Zoned NVMe™ Namespaces

Dmitry Fomichev

Western Digital Research, System Software Group

August 8th, 2020

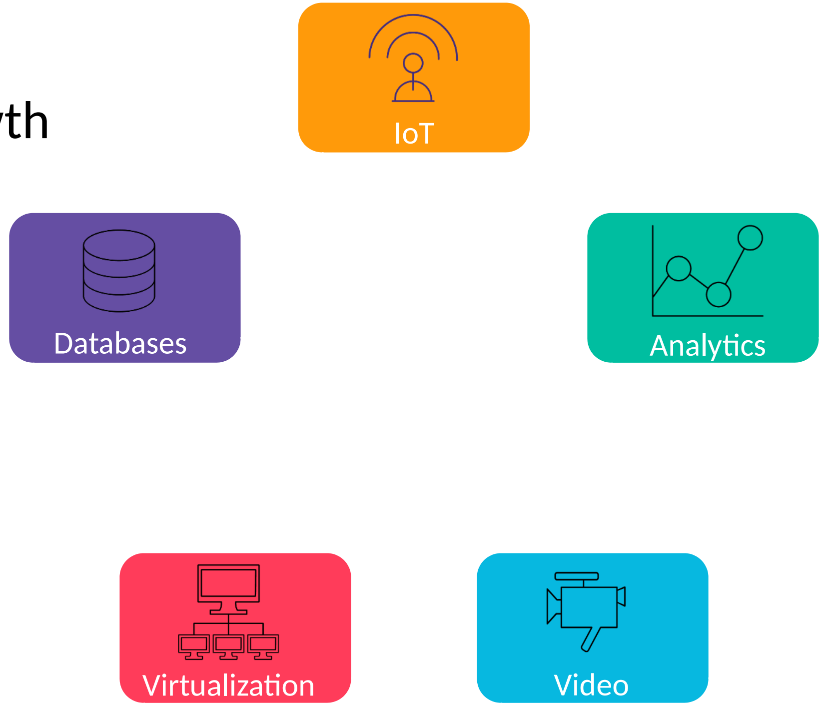
Agenda

1. Data Growth & Conventional SSDs
2. Zoned Namespaces (ZNS) SSDs
3. Zoned Storage Linux Software Ecosystem
4. Demo - Live Emulated ZNS SSD on Linux

Data Growth

Rethinking Storage Architectures for the Zettabyte Age

- IDC expects that 103 zettabytes of data will be generated worldwide by 2023*
  - Proliferation of IoT devices, 5G-enabled technologies, massive growth of video
  - How to store? Manage? Extract Value?
- Scaling Data Centers
  - Dis-aggregation
    - Manage volume, velocity, and variety of the data
  - One SSD solution does not fit all
    - Balancing performance, density, and cost
  - Collaboration and Intelligence
    - Hardware and software collaborate to improve performance and cost



* https://www.idc.com/getdoc.jsp?containerId=US45066919

Solid State Drive

What is the building blocks of an SSD?

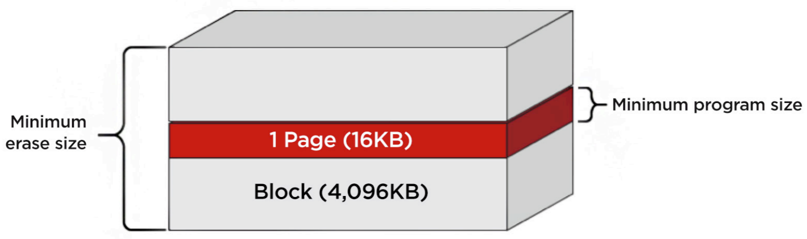
- An SSD bundles 10-100s NAND chips and an SSD controller together
- The SSD controller manages NAND chips characteristics and expose the storage through a storage interface
- A NAND chip is composed of erase blocks, consisting of many pages
  - Within each erase block, you can **only write sequentially**
  - Erase block **must be erased** before new writes
  - **Limited number of erases** of an erase block

Solid State Drives

SSD controller and
10-100x NAND chips

Massive Parallelism

NAND chip layout



Performs a single operation (Read/Program/Erase) at a time

Deploying Conventional SSDs

Why is scaling SSDs difficult?

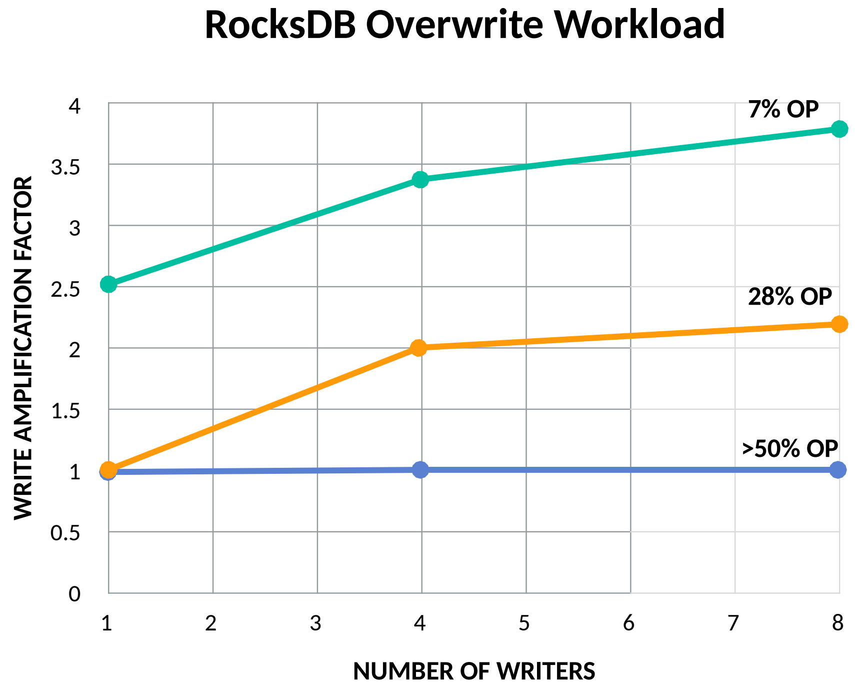
- Great performance needs to be affordable to make the modern scale possible.
- The main cost of SSDs is its media, making up the bulk of the cost.
- Every SSD has more flash inside than users can see (over-provisioning)
- NAND cells can't be scaled any smaller (post Moore's law territory).
- Capacity increases are achieved by stacking layers of NAND and increasing the number of bits per NAND cell.
- SSDs require a complex, highly parallel controller that can contribute to the overall cost of the drive.

Western Digital.

Deploying Conventional SSDs

Performance effects of over-provisioning

- Non-sequential workloads require SSDs to perform garbage collection and CG becomes less impactful with higher over-provisioning (OP)
- Workload has a key impact of the overall SSD performance
  - Multi-tenant environments increase the Write Amplification Factor (WAF) of an SSD
    - Can be separate users, or simply an application with multiple workers (as in the RocksDB example to the right)
  - High WAF
    - Accelerates wear of storage media
    - Reduces performance and impacts QoS
- Media OP improves WAF
  - OP is Typically 7-28% of the media on an SSD.
- **Can OP be minimized, while also improving performance, improve cost and enable new media types?**
  - **i.e., can we have our cake and eat it, too?**



80% Read/20% Random Write Workload

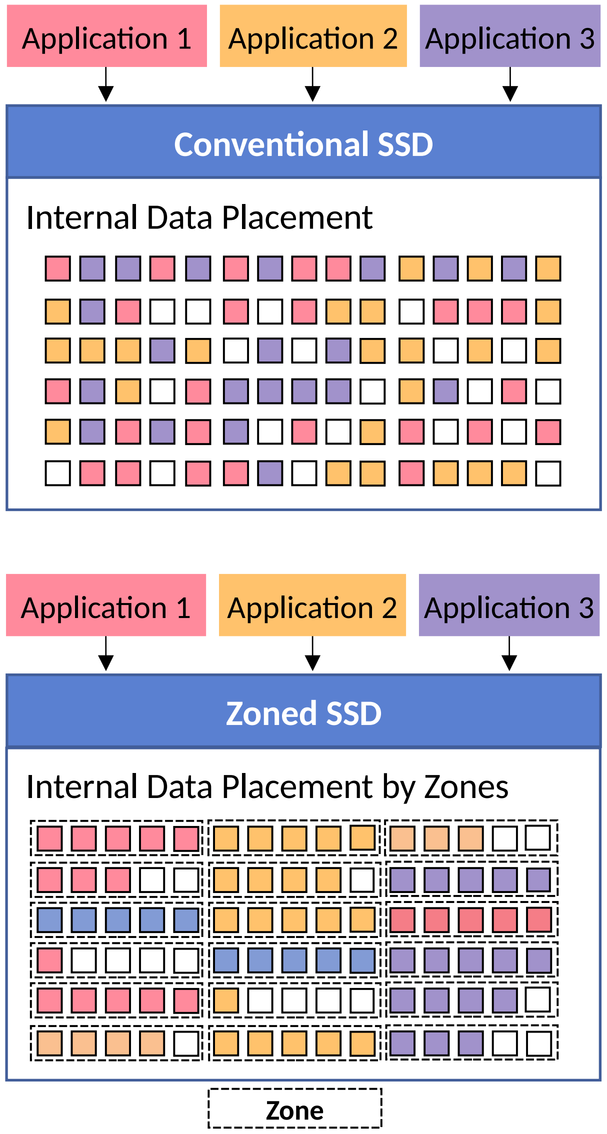
80/20% Read/Write	4K Read Latency			Cost
	Avg (us)	99% (us)	99.9% (us)	
1x WAF (Baseline)	103	338	545	$$$ ($>50%)
2x WAF	135 (+31%)	445 (+32%)	676 (+24%)	$$ (>20%)
4x WAF	192 (+86%)	490 (+45%)	750 (+37%)	$ (>5%)

Western Digital.

Have your Cake and Eat it, Too!

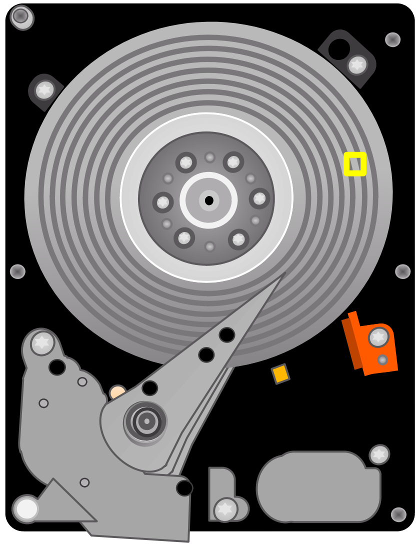
Introducing Host-managed Zoned Storage Interface

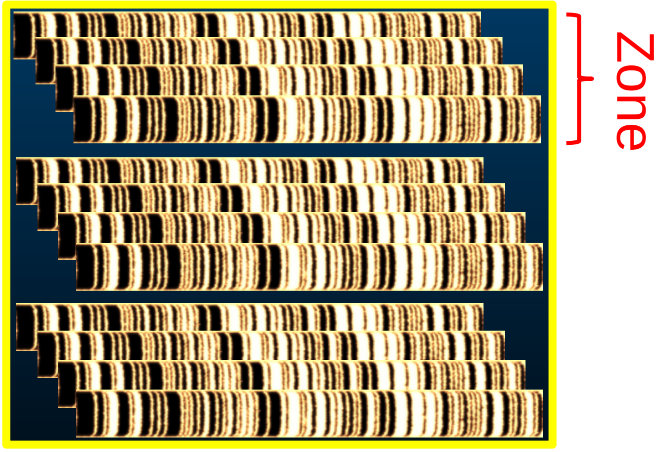
- Host can help by providing a flash-friendly workload
- Use a zoned storage interface to allow host and SSD to collaborate on data placement, such that the host naturally write sequentially within a set of erase blocks
  - **Eliminate** the internal SSD **media maintenance** caused by lifetime mismatch when writing data
    - No longer need to reserve over-provisioned media
    - **Can Prolong life of the media by 2-5x** due to WAF equal to ~1x
    - **~7-28% more storage capacity**
  - Also enables QLC media (4 bits per cell) to be deployed in use-cases previously fulfilled by TLC media (3 bits per cell)
    - QLC has less endurance and performance than TLC, but ZNS makes up for it
    - **Can add additional 33% storage capacity at the same cost**

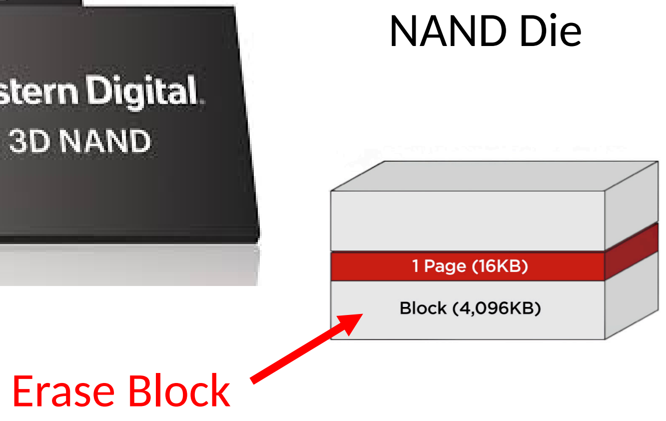


Zoned Storage?

SMR HDD and SSD Have Similar Constraints







- SMR HDDs consist of regions (zones) in which the tracks are overlapped, eliminating wasted space between tracks
- ECC encoding is used to read data correctly.
- Within each zone, you can only write sequentially.

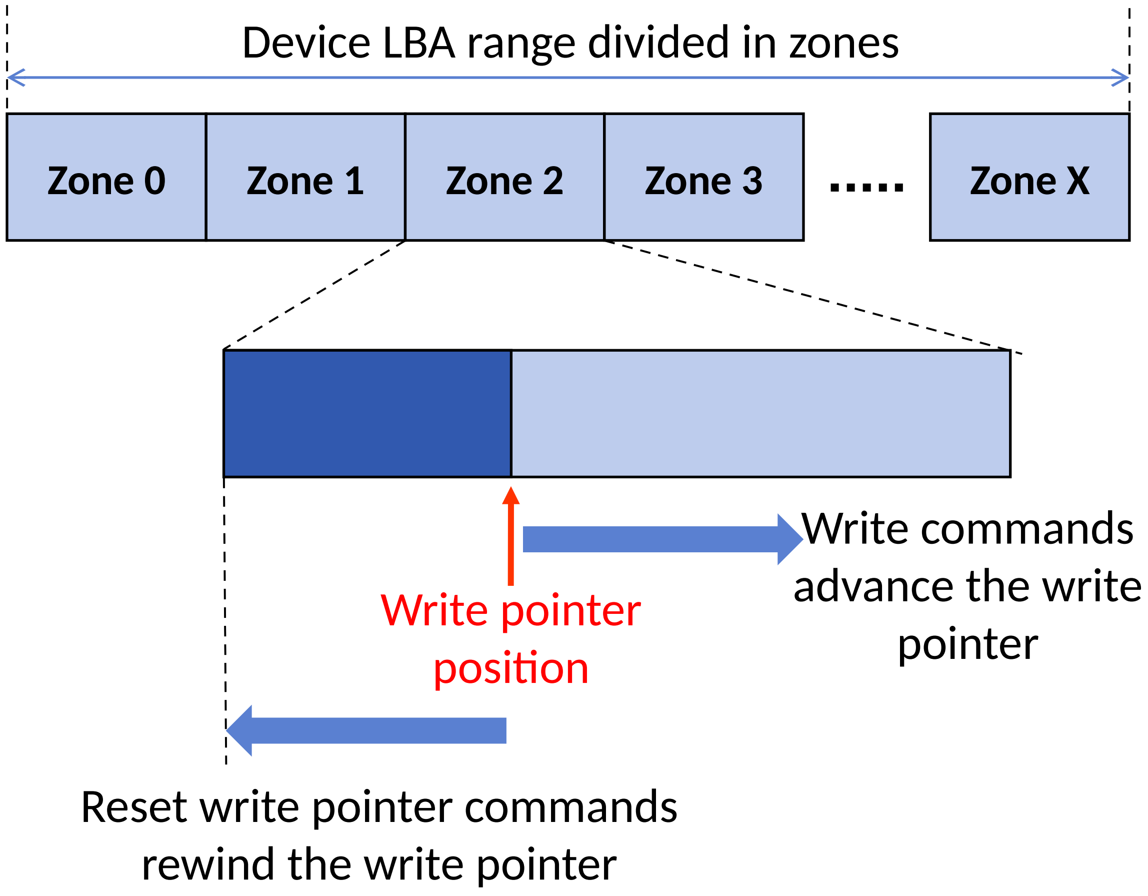
- NAND die are composed of erase blocks, consisting of many pages.
- Within each erase block, you can only write sequentially.
- One or multiple erase blocks can be considered a zone.

Raw SMR HDD and NAND Media Both Require Sequential Write Within Zones

Zoned Storage?

Zoned storage concept

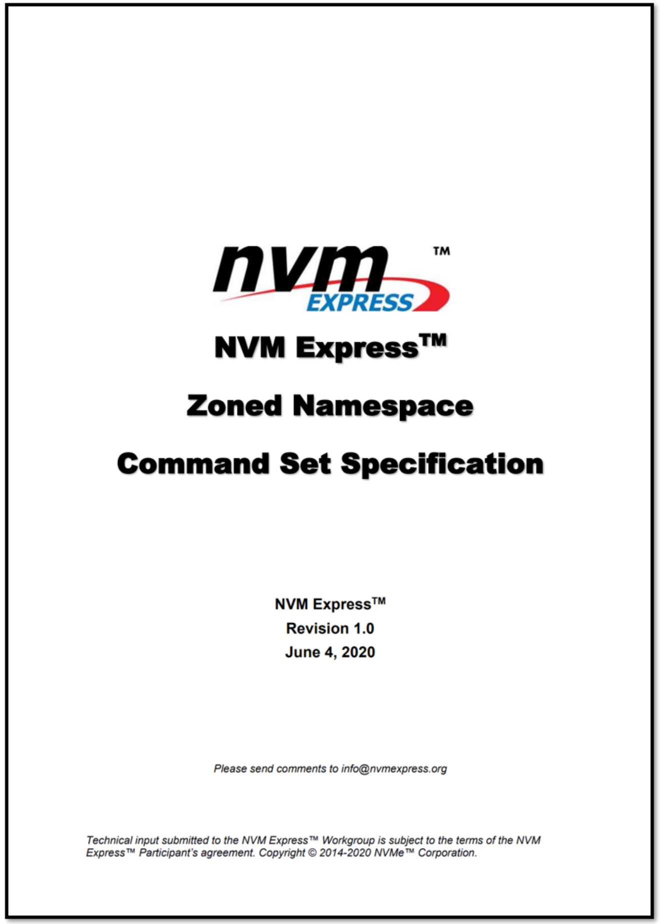
- The storage device logical block addresses are divided into ranges of zones.
- Writes within a zone must be sequential.
- The zone must be erased before it can be rewritten.
- Writing position can be reset to the beginning of the zone. This erases any previously written data.
- Standardized for SCSI and ATA.
- Supported in Linux kernel, extensive support in userspace is available.



NVMe™ Zoned Namespace Command Set

Specification Released in June 2020

- Introduces the Zoned Storage Model for NVMe
- Introduces a new namespace type (Zoned Namespaces)
  - Exposes a set of zones of fixed size to be written sequentially and reset for new writes (matches the NAND media characteristics)
  - Implements the Zoned Namespaces Command Set
  - The command set inherits the NVM Command Set
    - i.e., Read/Write/Flush commands are available.
- Optimized for SSDs
  - Writable capacity of a zone
  - SSD events and hints to further improve performance
  - Out-of-order writes with the Zone Append command


NVM Express™

Zoned Namespace

Command Set Specification

NVM Express™
Revision 1.0
June 4, 2020

Please send comments to info@nvmexpress.org

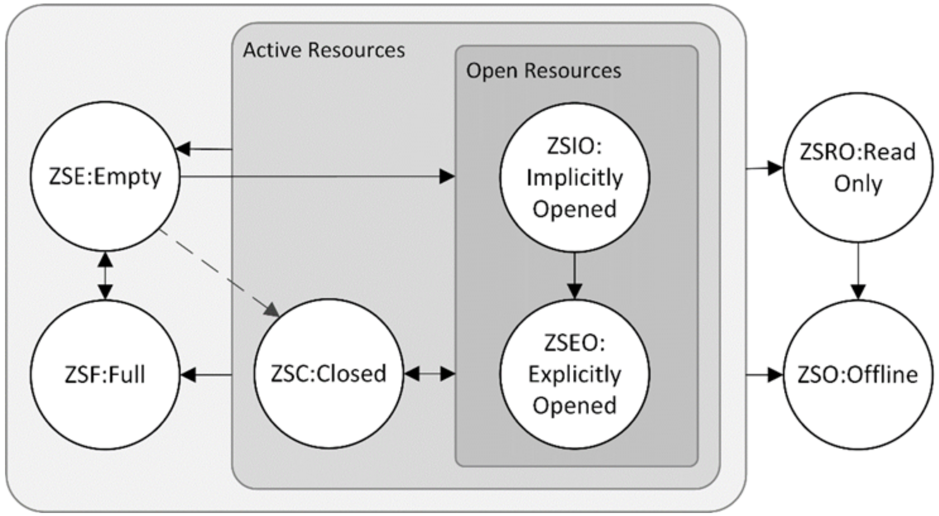
Technical input submitted to the NVM Express™ Workgroup is subject to the terms of the NVM Express™ Participant's agreement. Copyright © 2014-2020 NVMe™ Corporation.

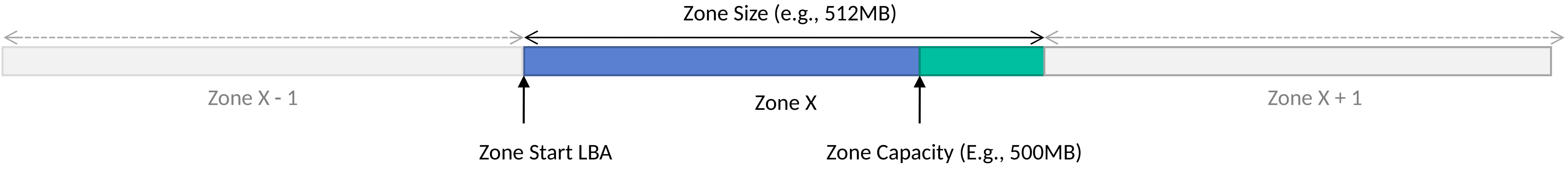
https://nvmexpress.org/developers/nvme-specification/
(Available in the 1.4 TP package)

Zoned Storage Model Overview

Very similar to host-managed ZAC/ZBC

- Zone States
  - Empty, Implicitly Opened, Explicitly Opened, Closed, Full, Read Only, and Offline.
  - Transitions on writes, zone management commands, and device resets.
- Zone Management
  - Open Zone, Close Zone, Finish Zone, and Reset Zone
- Zone Size & Zone Capacity
  - Zone Size is fixed
  - Zone Capacity is the writable area within a zone
- Active and Open Resources associated to a zone
  - Limits the maximum active and open zones in a Zoned Namespace

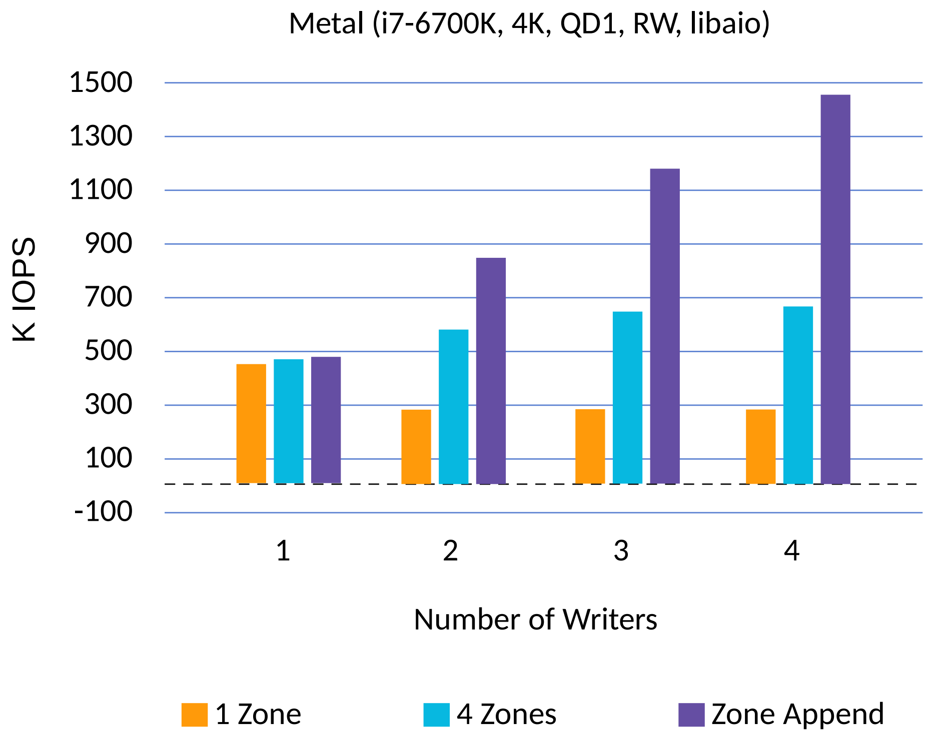




Zone Append

Major sequential write I/O performance boost

- Sequential Writes requires strict write ordering
  - Limits write performance, increases host overhead
- Low scalability with multiple writers to a zone
  - One writer per zone -> Good performance
  - Multiple writers per zone -> Lock contention
- Can improve by writing multiple zones, but performance is limited
- Zone Append to the rescue
  - Anonymous Write Concept
  - Append data to a zone with implicit write pointer
  - Drive returns the LBA where data was written
- No contention. With Zone Append, we scale!

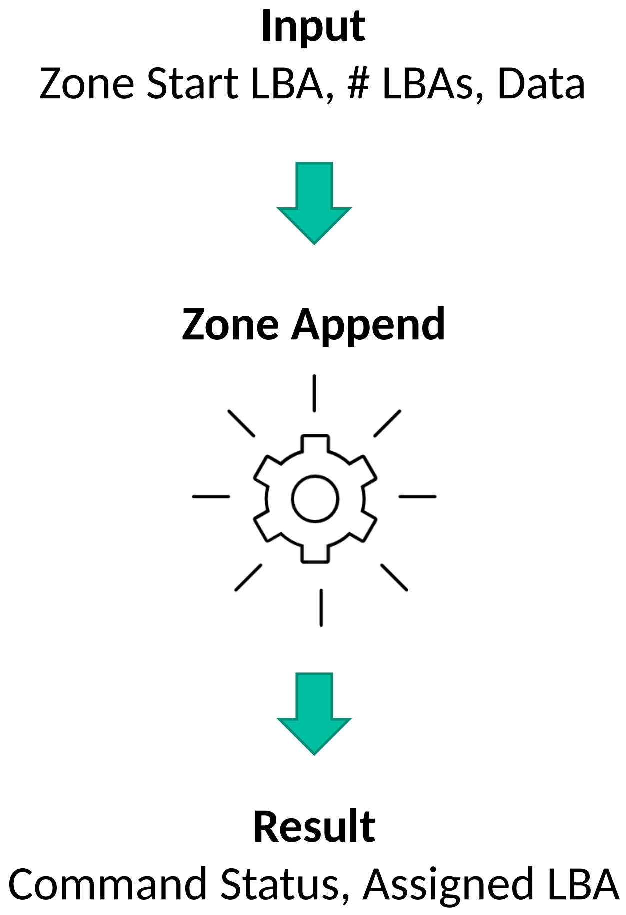


Western Digital.

What is Zone Append?

What makes it powerful?

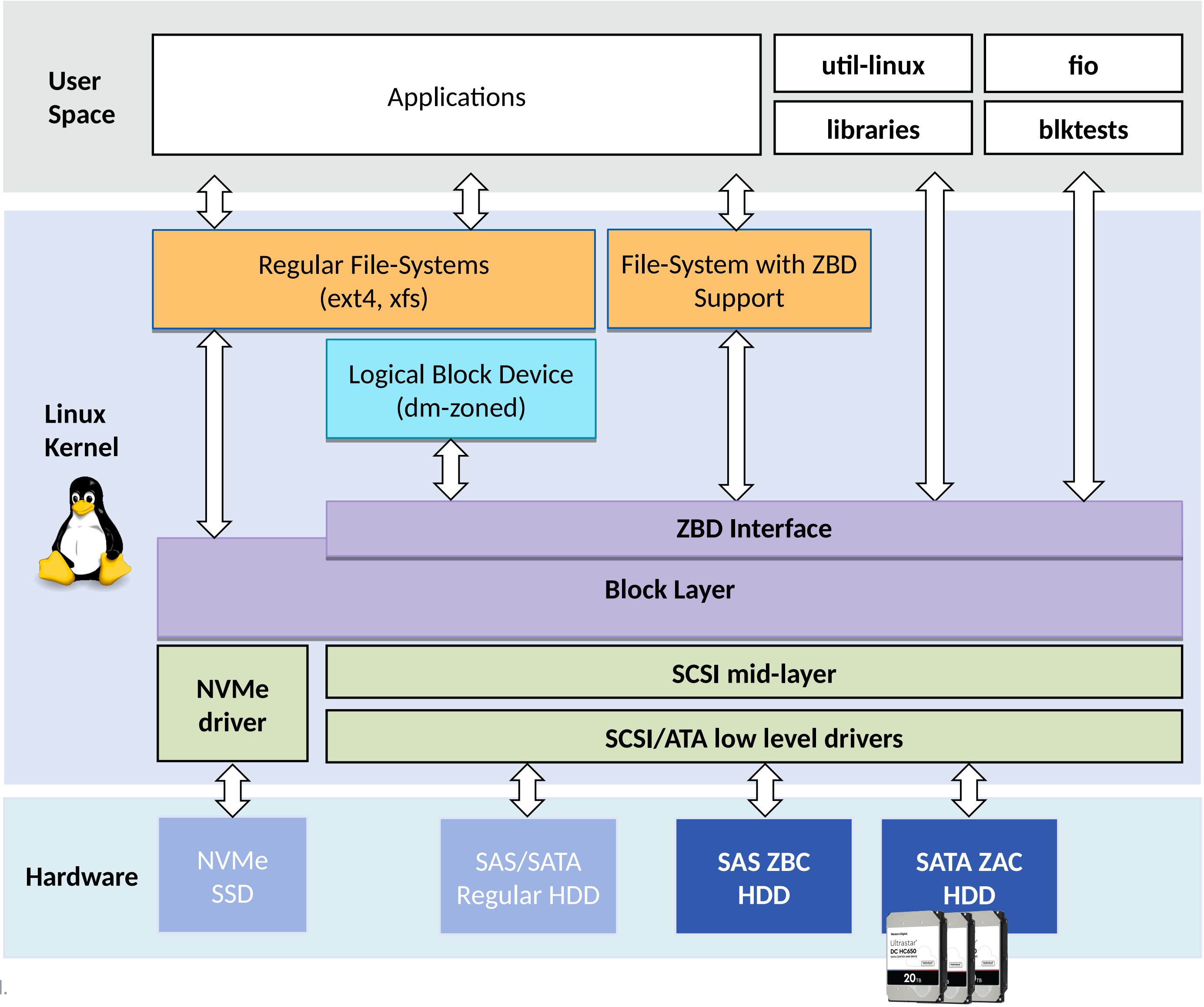
- Zone Append is like a block allocator
  - It chooses which LBAs to use for a write in a zone
- However, block allocators are hard!
  - You're tracking free space…
  - i.e., tracking it, avoiding holes, and fragmentations is a significant overhead in modern implementations
- Zone Append does one thing great – and only one thing
  - Appends are tracked per Sequential Write Required Zone
    - i.e., append point is always known – it's simply the write pointer
    - Easy to implement – works great in hardware.
  - New co-design opportunities
    - SSD tracks fine-grained writes to a zone
    - Host tracks free-space (i.e., zones). The host must only maintain a coarse-grained allocation, thereby avoiding the per LBA allocation overhead.



Linux Zoned Block Device Support

Block device abstraction interface created for SMR disks

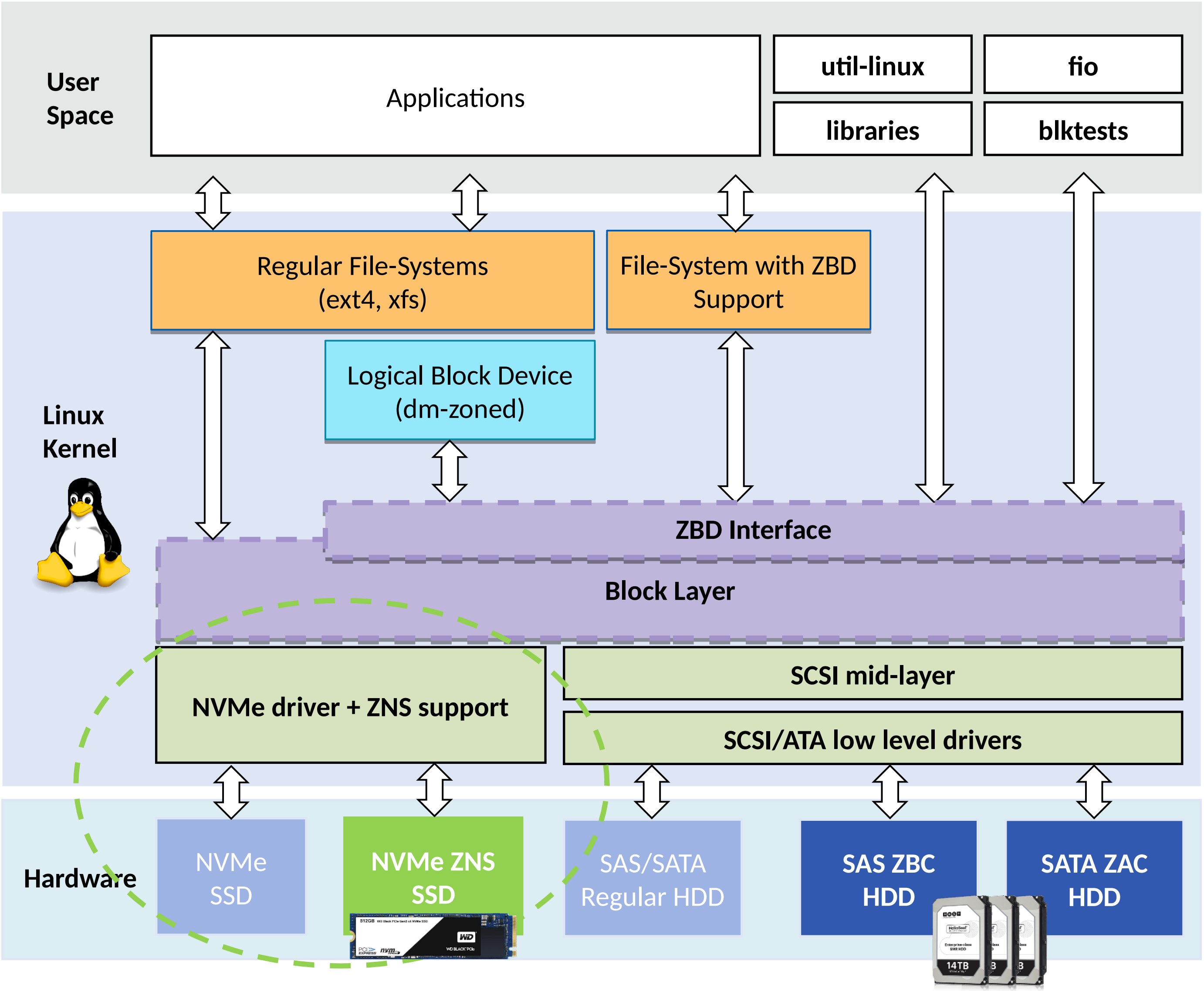
- Development started in 2016
  - First enabled in kernel version 4.10
- Introduces "zoned" disk abstraction API for user applications and in-kernel components
  - Device zone report and management functions
  - Write ordering guarantees to support the device sequential write constraint
- Mature storage stack for zoned block device through enablement of SMR HDDs:
  - Linux eco-system enablement
    - Device drivers, block layer (zoned subsystem), general plumbing
    - Device mappers (dm-zoned, dm-linear, dm-flakey)
    - File-systems with zone enablement: f2fs, btrfs, zonefs
    - Tools enabled: fio, libzbd, blkzone, gzbc, and blktests
- Mature, robust, and adopted by some of the biggest consumers of storage



Linux Kernel

Enabling ZNS Software Eco-system

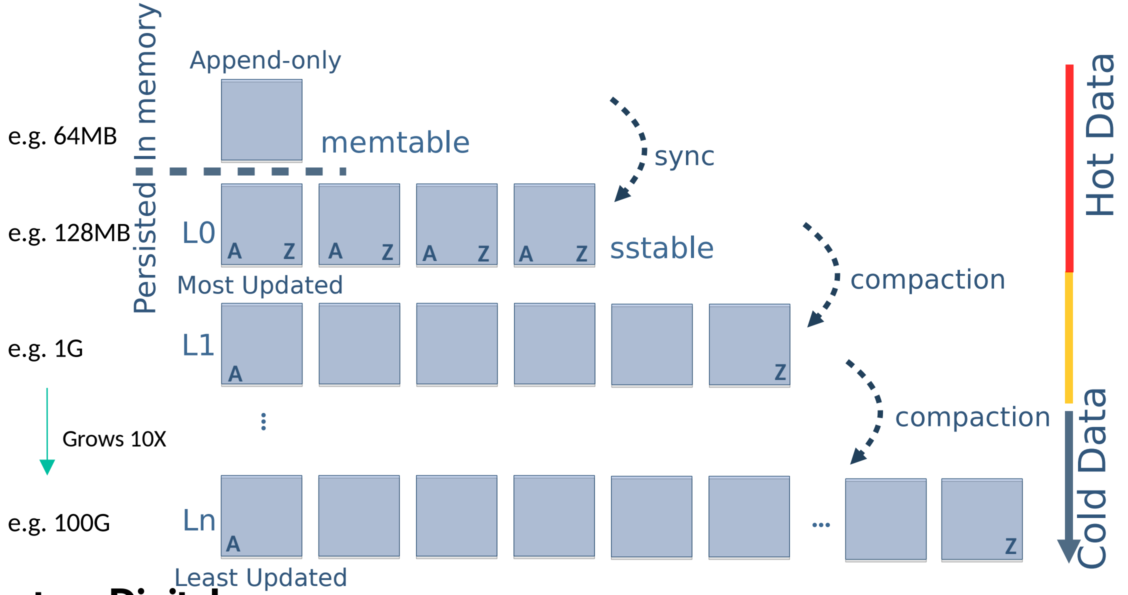
- ZNS integrated into Zoned Block Device(ZBD) interface
  - All in-kernel components and applications already supporting SMR disks can be easily modified for ZNS optimization
  - Main adjustment points are support for zone capacities lower than zone size and limited number of active zones
- Zone append emulation support in SCSI stack further simplifies integration
  - Zone append is now the preferred default write path for zoned block devices
- ZNS support available in the upcoming releases
  - Linux kernel
    - https://lwn.net/Articles/823737/
  - qemu – virtual NVMe backend
    - https://lists.nongnu.org/archive/html/qemu-block/2020-06/msg00720.html
  - nvme-cli
    - https://github.com/linux-nvme/nvme-cli/commit/c1fc890937e7d644f1a4a6f3934af6aae33d018a
  - libzbd
    - https://github.com/westerndigitalcorporation/libzbd

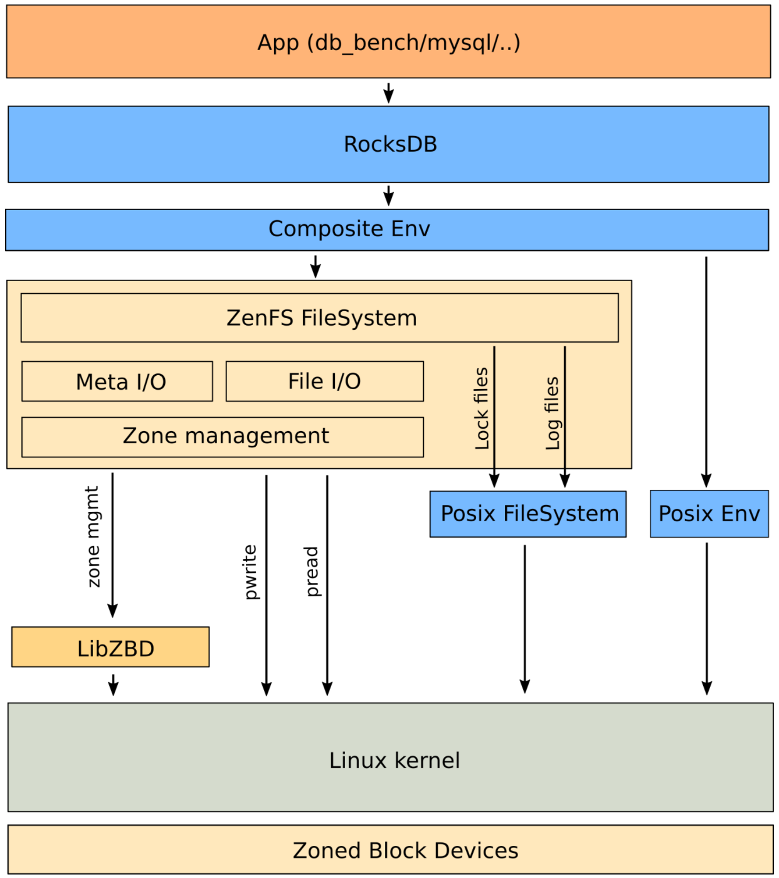


Enabling RocksDB

End-to-End Integration of Zones

- Key-value store where keys and values are arbitrary byte streams.
- ZenFS – a new storage backend for RocksDB
  - Maps zones to sstables
  - ~1X device write amplification (3-6X WAs measured)
- Zone support in progress
  - https://github.com/facebook/rocksdb/pull/6961





Demo Time!

Ongoing Work

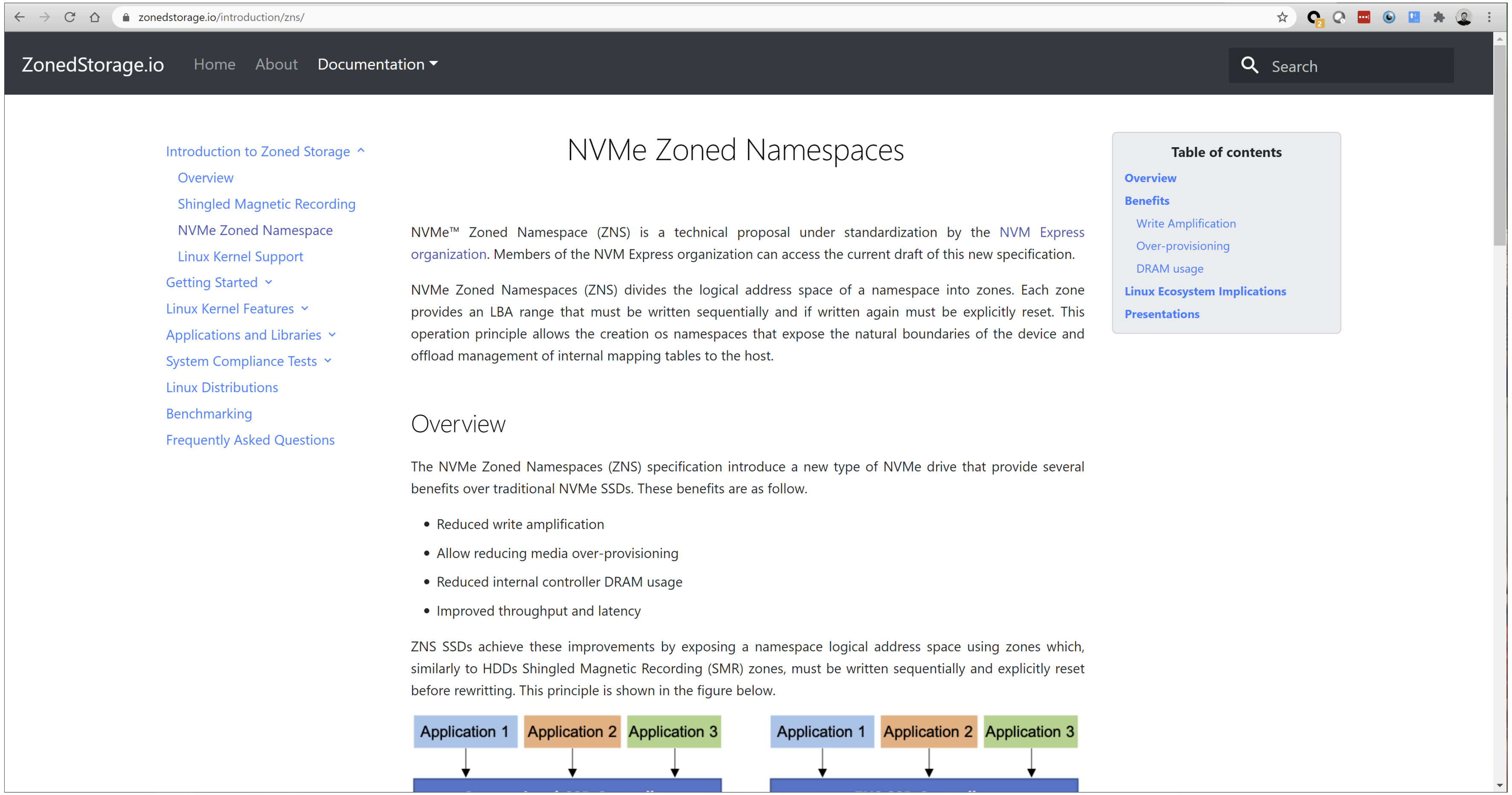
Road Ahead for ZNS

- Extend Zone Append support
  - To add to kernel block subsystem, io_uring, zoneFS and fio
- Move Btrfs to use Zone Append for data writeout
  - Encouraging results in terms of performance
- Add Zone Capacity support in fio
- QEMU integration

ZonedStorage.io

Community Site

See [Zonedstorage.io](https://zonedstorage.io) for technical documentation on zoned storage software, kernel interface, etc.



Western Digital®

