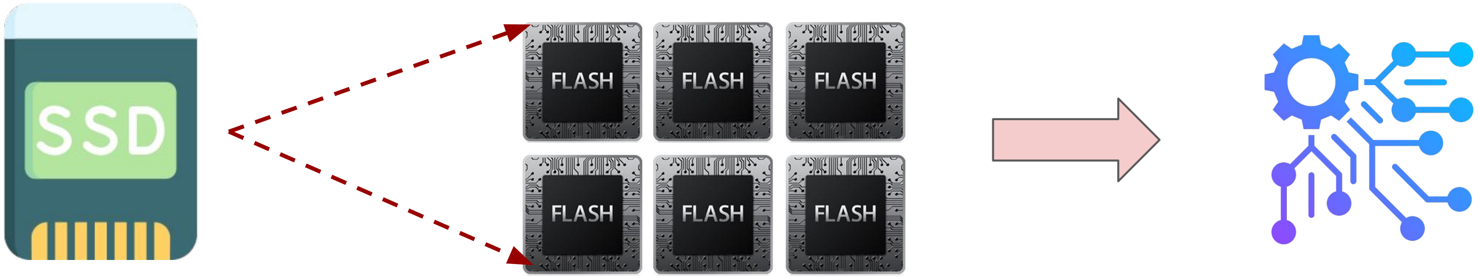
zns-tools: An eBPF-powered, Cross-Layer Storage Profiling Tool for NVMe ZNS SSDs

Nick Tehrany, Krijn Doekemeijer, and **Animesh Trivedi**
Vrije Universiteit Amsterdam

https://github.com/stonet-research/zns-tools

Cloud Computing + Storage

Challenges with SSDs



Read anywhere
Write anywhere

Write **once**
Write **sequentially**
Erase (or RESET)
Garbage collection (GC)
Finite P/E cycle
Errors
Chip management

Complexity
Unpredictability
Interference

(all hidden from us)

Lots of research

Rise of the Open SSD Interfaces

Idea: make SSD internal state and operations more …

- Visible
- Under the control of the host system software (OS)

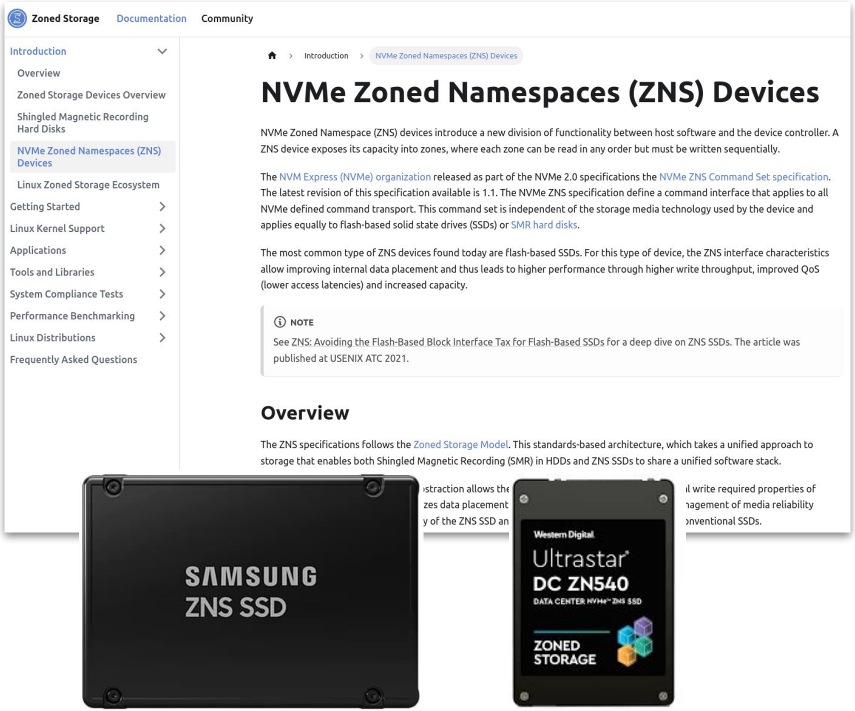
Multiple design points: Software-defined Flash, Open Channel SSD, Stream SSD…

Today: Zone Namespace SSDs or ZNS SSD

Benefits:

- An OS can decide how store data and manage SSD storage
- An OS can plan (resource management) for a particular QoS
- An OS can explain why performance suffered
- Simplified SSD internals

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.

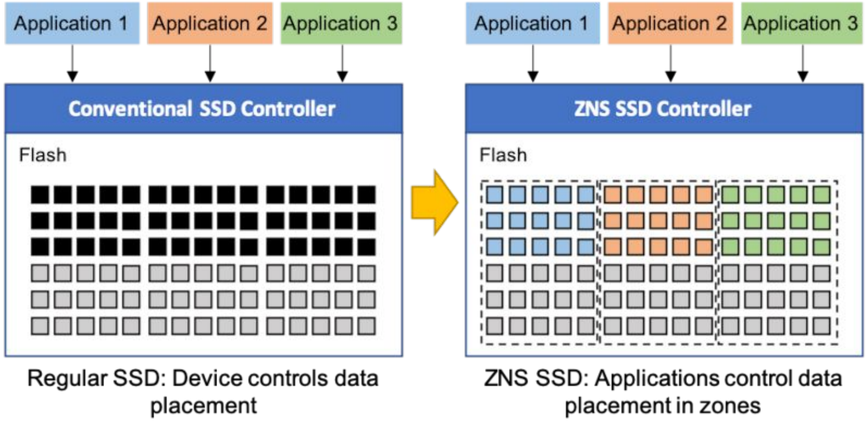
The NVM Express (NVMe) organization released as part of the NVMe 2.0 specifications the NVMe ZNS Command Set specification. The latest revision of this specification available is 1.1. The NVMe ZNS specification define a command interface that applies to all NVMe defined command transport. This command 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.



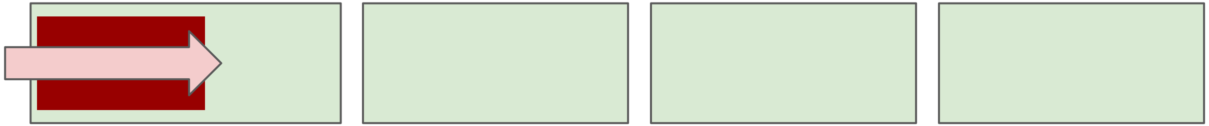
https://zonedstorage.io/docs/introduction/zns

Standardized in the NVMe 1.4, July 2021

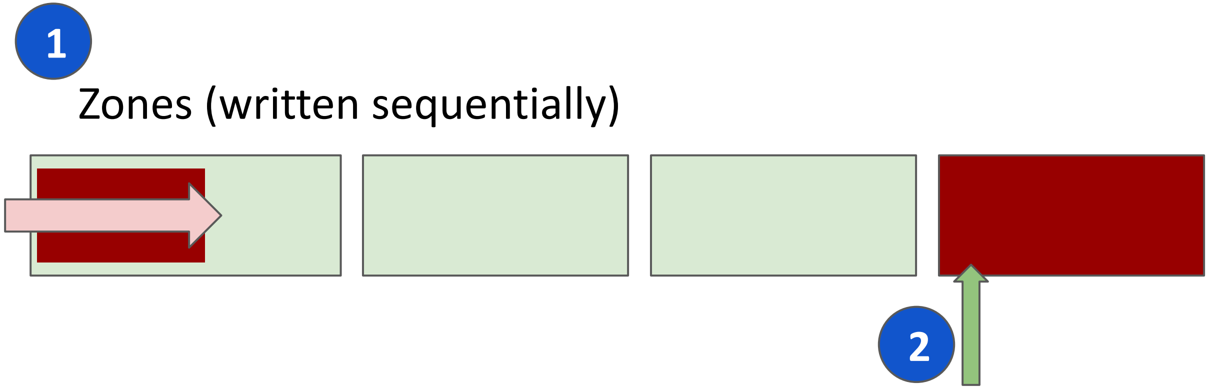
ZNS Introduction: New I/O and Management Operations

1

Zones (written sequentially)



ZNS Introduction: New I/O and Management Operations



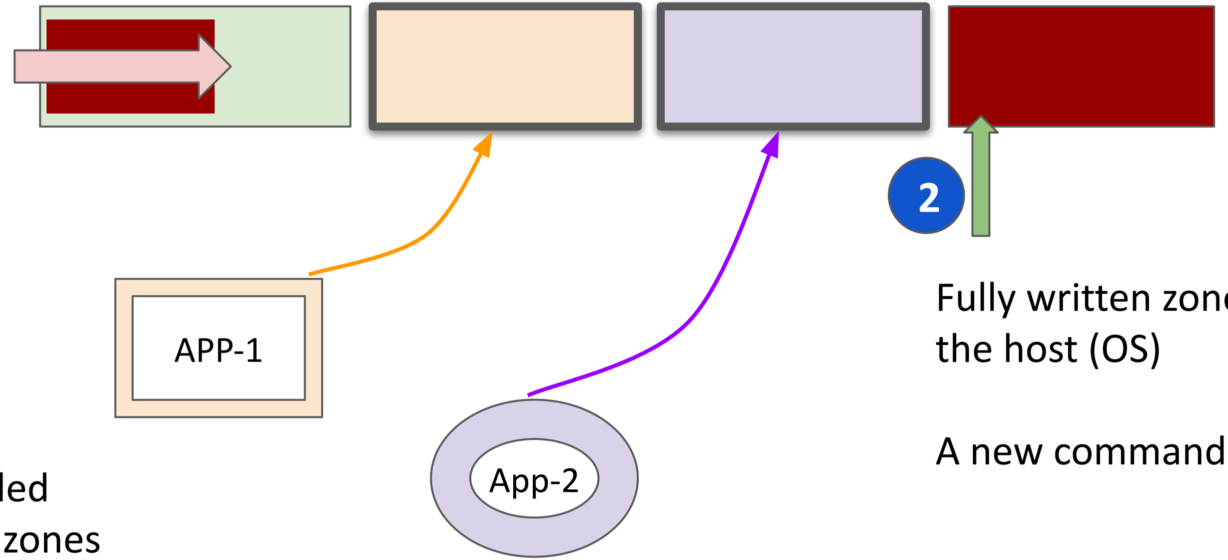
Fully written zone is Erased by the host (OS)

A new command : RESET

ZNS Introduction: New I/O and Management Operations

1

Zones (written sequentially)



2

Fully written zone is Erased by the host (OS)

A new command : RESET

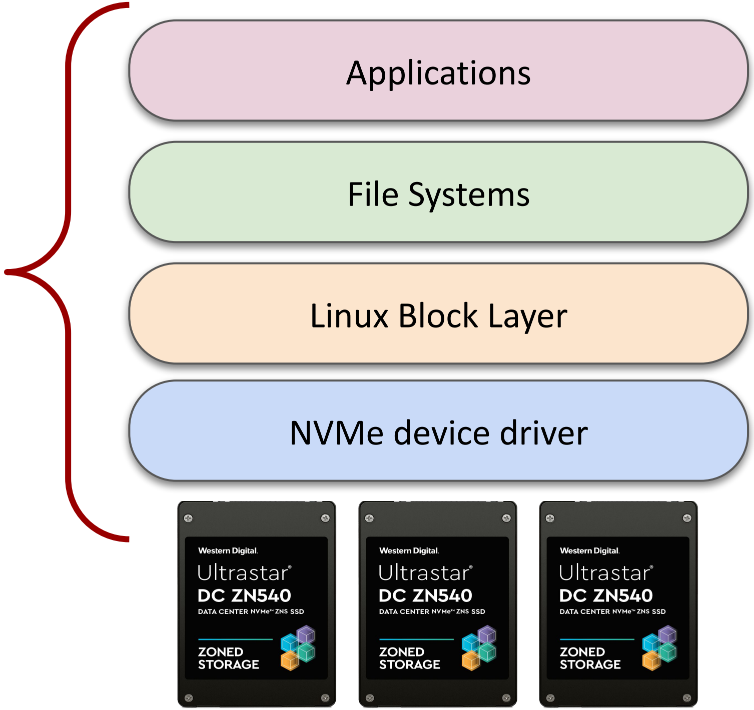
3

Application-controlled data placements in zones

What Does This Mean?

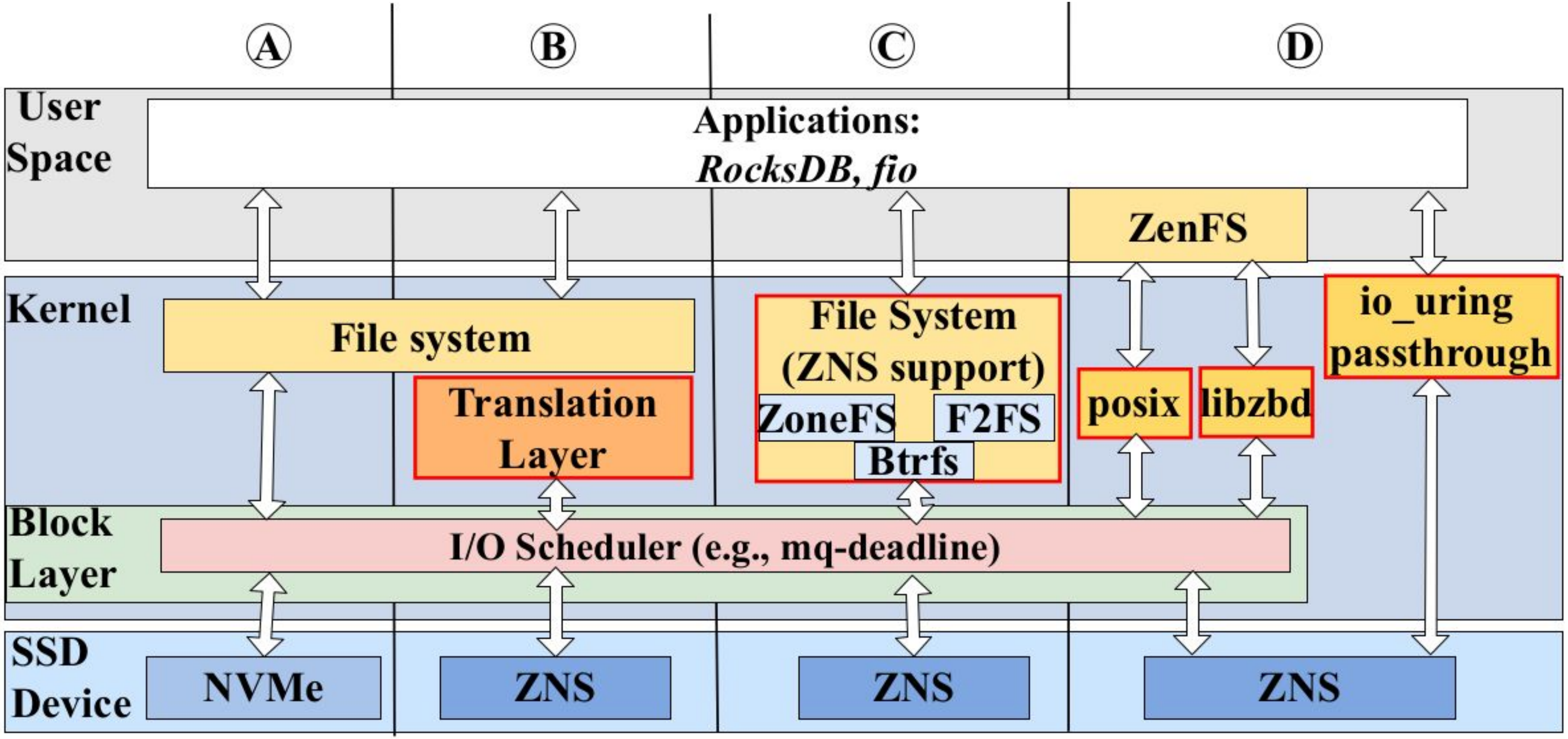
Better control (visibility) on …

1. Data placement in zones
2. Parallelism management
3. Garbage collection
4. Zone resets



With this visibility, we have an opportunity to build a more expressive, complete & comprehensive **Data Lifecycle Event tracing framework!**

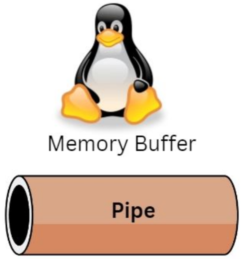
ZNS : Software Integration Challenges (non-Trivial)



zns-tools: An eBPF-powered Profiling Tool for NVMe ZNS SSDs

Do one thing well (the UNIX pipe philosophy)

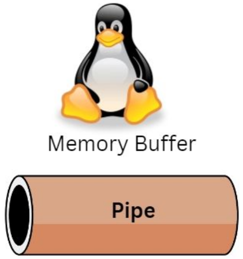
- Collection of tools
- (WiP) `user@system:~$ tool1 | tool2 | tool3`



zns-tools: An eBPF-powered Profiling Tool for NVMe ZNS SSDs

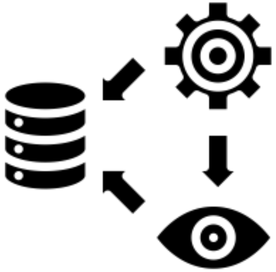
Do one thing well (the UNIX pipe philosophy)

- Collection of tools
- (WiP) `user@system:~$ tool1 | tool2 | tool3`



Keep it modular and standardized

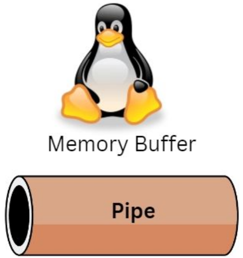
- Follows the Model-View-Controller model
- Decouples trace gathering, processing, and visualization
- BPFtrace output and JSON format (but extensible)



zns-tools: An eBPF-powered Profiling Tool for NVMe ZNS SSDs

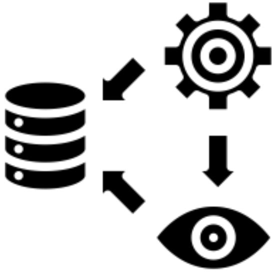
Do one thing well (the UNIX pipe philosophy)

- Collection of tools
- (WiP) `user@system:~$ tool1 | tool2 | tool3`



Keep it modular and standardized

- Follows the Model-View-Controller model
- Decouples trace gathering, processing, and visualization
- BPFtrace output and JSON format (but extensible)

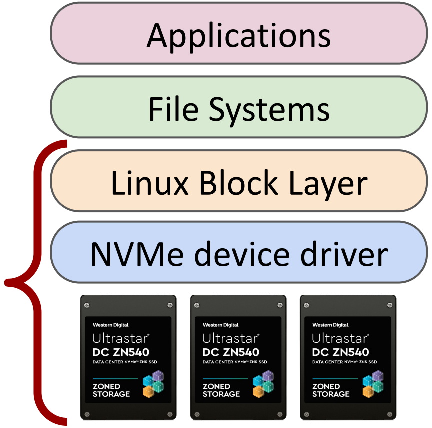


Keep it lightweight

- eBPF !
- Supports complex data structure walks, and parsing

[1/3] zns-tools.nvme: Profiling the ZNS Device

NVMe driver (ZNS) and the Linux block layer profiler



What to trace?

- All I/O commands: Read, Writes, Appends
- All management commands: Reset, Finish
- Size of the payload, timestamp …

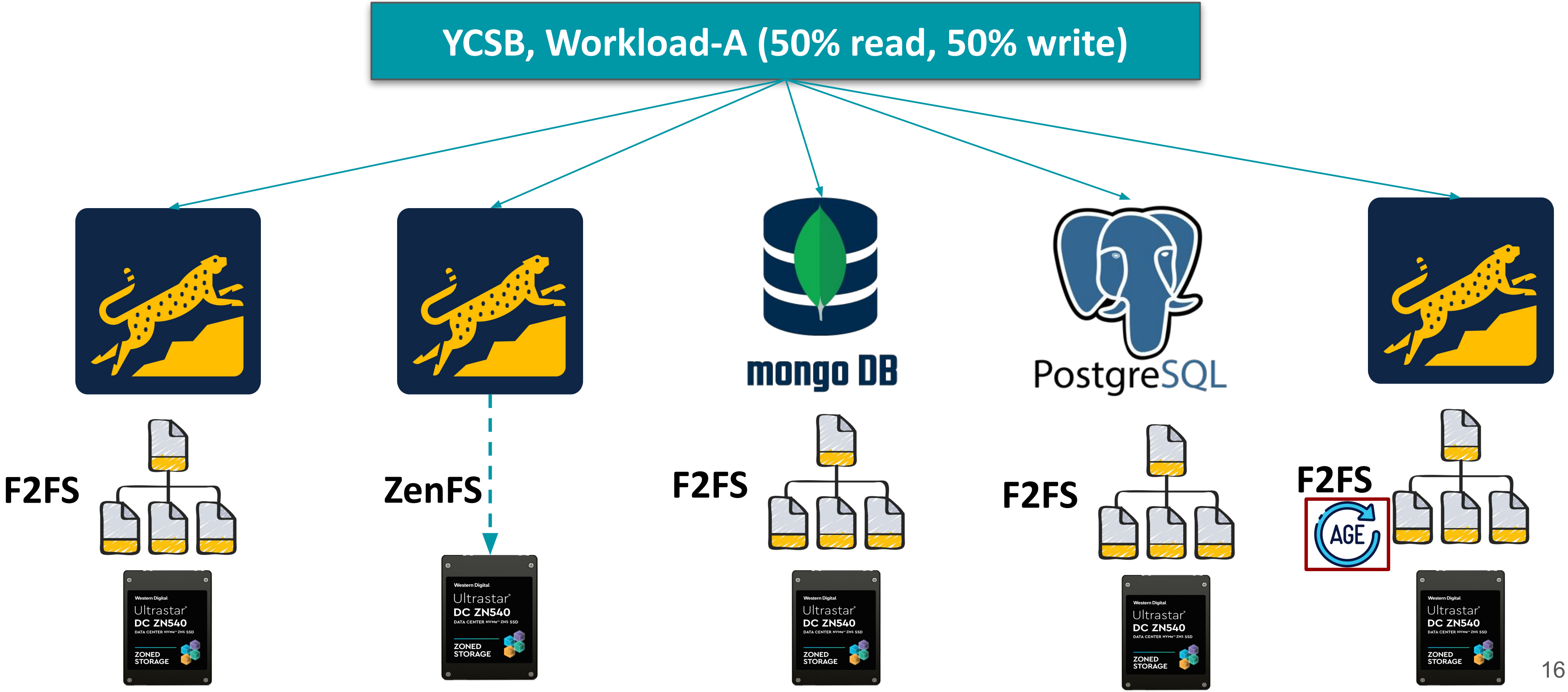
Data is then grouped on per-zone basis

```
user@system:~$ zns-tools.nvme nvme2n1
```

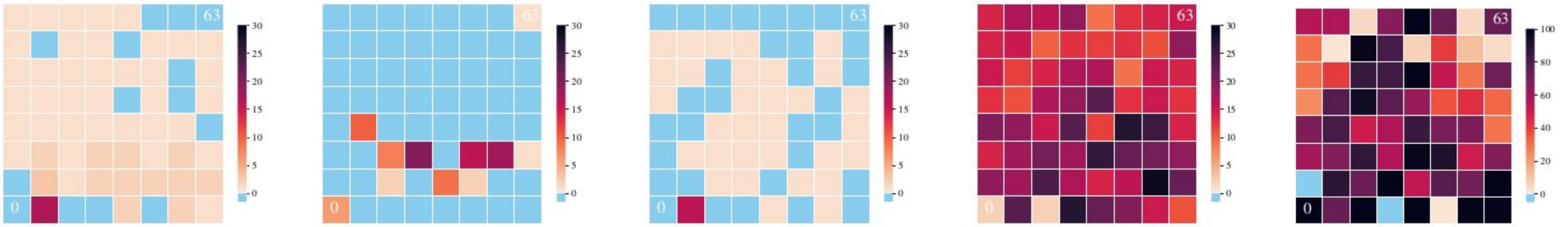
Insights:

- *Are all zones uniformly used (wear-leveling)?*
- *Are there heavily over written or read zones?*

Exploring the ZNS Software Integration Options

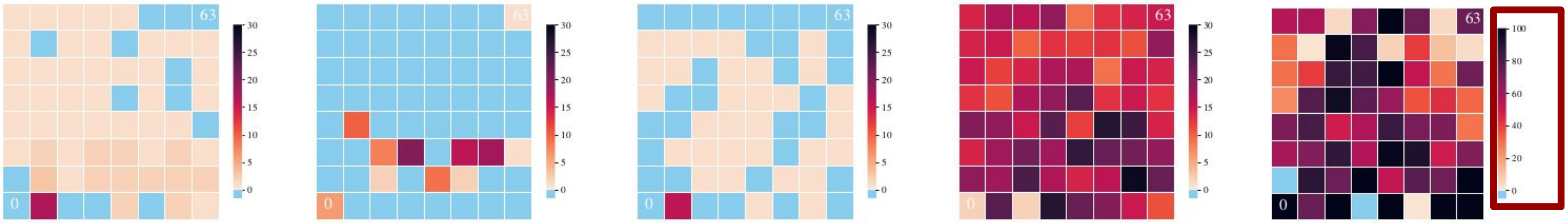


Insight: Not all the ZNS Integration options are the same!



(a) RocksDB + F2FS (b) RocksDB + ZenFS (c) MongoDB + F2FS (d) PostgreSQL + F2FS (e) RocksDB + aged F2FS

Insight: Not all the ZNS Integration options are the same!



(a) RocksDB + F2FS (b) RocksDB + ZenFS (c) MongoDB + F2FS (d) PostgreSQL + F2FS (e) RocksDB + aged F2FS

F2FS has an even zone usage

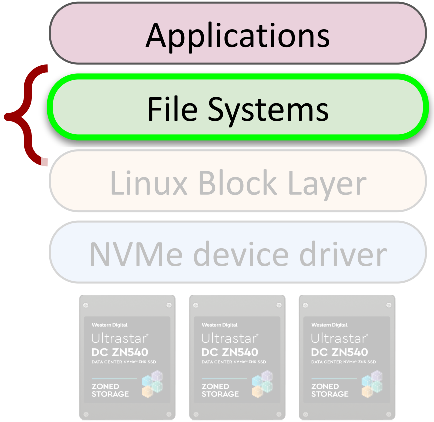
Domain-specific, ZenFS does not any wear-leveling on zones

PostgresSQL issues a lot more zone Resets

Aged F2FS issues significantly more Resets

[2/3]`zns-tools.fs`: Profiling the File System

Linux file system and the VFS-level event tracer



What to trace?

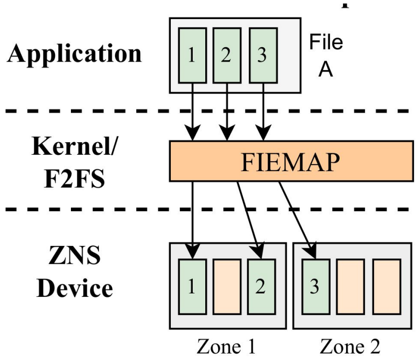
- **File-level** I/O events (open, read, write, close)
- **File system-level** life cycle events
  (garbage collections for LFS, file hotness levels)

Three subtools: `fs.fiemap, fs.imap, and fs.segmap [github]`

Insights:

- *In which zones my files are stored, what is the fragmentation level?*
- *Is my file stored in hot or cold ZNS zone?*
- *How many hot files are in this directory?*

`zns-tools.fs`: Two Specific Responsibilities



[Any FS] Locate a file (name) to its zone storage location

- Uses FIEMAP (ioctl) call to extract file extents (LBA address, length)
- Generate: address-to-zone mappings, extent distribution (min, max, percentiles), hole statistics, zone-level placement information

[F2FS-specific, semantic] Identify hotness classifications of F2FS segments and ZNS zones

- **Data:** file data and file metadata (inode), **Temperate classes:** Hot, Warm, Cold
- Read F2FS metadata (/proc, F2FS checkpoints for NAT)
- Any file/directory → set {F2FS segments, zones, temperature}
- Any zone → set {F2FS file segments, inode segments, file names, offsets}

[3/3]`nvme.app`: Profiling the Whole System

Profile any user defined events functions with {kernel, userspace} eBPF probes

What to trace?

- **Application-level:** (LSM KV) compaction, garbage collection, WAL and memtables flushes
- **File system-level:** All previously discussed file system related events
- **NVMe-level:** All previously nvme, block-level events

Insights:

- *Why does my application data got mixed with other application data?*
- *Why there was a reset issues to this particular zone?*
- *Why did my P99 latencies increased from the baseline?*

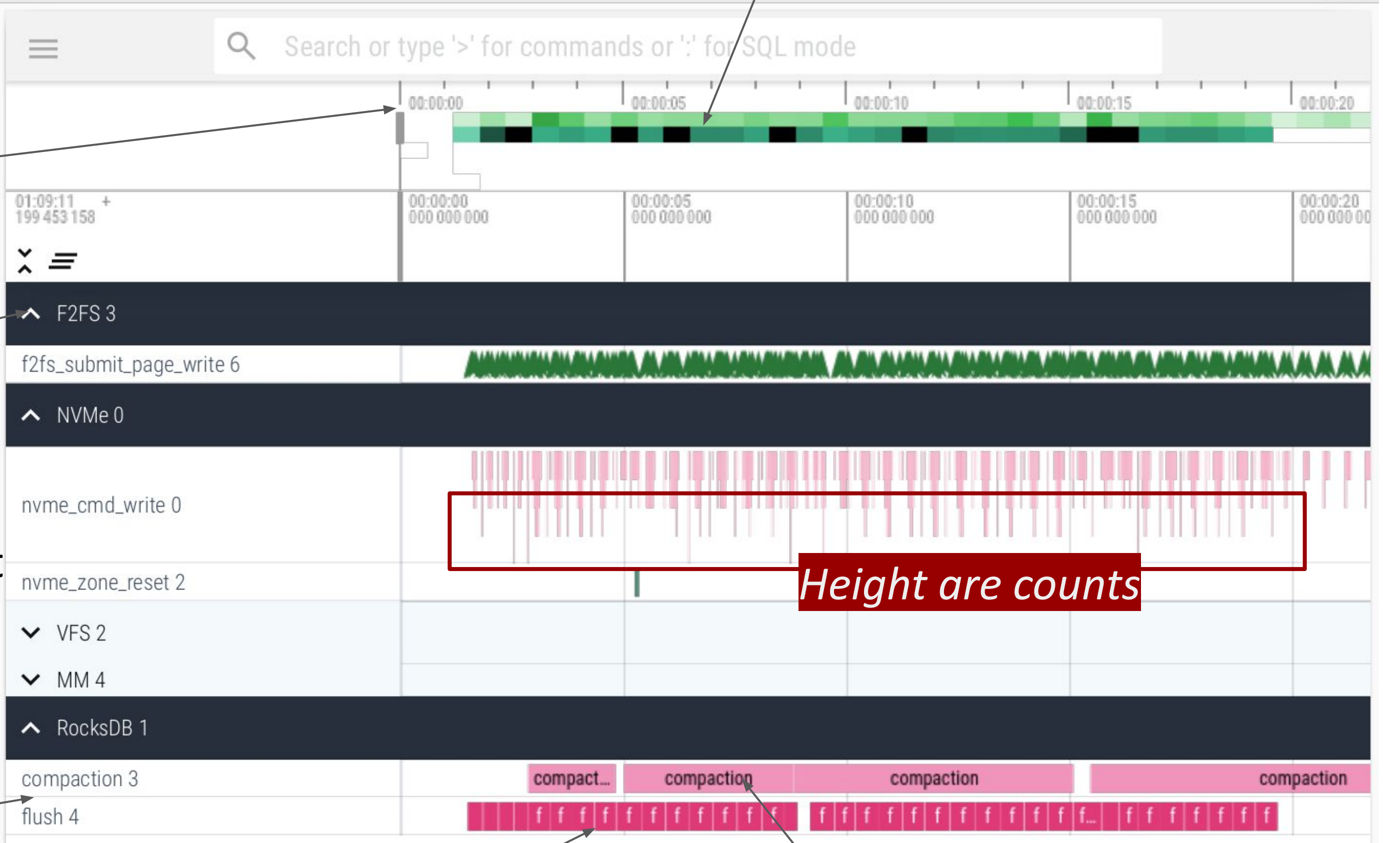
Example Run

Total event count

Timeline stamps

Which layer

Expanded version with tracing of write and reset



Application-level events

"flush" probes

"compaction" probes

Future Work

- **Implement the UNIX pipe design**
- Deep integration with F2FS and Btrfs
- Common abstraction to trace requests across the stack (currently time based)
- Expand to new applications
- NVMe FDP support
- Performance evaluation to high-capacity ZNS arrays (scaling challenge)
- Interactive visualization
- Storage traces in databases for more expressive analysis

Thank you

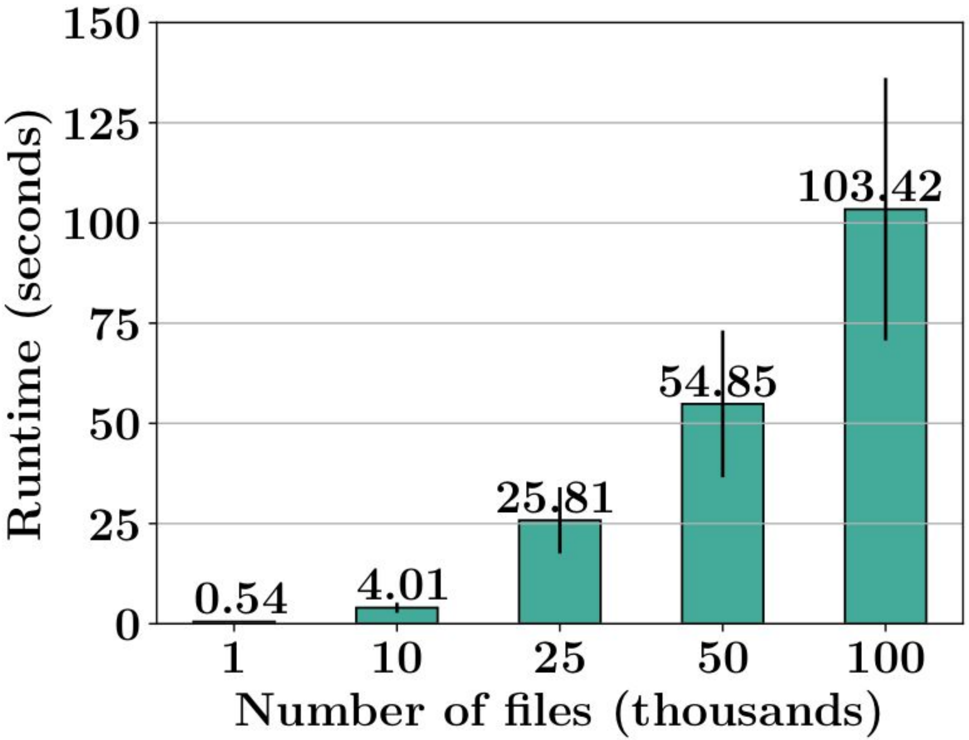
Nick Tehrany   Krijn Doekemeijer

https://github.com/stonet-research/zns-tools

https://atlarge-research.com/pdfs/2024-zns-tools.pdf

Acknowledgments: The Dutch Research Council (NWO) and Western Digital

Overheads



Runtime (seconds)
150
125
100
75
50
25
0
0.54
4.01
25.81
54.85
103.42
1
10
25
50
100
Number of files (thousands)