone Namespace Interface

Storage Devices (flash and NVM storage)

[Part - 1/2] : Study: I/O Performance and Scheduling Overheads

Diego Didona, Jonas Pfefferle, Nikolas Ioannou, Bernard Metzler, and Animesh Trivedi. 2022. **Understanding modern storage APIs: a systematic study of libaio, SPDK, and io_uring**. In Proceedings of the 15th ACM International Conference on Systems and Storage (**SYSTOR '22**). Association for Computing Machinery, New York, NY, USA, 120–127. https://doi.org/10.1145/3534056.3534945

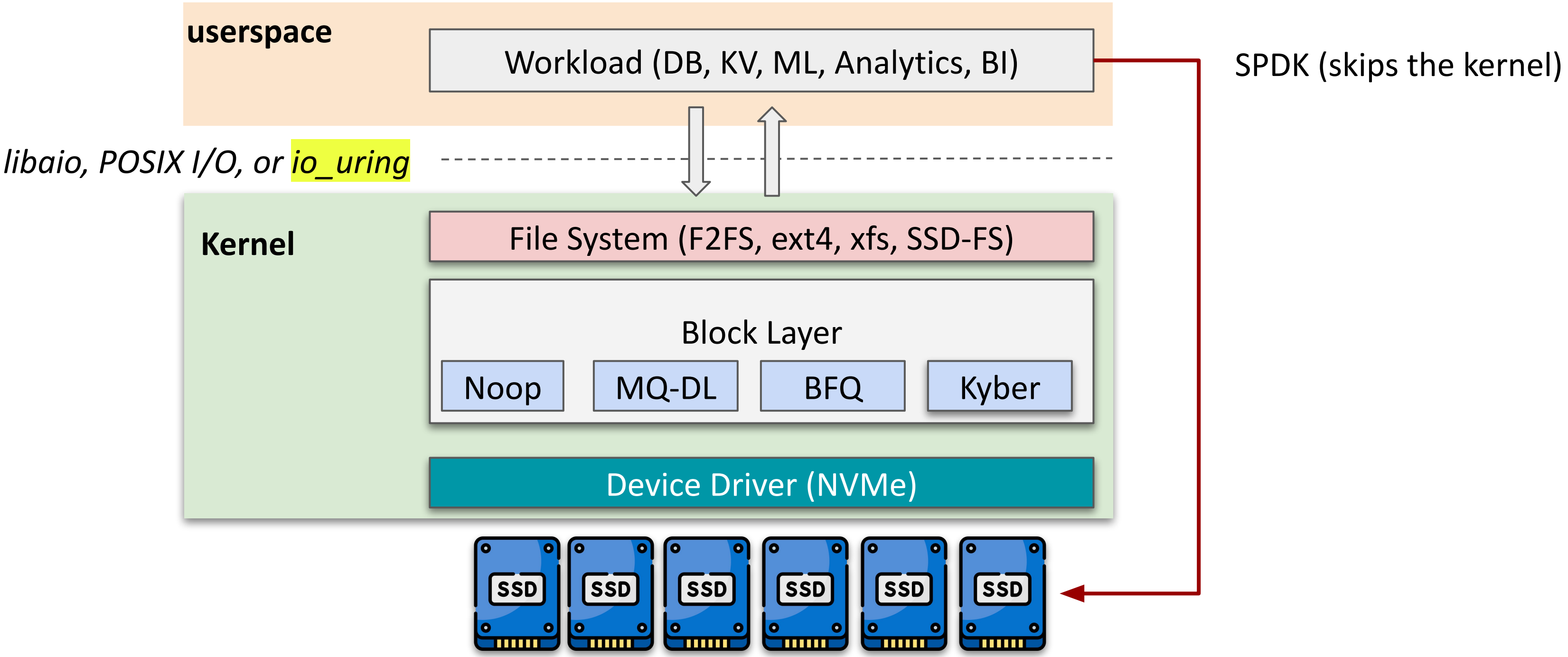
Zebin Ren and Animesh Trivedi. 2023. **Performance Characterization of Modern Storage Stacks: POSIX I/O, libaio, SPDK, and io_uring.** In Proceedings of the 3rd Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems (**CHEOPS '23**). Association for Computing Machinery, New York, NY, USA, 35–45. https://doi.org/10.1145/3578353.3589545

Zebin Ren, Krijn Doekemeijer, Nick Tehrany, Animesh Trivedi. 2024. BFQ, **Multiqueue-Deadline, or Kyber? Performance Characterization of Linux Storage Schedulers in the NVMe Era**, to appear in the 2024 ACM/SPEC International Conference on Performance Engineering (**ICPE '24**), London, UK.

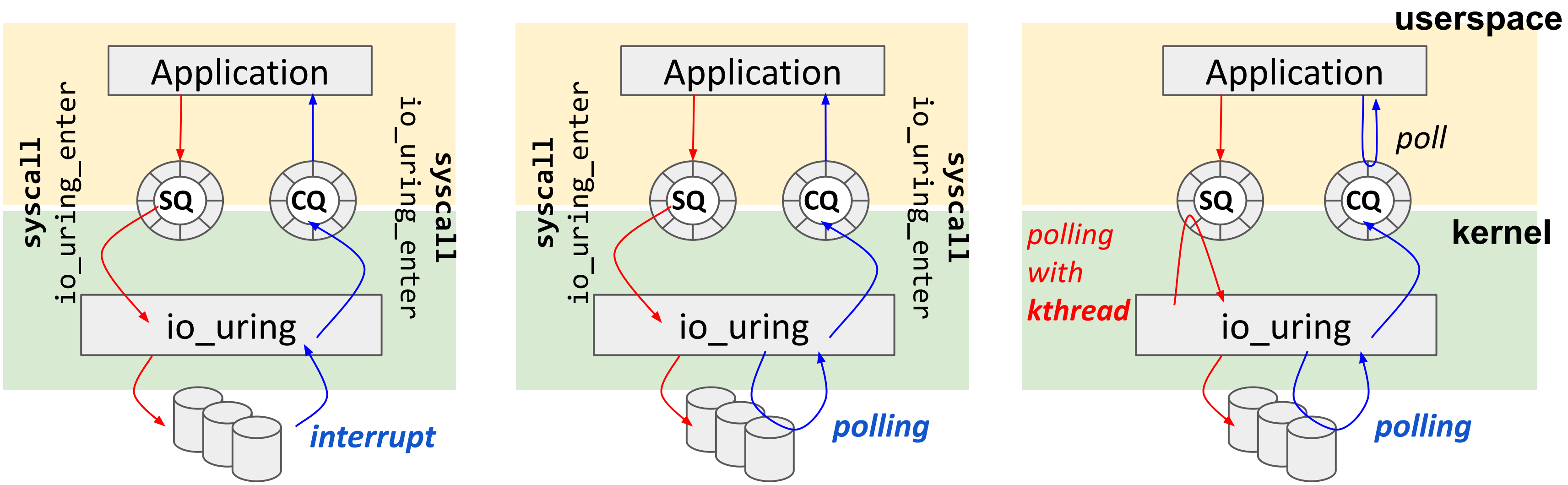
[Part - 2/2] : Zone Namespace Devices (ZNS) Performance Characterization

Krijn Doekemeijer, Nick Tehrany, Bala Chandrasekaran, Matias Bjørling and Animesh Trivedi. **Performance Characterization of NVMe Flash Devices with Zoned Namespaces (ZNS).** 2023 IEEE International Conference on Cluster Computing (**CLUSTER'23**), Santa Fe, NM, USA, 2023, pp. 118-131, doi: https://doi.org/10.1109/CLUSTER52292.2023.00018 .

Workload-NVMe Interaction



Three Modes of io_uring API

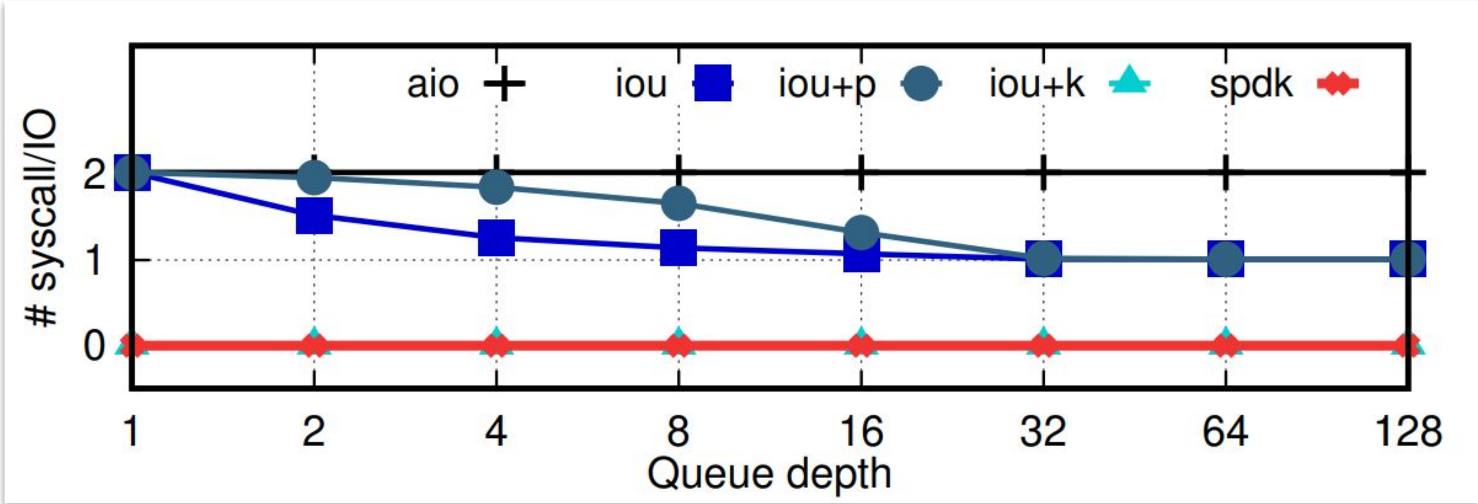


(a) default with syscalls

(b) [**iou+p**] with completion polling

(c) [**iou+k**] with submission polling

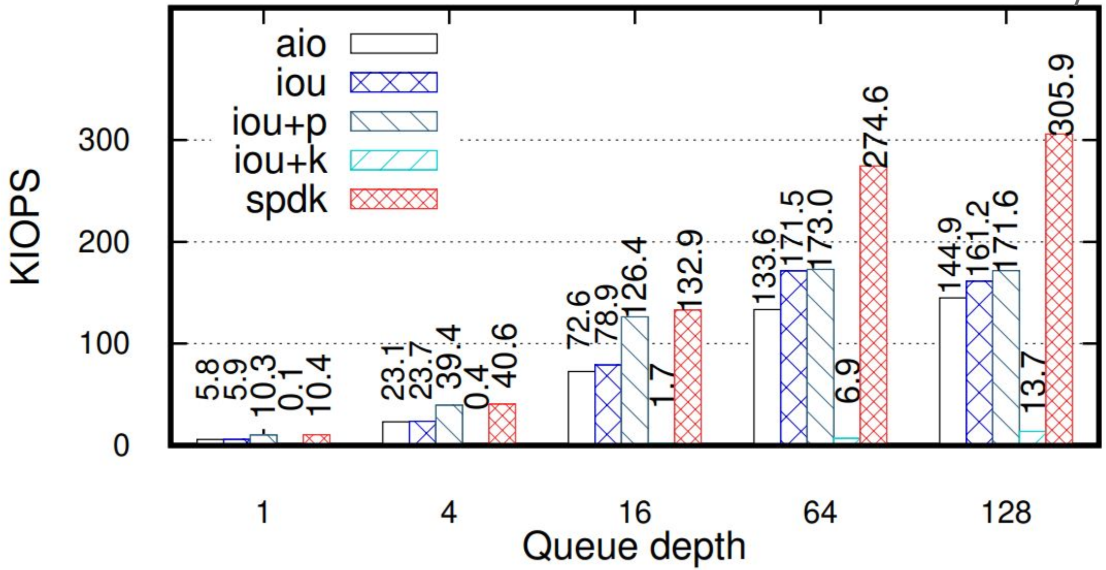
io_uring: System call study



Just like SPDK, io_uring can support a pure polling based, ZERO system calls I/O path!

Results: Efficiency (single CPU core)

Systor'22

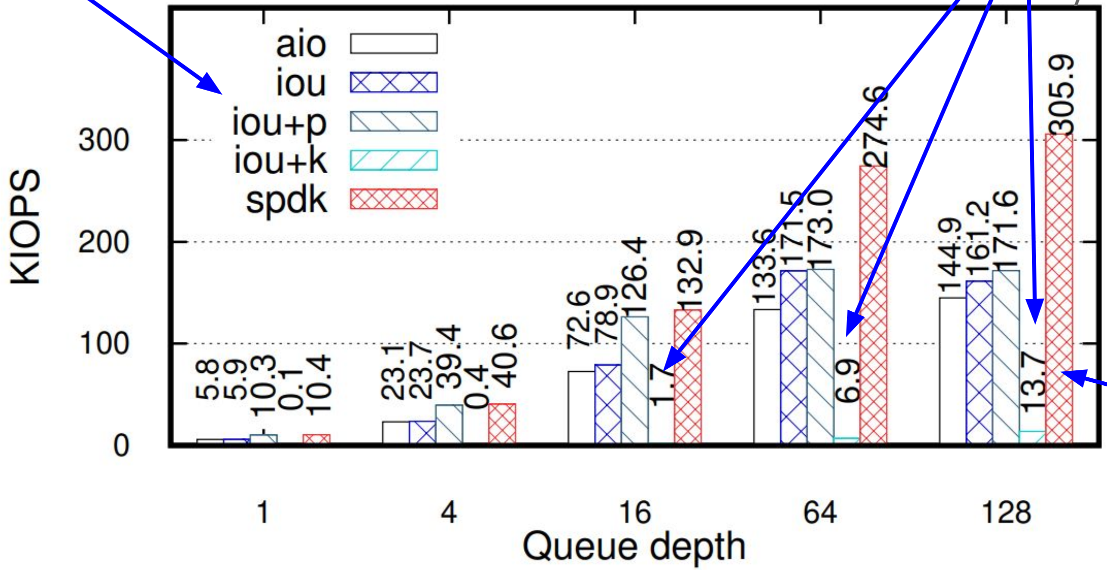


Results: Efficiency (single CPU core)

io_uring sits between libaio and SPDK

Performance collapses with the kernel polling

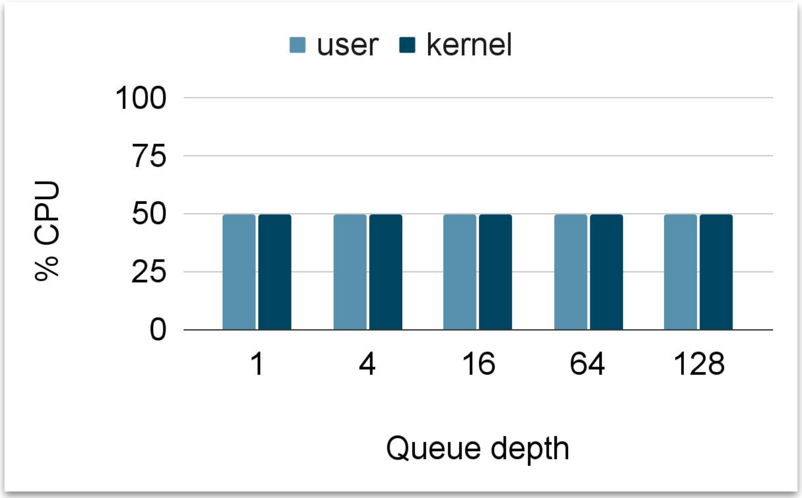
Systor'22



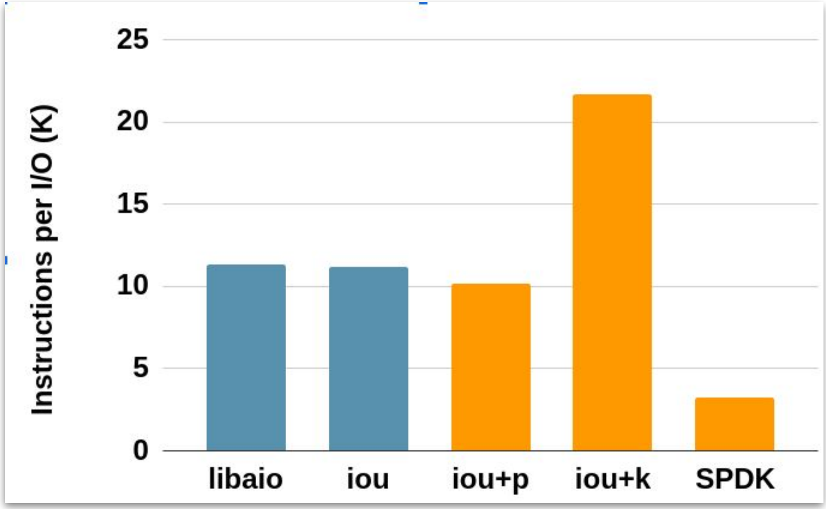
SPDK is the fastest API (still)!

Analysis: CPU Profile

Systor'22



CHEOPS'23

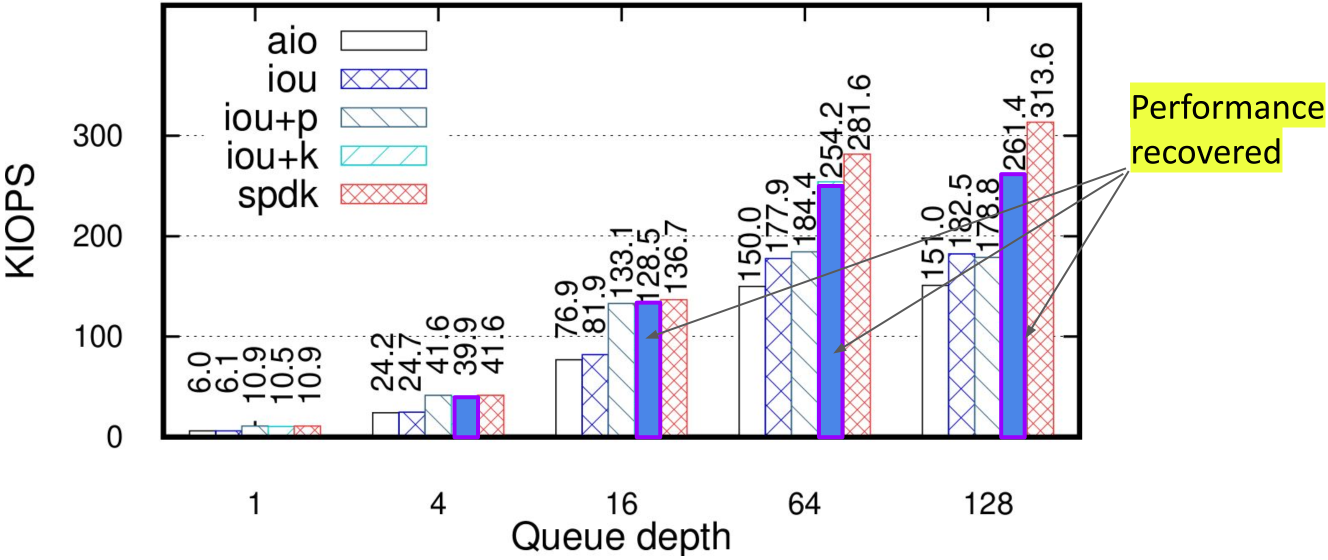


50:50 CPU sharing with polling - ***Careful*!**

SPDK stack is still 5x more CPU efficient

Results: Efficiency with TWO CPU cores

Systor'22

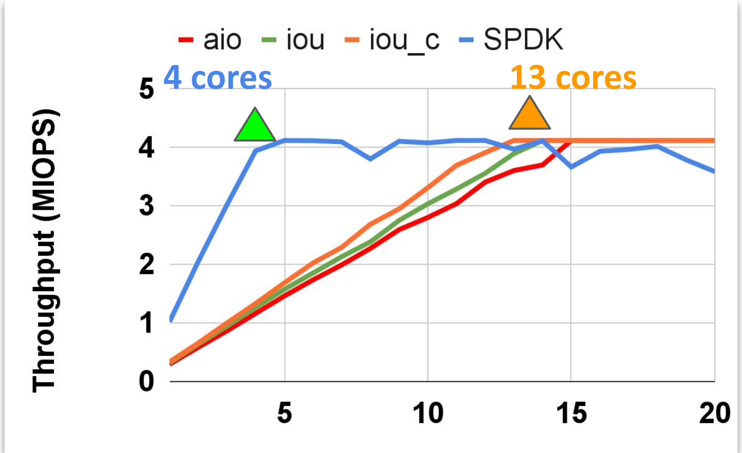


[aio < iou < iou with polling < iou with kernel poll < SPDK]

The "normal performance" order can be resumed (**but,** at the cost of 2x CPU cores)!

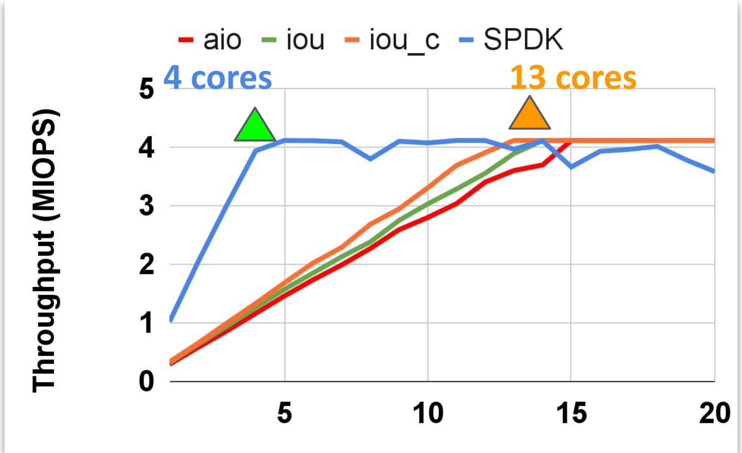
Pure Performance Scaling

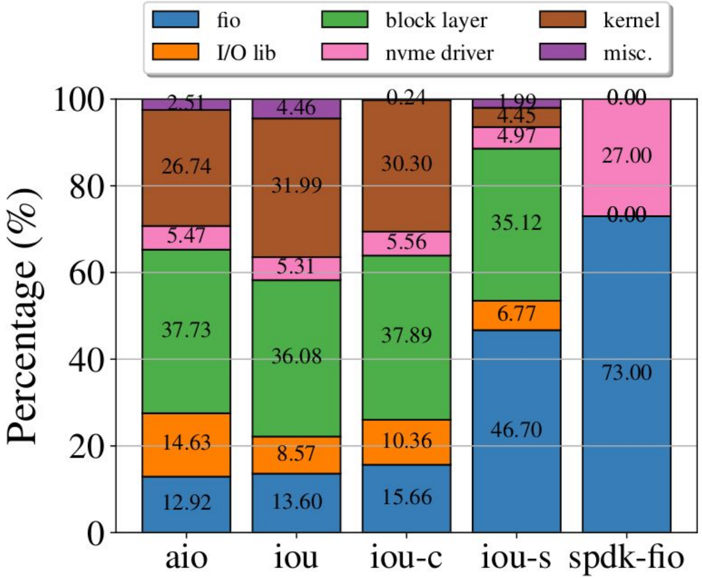
CHEOPS'23



Pure Performance Scaling

CHEOPS'23





- There is a large gap (10x) in the CPU efficiency between SPDK and io_uring stacks
- In the Linux kernel, the block layer is the primary consumer of the CPU cycles

So, What's Wrong with SPDK?

Takes a pure performance-based approach

Highly CPU inefficient (only poll, 100% CPU utilization)

Fragile performance when polling on > #CPU cores

Does not have a file system

Does not have multi-tenancy (only single process)

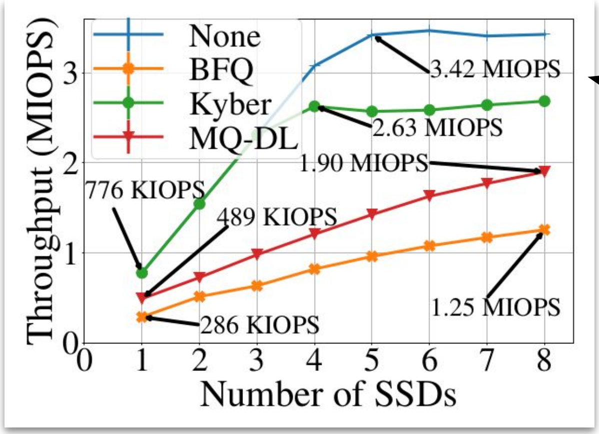
No support for any other kind of devices except NVMe

No provision for the kernel supported services:

- Caching, buffering, security …
- **Importantly: Sharing and I/O Scheduling**

What are the Scheduling Challenges

ICPE'24



High performance scaling with the none I/O scheduler
1.3 - 2.7x slowdown with other schedulers

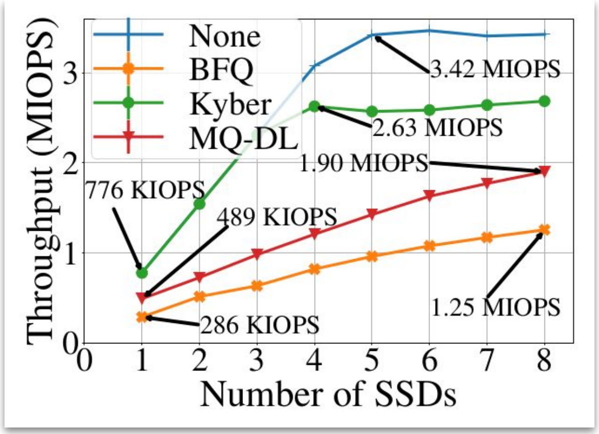
(a) IOPS performance of schedulers;

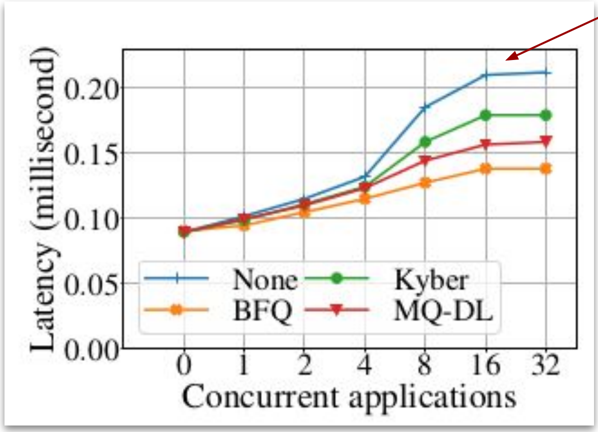
Zebin Ren, Krijn Doekemeijer, Nick Tehrany, Animesh Trivedi. 2024. BFQ, **Multiqueue-Deadline, or Kyber? Performance Characterization of Linux Storage Schedulers in the NVMe Era**, to appear in the 2024 ACM/SPEC International Conference on Performance Engineering (ICPE '24), London, UK.

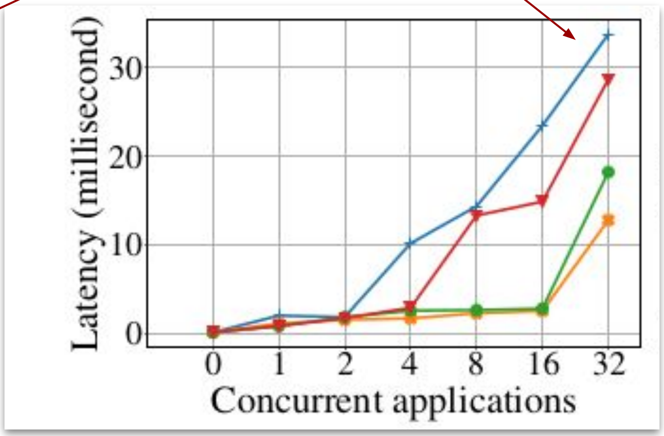
What are the Scheduling Challenges

P95 latencies degradation

ICPE'24



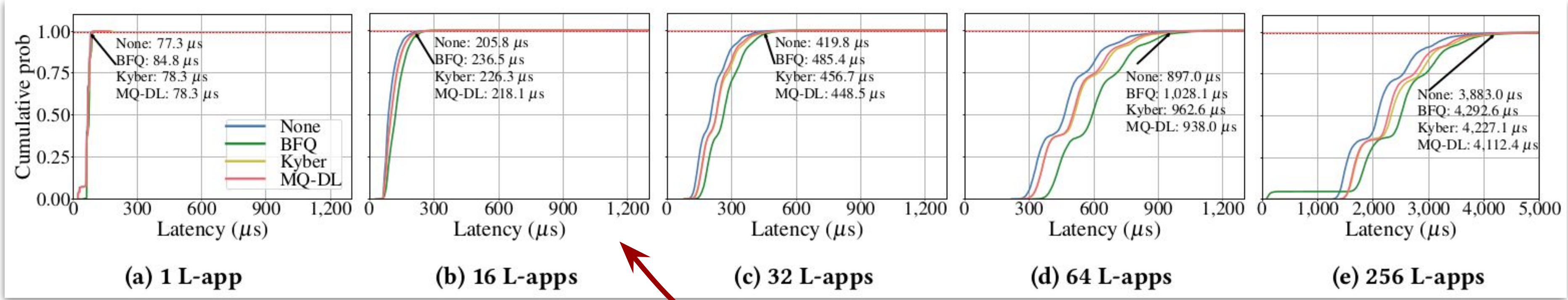




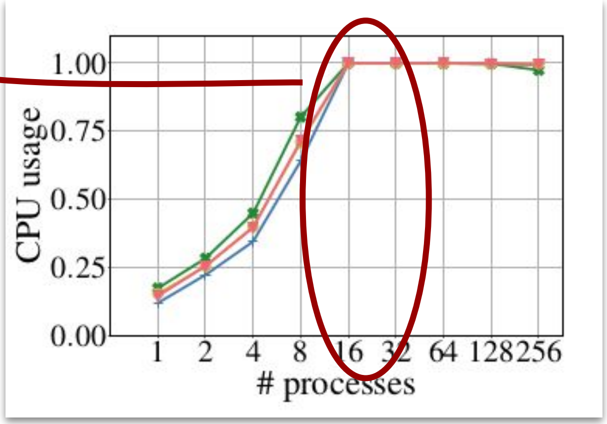
(a) IOPS performance of schedulers; Latency (P95) with background **(b) reads and (c) writes traffic**

- **No scheduling (NOOP) helps with pure performance scaling**
- **No scheduling (NOOP) has poor performance isolation with <u>interfering tasks</u>**

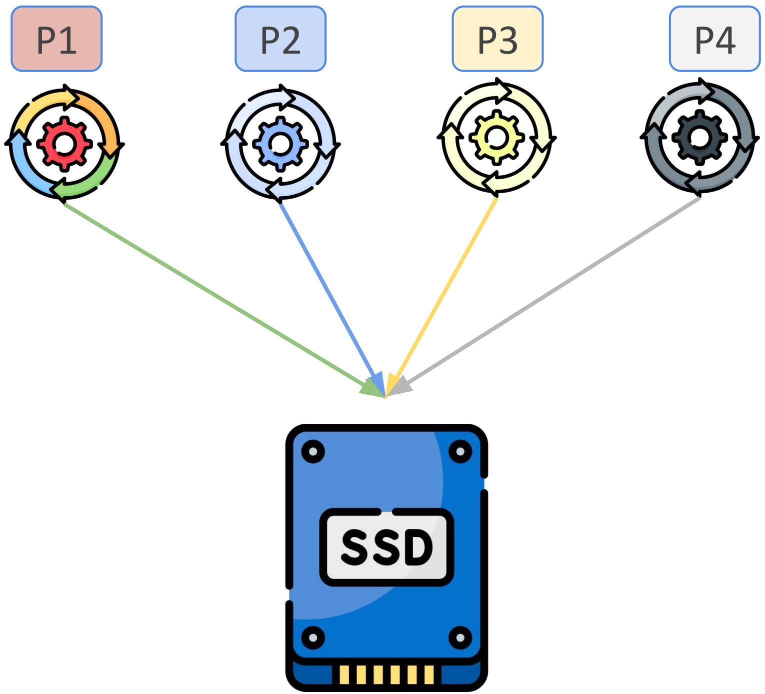
The Tipping Point - the CPU bottleneck



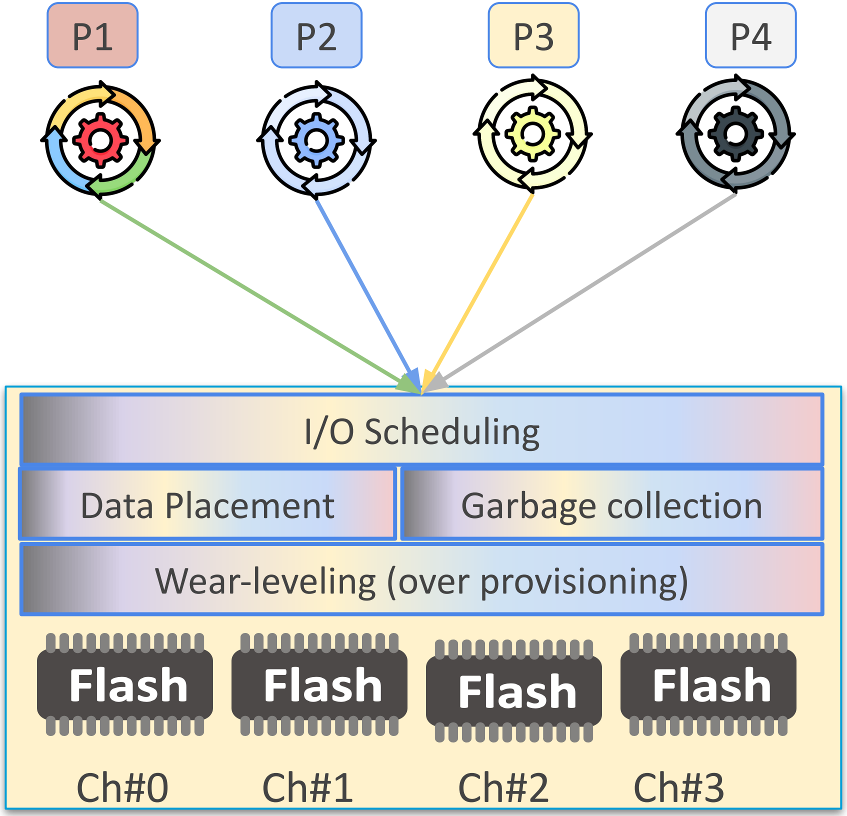
While the CPU is not the bottleneck, all I/O schedulers typically deliver the same performance



Can We Look at the SSD to Get Help for QoS Support?



The Interference Control (or delivering Quality-of-Service)



I/O Scheduling interference and overheads

Inside an SSD

- Mixing of data (lifetime, workloads)
- I/O Scheduling
- Interference from GC
- Over provisioning
- Parallelism management
- …

~~[Part - 1/2] : Study: I/O Performance and Scheduling Overheads~~

~~Diego Didona, Jonas Pfefferle, Nikolas Ioannou, Bernard Metzler, and Animesh Trivedi. 2022. **Understanding modern storage APIs: a systematic study of libaio, SPDK, and io_uring**. In Proceedings of the 15th ACM International Conference on Systems and Storage (SYSTOR '22). Association for Computing Machinery, New York, NY, USA, 120–127. https://doi.org/10.1145/3534056.3534945~~

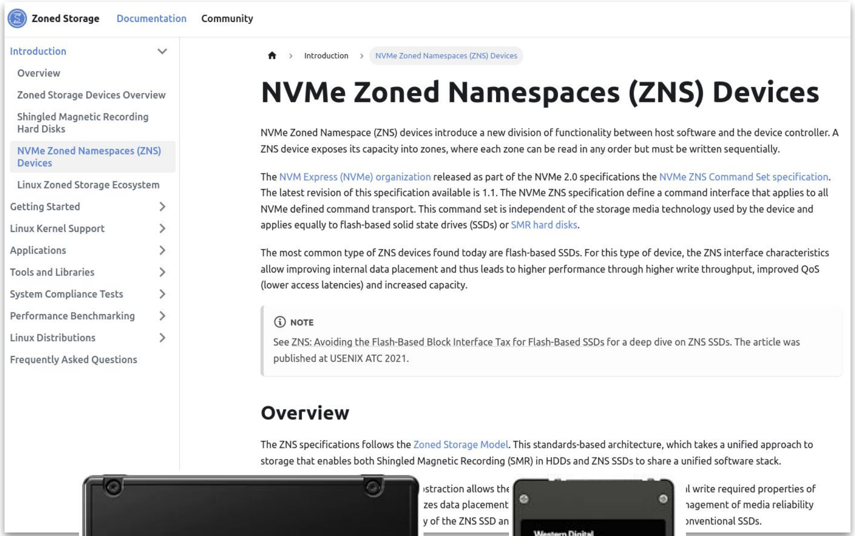
~~Zebin Ren and Animesh Trivedi. 2023. **Performance Characterization of Modern Storage Stacks: POSIX I/O, libaio, SPDK, and io_uring.** In Proceedings of the 3rd Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems (CHEOPS '23). Association for Computing Machinery, New York, NY, USA, 35–45. https://doi.org/10.1145/3578353.3589545~~

~~Zebin Ren, Krijn Doekemeijer, Nick Tehrany, Animesh Trivedi. 2024. BFQ, **Multiqueue-Deadline, or Kyber? Performance Characterization of Linux Storage Schedulers in the NVMe Era**, to appear in the 2024 ACM/SPEC International Conference on Performance Engineering (ICPE '23), London, UK.~~

[Part - 2/2] : Zone Namespace Devices (ZNS) Performance Characterization

Krijn Doekemeijer, Nick Tehrany, Bala Chandrasekaran, Matias Bjørling and Animesh Trivedi. **Performance Characterization of NVMe Flash Devices with Zoned Namespaces (ZNS).** 2023 IEEE International Conference on Cluster Computing (CLUSTER), Santa Fe, NM, USA, 2023, pp. 118-131, doi: https://doi.org/10.1109/CLUSTER52292.2023.00018 .

ZNS: The New Storage Interface and Capabilities


Zoned Storage Documentation Community

Introduction
Overview
Zoned Storage Devices Overview
Shingled Magnetic Recording Hard Disks
NVMe Zoned Namespaces (ZNS) Devices
Linux Zoned Storage Ecosystem
Getting Started
Linux Kernel Support
Applications
Tools and Libraries
System Compliance Tests
Performance Benchmarking
Linux Distributions
Frequently Asked Questions

Introduction › NVMe Zoned Namespaces (ZNS) Devices

NVMe Zoned Namespaces (ZNS) Devices

NVMe Zoned Namespace (ZNS) devices introduce a new division of functionality between host software and the device controller. A ZNS device exposes its capacity into zones, where each zone can be read in any order but must be written sequentially.

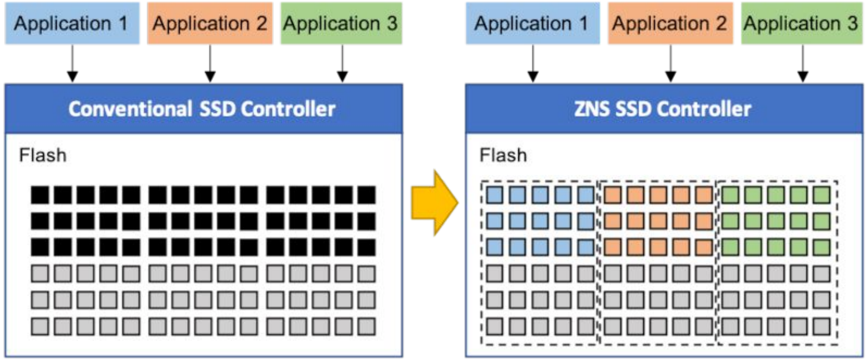
The NVM Express (NVMe) organization released as part of the NVMe 2.0 specifications the NVMe ZNS Command Set specification. The latest revision of this specification available is 1.1. The NVMe ZNS specification define a command interface that applies to all NVMe defined command transport. This command set is independent of the storage media technology used by the device and applies equally to flash-based solid state drives (SSDs) or SMR hard disks.

The most common type of ZNS devices found today are flash-based SSDs. For this type of device, the ZNS interface characteristics allow improving internal data placement and thus leads to higher performance through higher write throughput, improved QoS (lower access latencies) and increased capacity.

NOTE
See ZNS: Avoiding the Flash-Based Block Interface Tax for Flash-Based SSDs for a deep dive on ZNS SSDs. The article was published at USENIX ATC 2021.

Overview

The ZNS specifications follows the Zoned Storage Model. This standards-based architecture, which takes a unified approach to storage that enables both Shingled Magnetic Recording (SMR) in HDDs and ZNS SSDs to share a unified software stack.



Regular SSD: Device controls data placement

ZNS SSD: Applications control data placement in zones

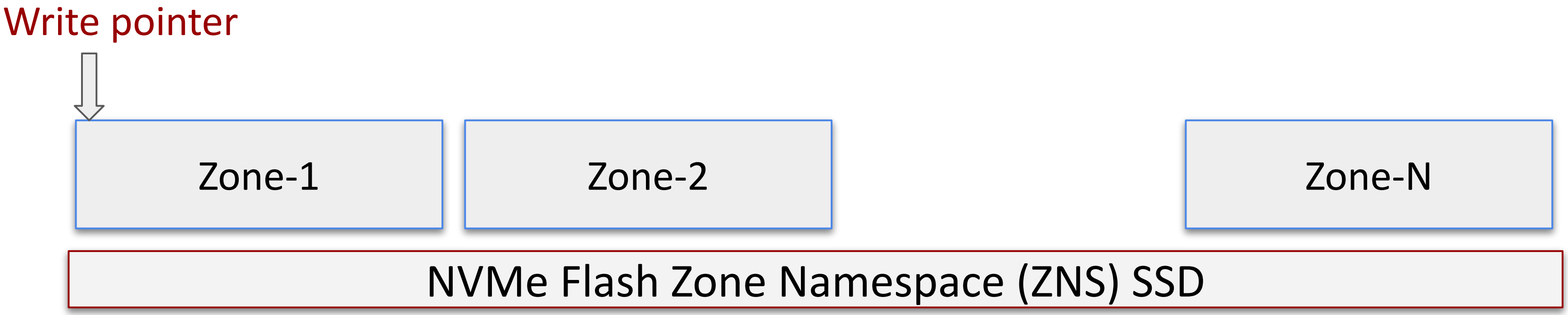
https://zonedstorage.io/docs/introduction/zns

Standardized in the NVMe 1.4, July 2021

Zone Namespace (ZNS) Devices : The Operational Model

A ZNS SSD is divided into Zones

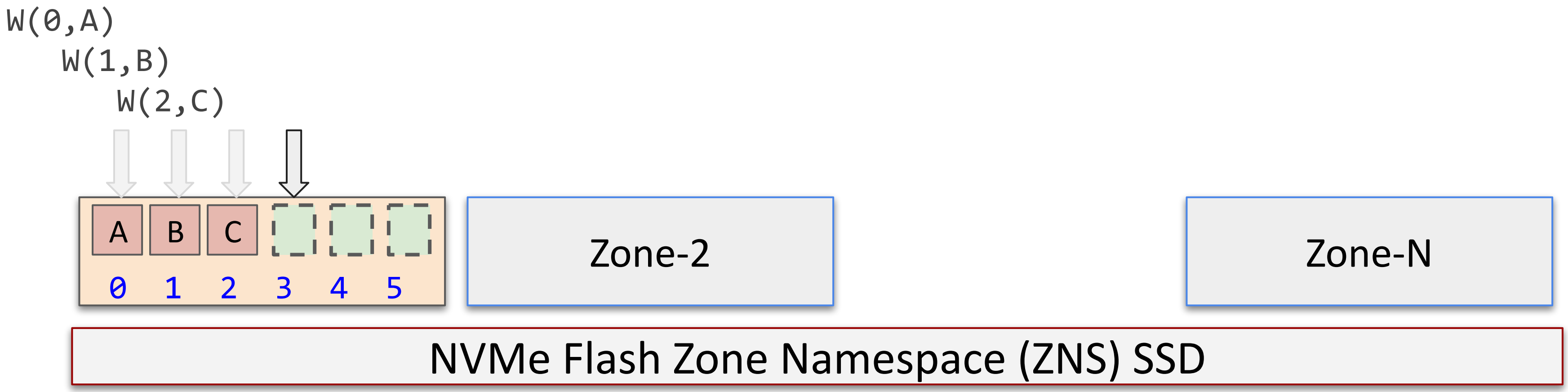
Each zone has its size and a **write pointer**



Zone Namespace (ZNS) Devices : The Operational Model

Each zone must be written sequentially

Limited **intra-zone** parallelism (only 1 write at a time)



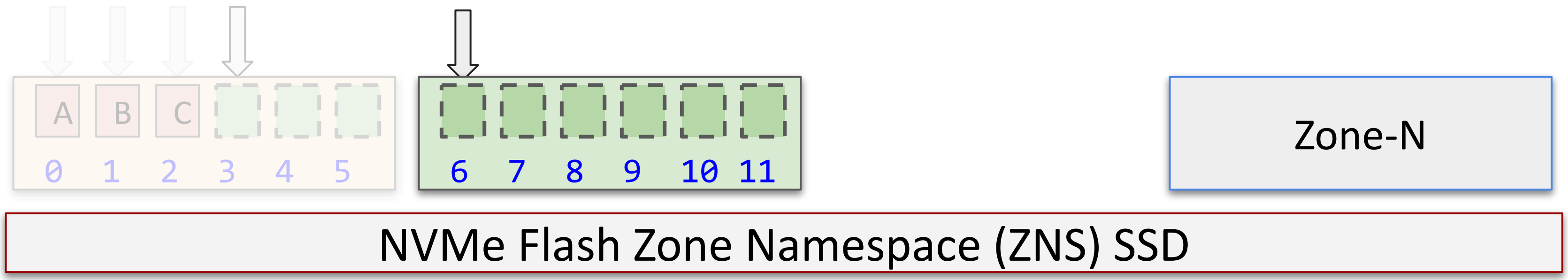
Zone Namespace (ZNS) Devices : The Operational Model

New I/O Command: **Append**

Multiple Append command can be issued to a zone (high **intra-zone** parallelism)

A(Z-2,M)
A(Z-2,N)
A(Z-2,O)

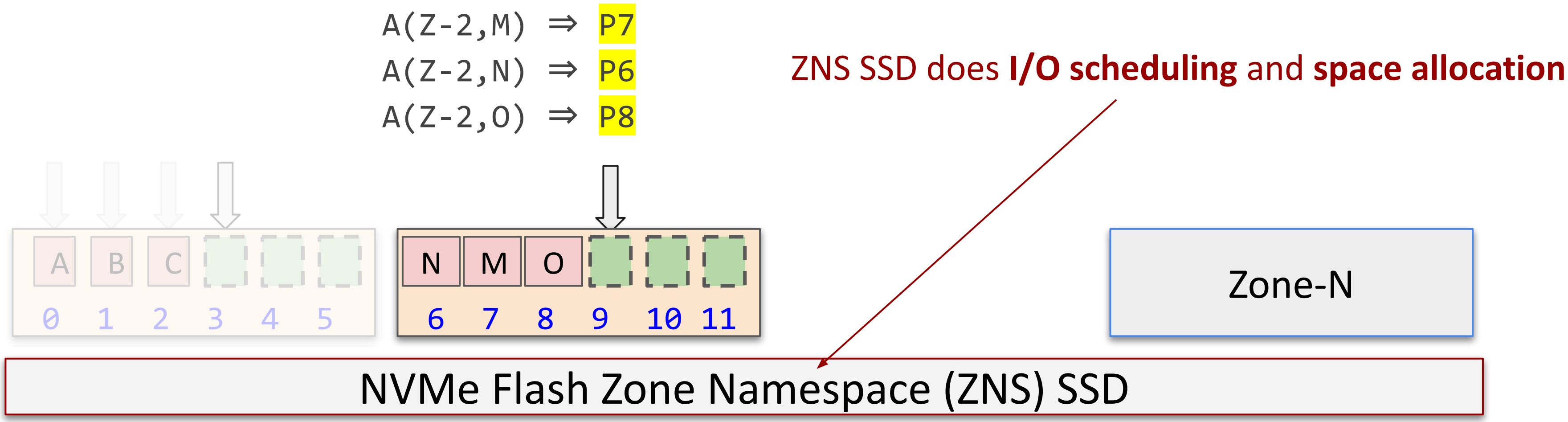
"Append M, N and O to Zone-2 (anywhere)"



Zone Namespace (ZNS) Devices : The Operational Model

New I/O Command: **Append**

Multiple Append command can be issued to a zone (high **intra-zone** parallelism)

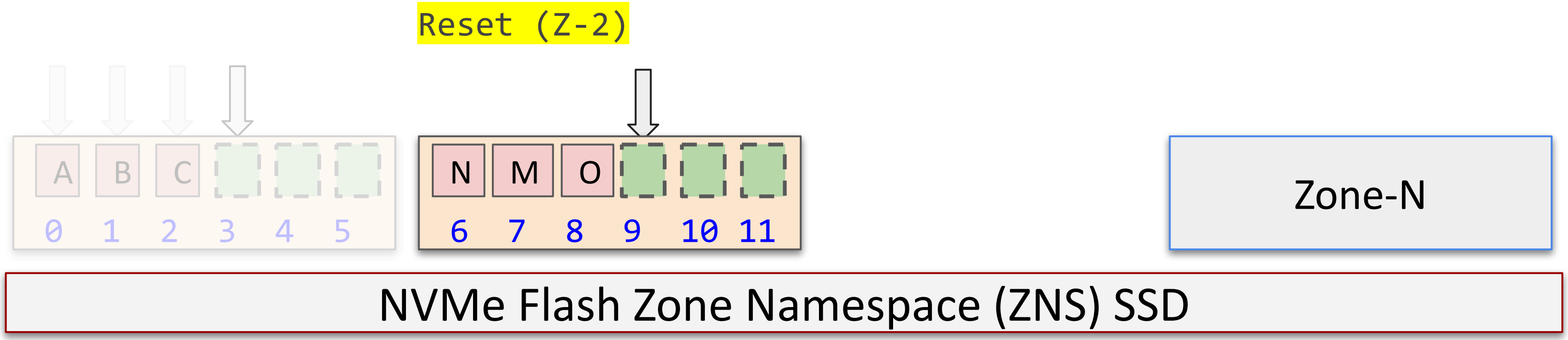


Zone Namespace (ZNS) Devices : The Operational Model

New zone-management commands: **Finish** and **Reset**

Finish: makes it read-only (release write resources)

Reset: garbage collect the zone

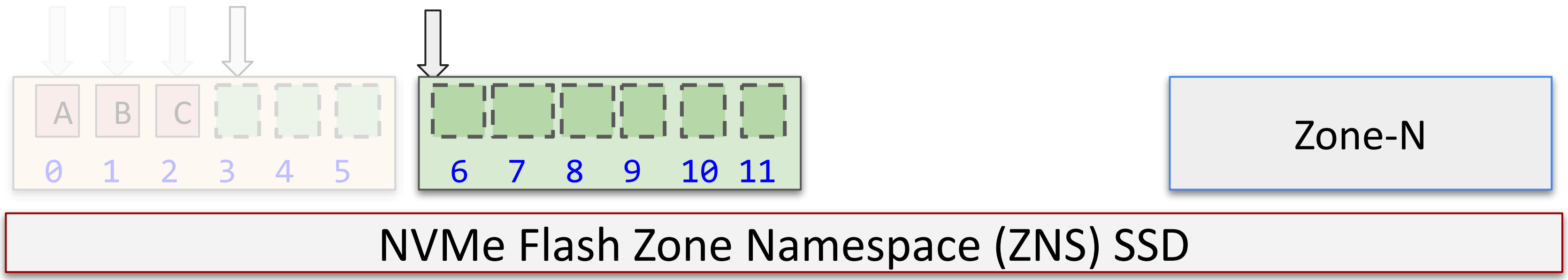


Zone Namespace (ZNS) Devices : The Operational Model

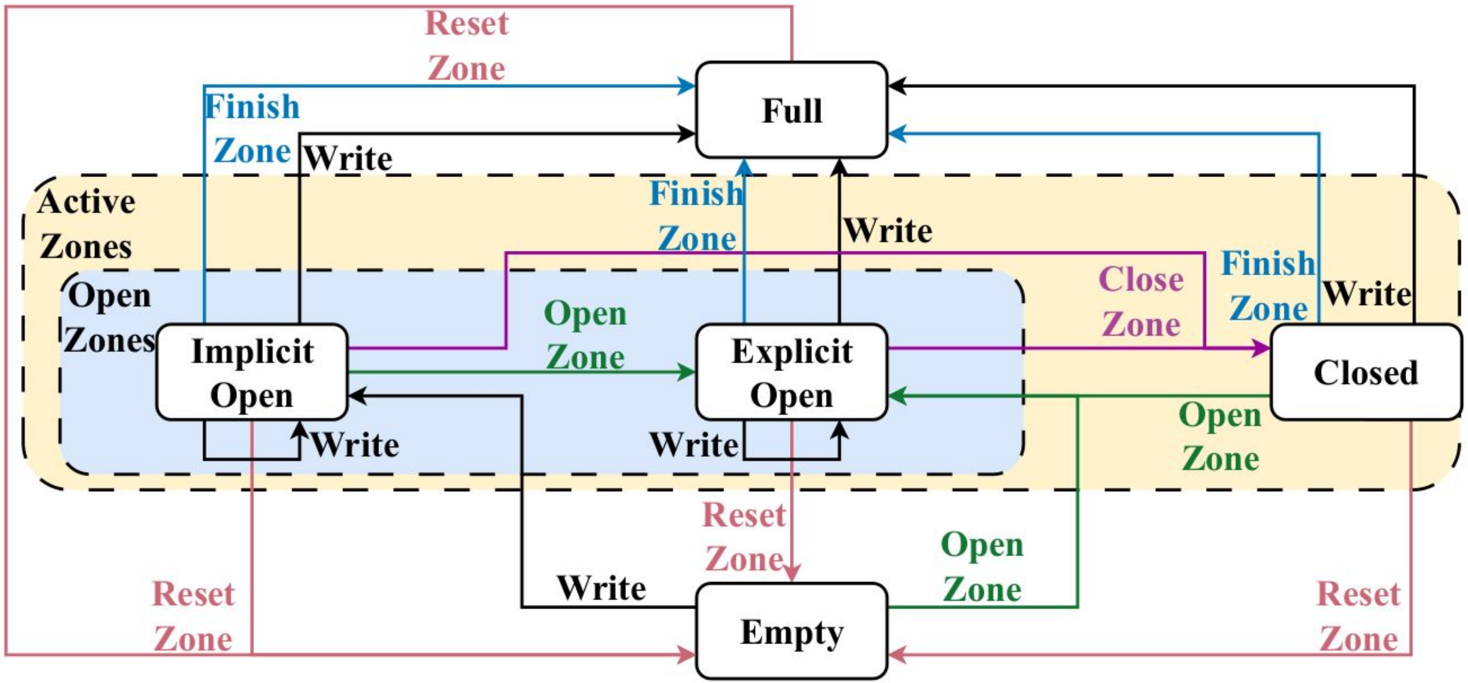
New zone-management commands: **Finish** and **Reset**

Finish: makes it read-only (release write resources)

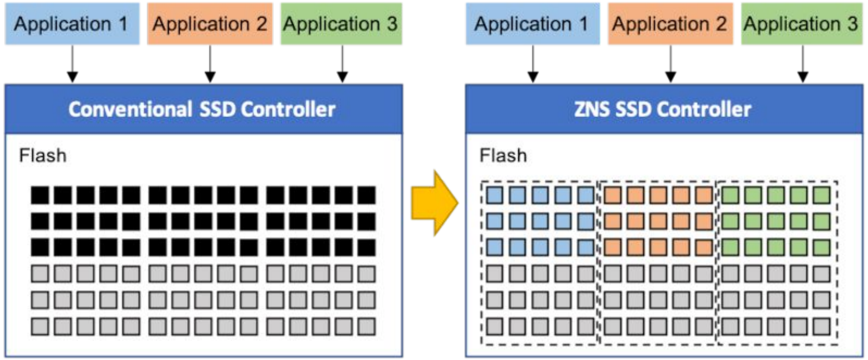
Reset: garbage collect the zone



Zone Namespace (ZNS) Devices: The State Machine



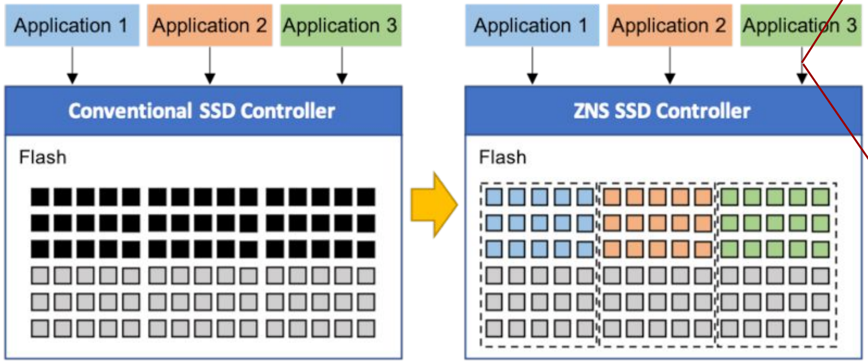
State of the ZNS Software



Regular SSD: Device controls data placement

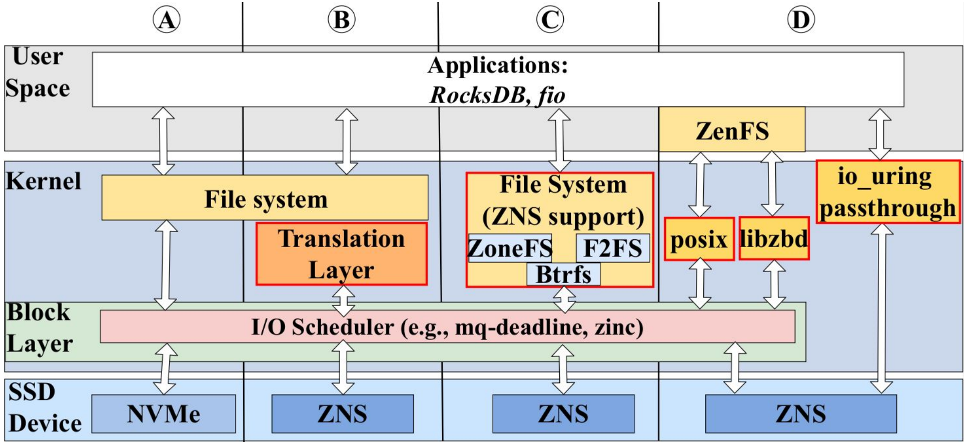
ZNS SSD: Applications control data placement in zones

State of the ZNS Software



Regular SSD: Device controls data placement

ZNS SSD: Applications control data placement in zones



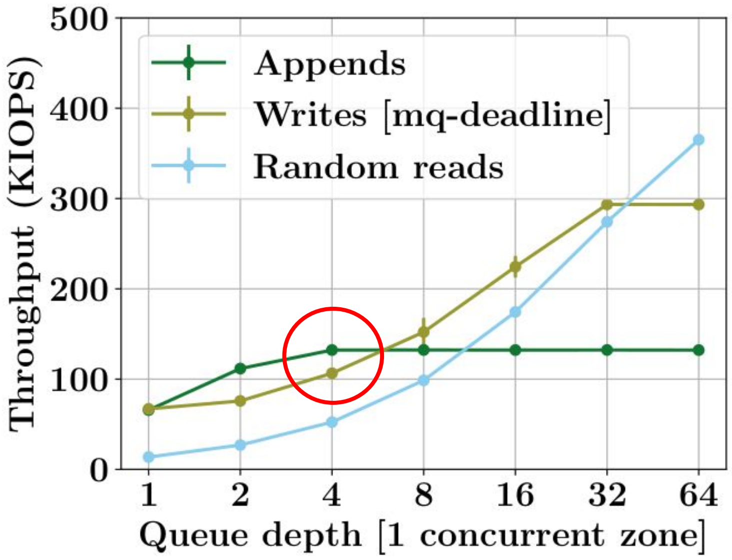
Understanding NVMe Zoned Namespace (ZNS) Flash SSD Storage Devices, Nick Tehrany, Animesh Trivedi, https://arxiv.org/abs/2206.01547 (2022).

Idea: Different zones helps to isolate workloads from each other and better Quality-of-Service (QoS)

But: There are multiple ways ZNS devices can be integrated

- Should I use **Append** or **Write**? How do I manage **parallelism**? Intra-zone or Inter-zone?
- What is the cost of **Reset** and **Finish**? And the state machine implementation
- Do ZNS SSDs deliver **isolation**?

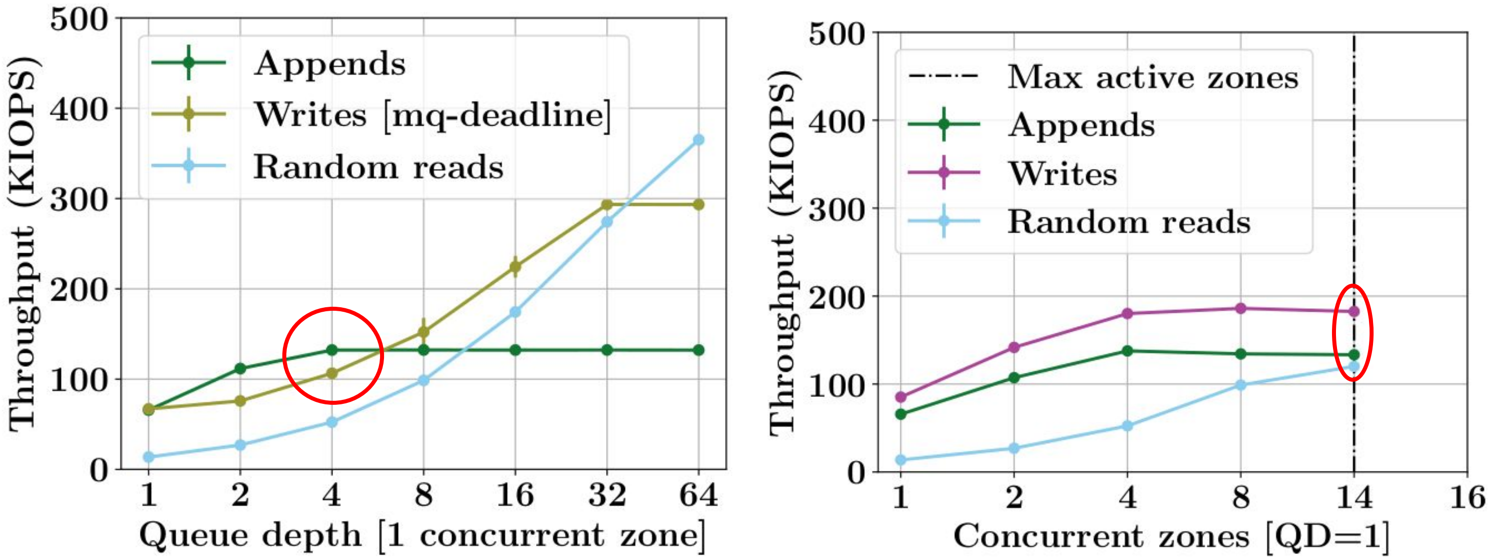
Result [1 / 3]: Write vs Append Parallelism Management



mq-deadline scheduler merges adjacent writes

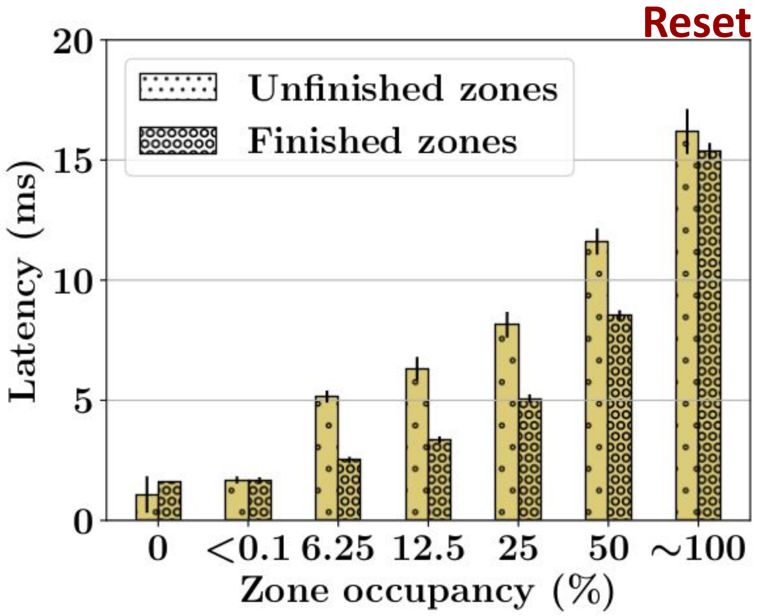
Single Zone Parallelism (intra-zone)

Result [1 / 3]: Write vs Append Parallelism Management

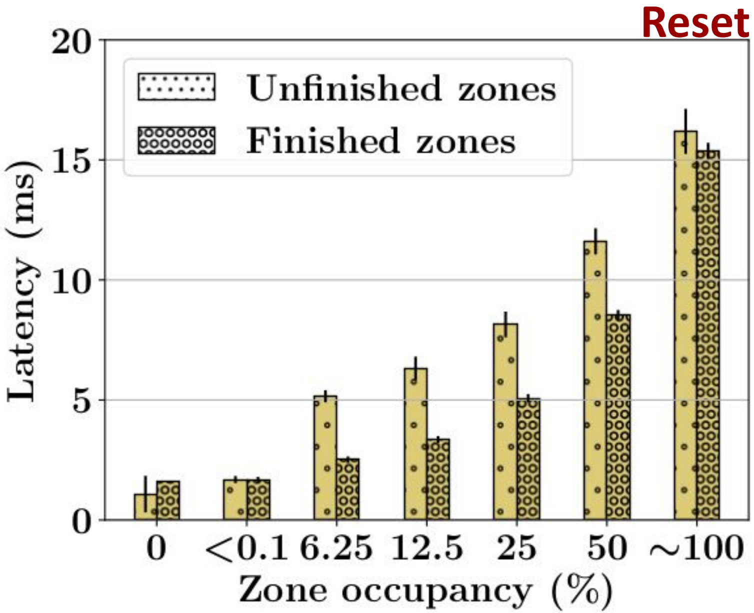


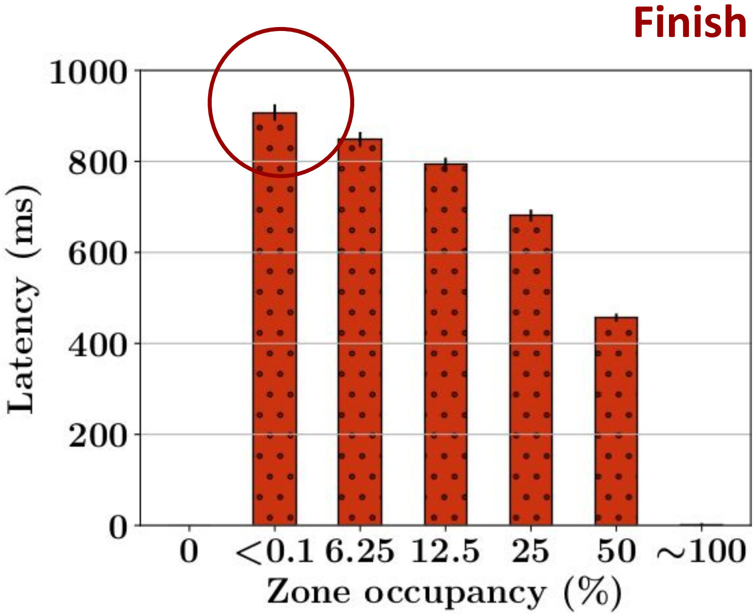
- Intra-Zone parallelism has higher performance
- Writes have better performance scalability than Appends (!)
- Append scalability is independent of intra- or inter-zone, but limited in performance

Result [2 / 3]: The Cost of Reset and Finish Operations



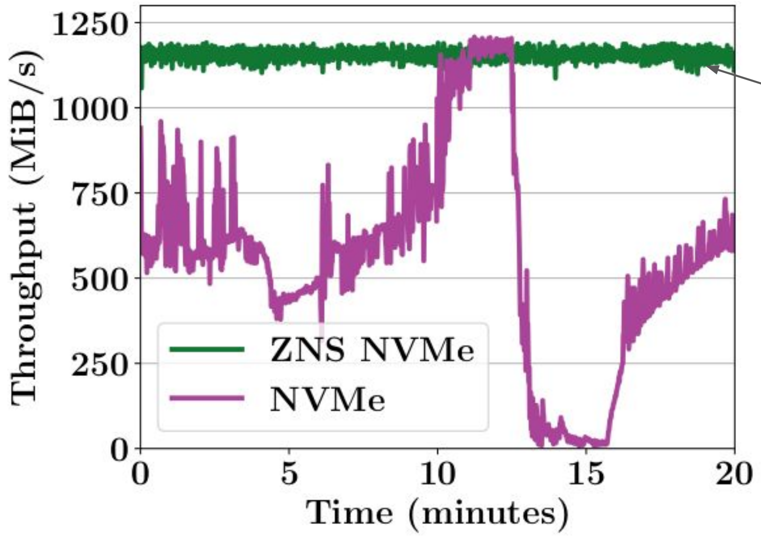
Result [2 / 3]: The Cost of Reset and Finish Operations





- The zone utilization --- Very important factor
- Finish is an extremely expensive operation (100 - 1,000s of milliseconds)
- Leverage intra-zone parallelism (*minimize half-written zones*)

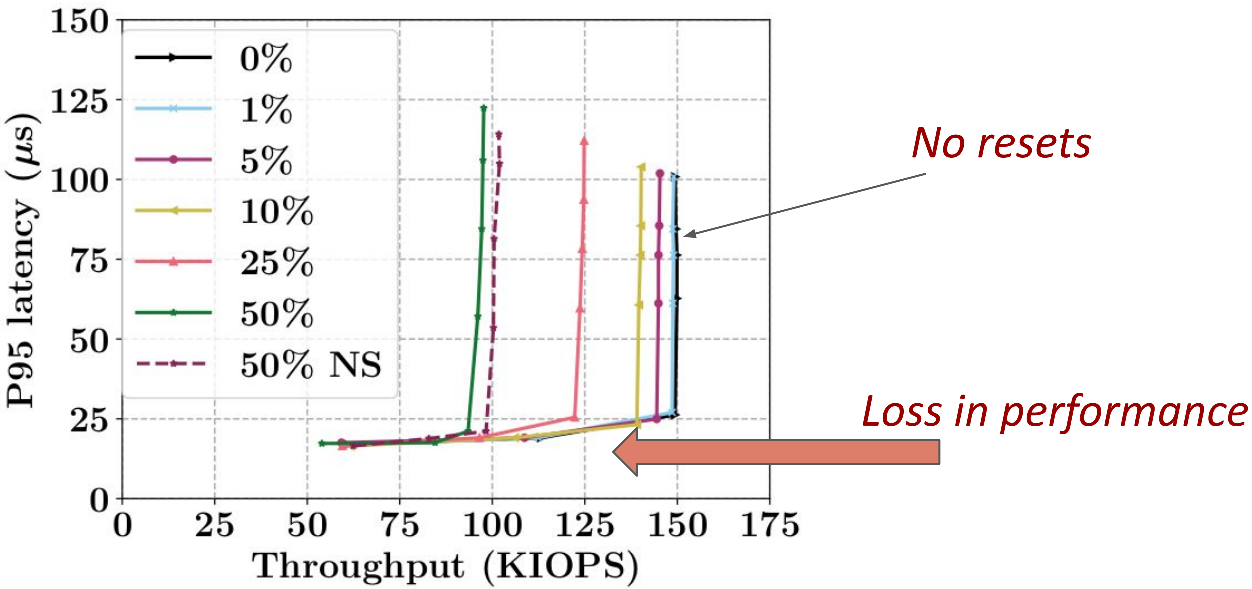
Result [3 / 3]: Read-Write Isolation on ZNS



Stable performance

- ZNS provides good read-write isolation when operating on multiple zones
- Stable performance (in comparison to NVMe)

Do Reset Commands Interfere with I/O Operations?



Initial results: Yes … part of an active research now :)

[Part - 1/2] : Study: I/O Performance and Scheduling Overheads

Diego Didona, Jonas Pfefferle, Nikolas Ioannou, Bernard Metzler, and Animesh Trivedi. 2022. **Understanding modern storage APIs: a systematic study of libaio, SPDK, and io_uring**. In Proceedings of the 15th ACM International Conference on Systems and Storage (SYSTOR '22). Association for Computing Machinery, New York, NY, USA, 120–127. https://doi.org/10.1145/3534056.3534945

Zebin Ren and Animesh Trivedi. 2023. **Performance Characterization of Modern Storage Stacks: POSIX I/O, libaio, SPDK, and io_uring.** In Proceedings of the 3rd Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems (CHEOPS '23). Association for Computing Machinery, New York, NY, USA, 35–45. https://doi.org/10.1145/3578353.3589545

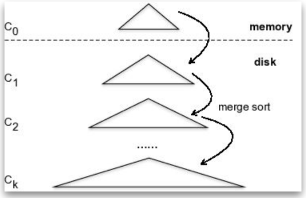
Zebin Ren, Krijn Doekemeijer, Nick Tehrany, Animesh Trivedi. 2024. BFQ, **Multiqueue-Deadline, or Kyber? Performance Characterization of Linux Storage Schedulers in the NVMe Era**, to appear in the 2024 ACM/SPEC International Conference on Performance Engineering (ICPE '23), London, UK.

[Part - 2/2] : Zone Namespace Devices (ZNS) Performance Characterization

Krijn Doekemeijer, Nick Tehrany, Bala Chandrasekaran, Matias Bjørling and Animesh Trivedi. **Performance Characterization of NVMe Flash Devices with Zoned Namespaces (ZNS).** 2023 IEEE International Conference on Cluster Computing (CLUSTER), Santa Fe, NM, USA, 2023, pp. 118-131, doi: https://doi.org/10.1109/CLUSTER52292.2023.00018 .

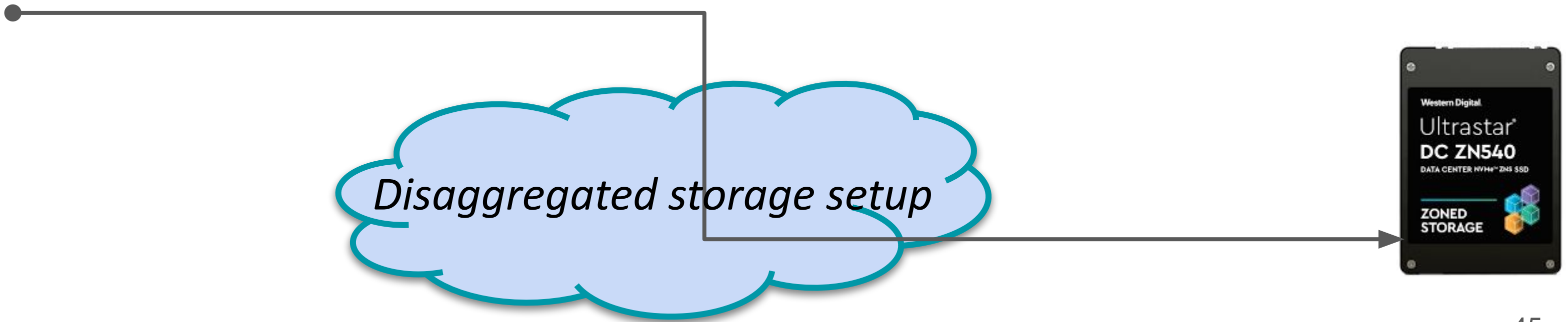
Delivering QoS in a Distributed Setting

RocksDB



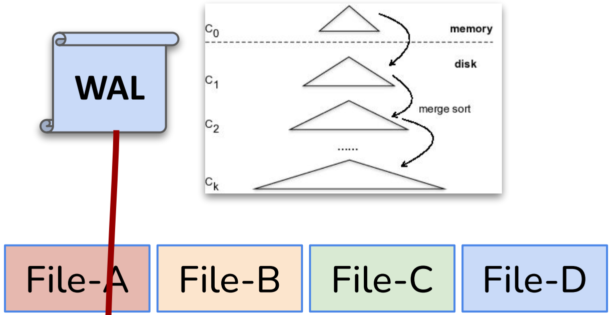
Read, Write, Append, Reset, Finish, Close, Open,

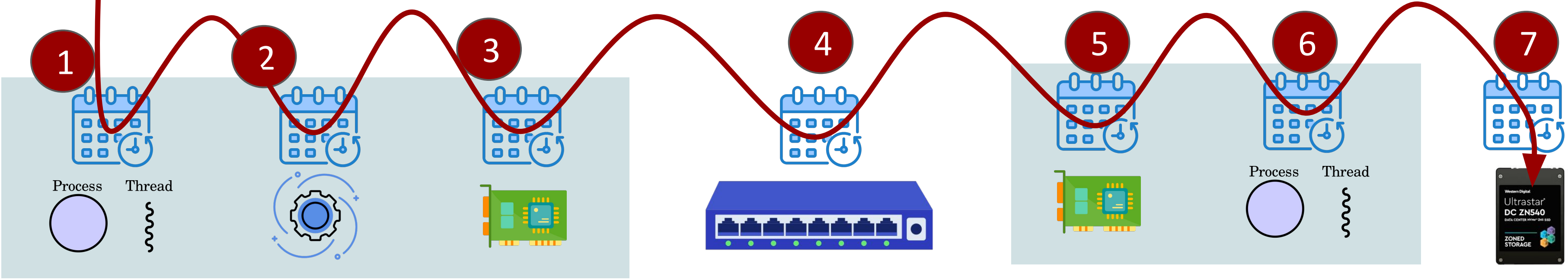
File-A	File-B	File-C	File-D



Delivering QoS in a Distributed Setting

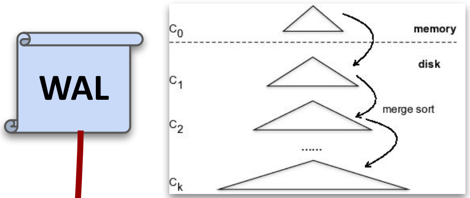
RocksDB





Delivering QoS in a Distributed Setting

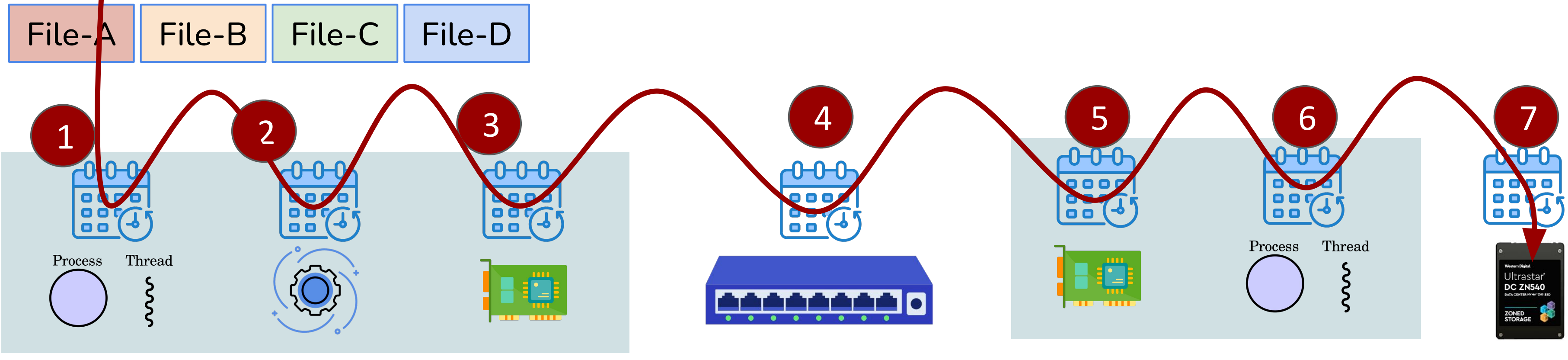
RocksDB



End-to-End abstraction for QoS:

Co-design workload-level storage-network data abstractions

Co-schedule them together (gang scheduling, co-flows)



Conclusion

Vision: build your favorite workload-specialized data structure I/O stack!

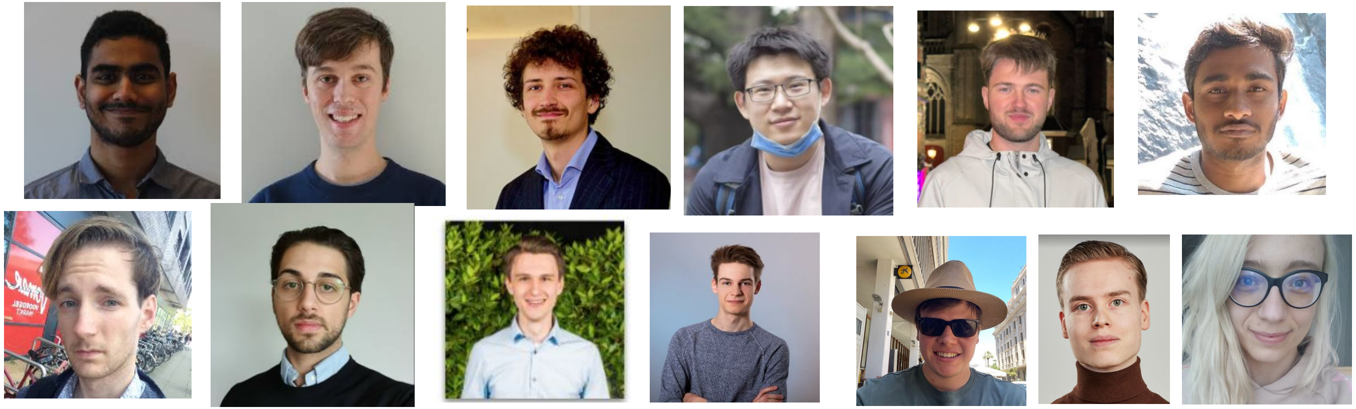
The era of workload-specialized storage stacks is here

We are exploring:

- Workload-specialized storage software abstractions
- Mapping software interfaces to the available hardware interfaces
  - NVMe ZNS, KV-SSD, CXL (emerging)

WiP: Co-scheduling (Network + ZNS Storage) ⇒ End-to-End QoS

Thank you!

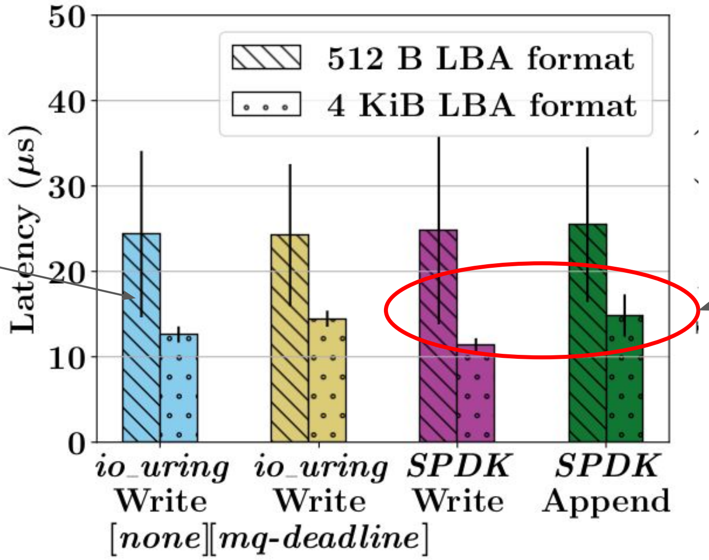


(past and present students)

Acknowledgments: The work presented here is generously funded by the Dutch Research Council (NWO) grants (OCENW.KLEIN.561 and OCENW.KLEIN.209), GFP 6G FNS, and the EU funded projects MCSA-RISE Cloudstars, Horizon GraphMassivizer, VU PhD innovation program, with ZNS hardware donation and support from Western Digital.

Result [1 / 4]: Write vs Append Latencies

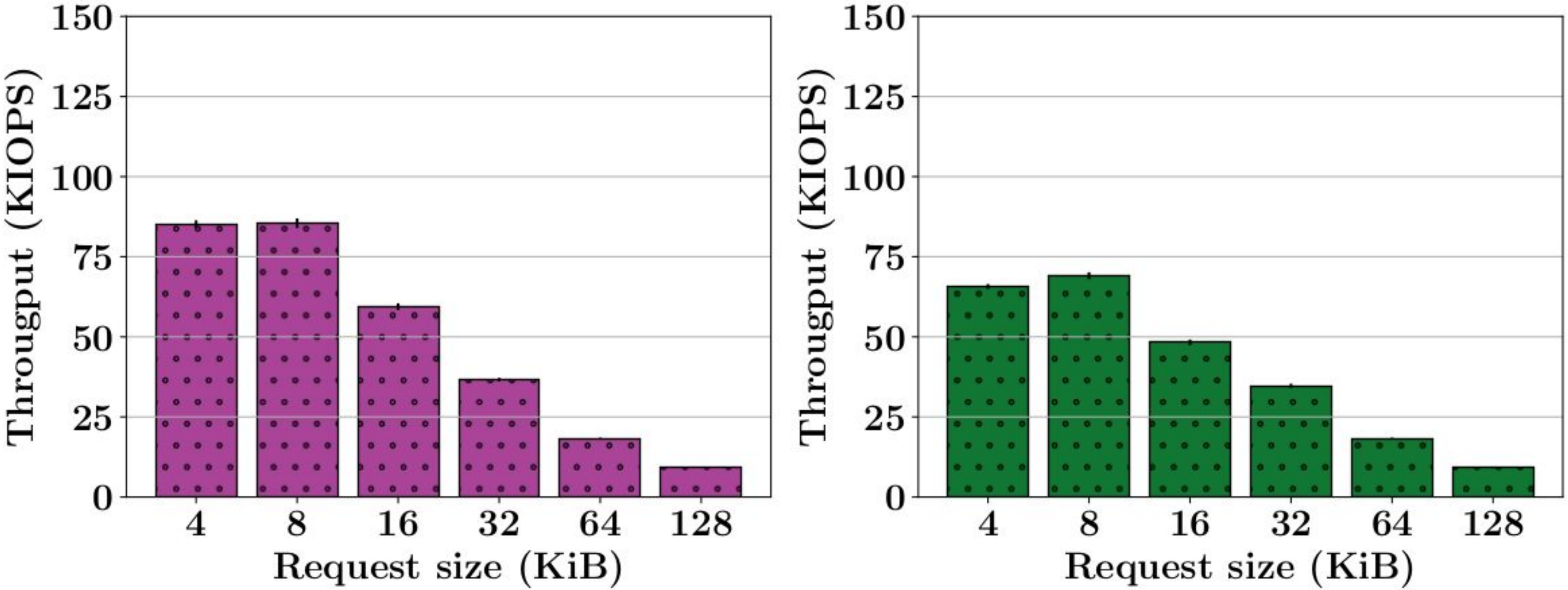
Large gap in the LBA format



Writes lower than Append

- 4KiB block size has lower latencies (up to 2x)
- Writes have lower latencies than Append operations in our experiments
- SPDK has lower latencies than the Linux I/O stack (none, mq-deadline)

Write and Append: Bandwidth



(a) *write* (b) *append*