Western Digital®

Zone Append: A New Way of Writing to Zoned Storage

Matias Bjørling

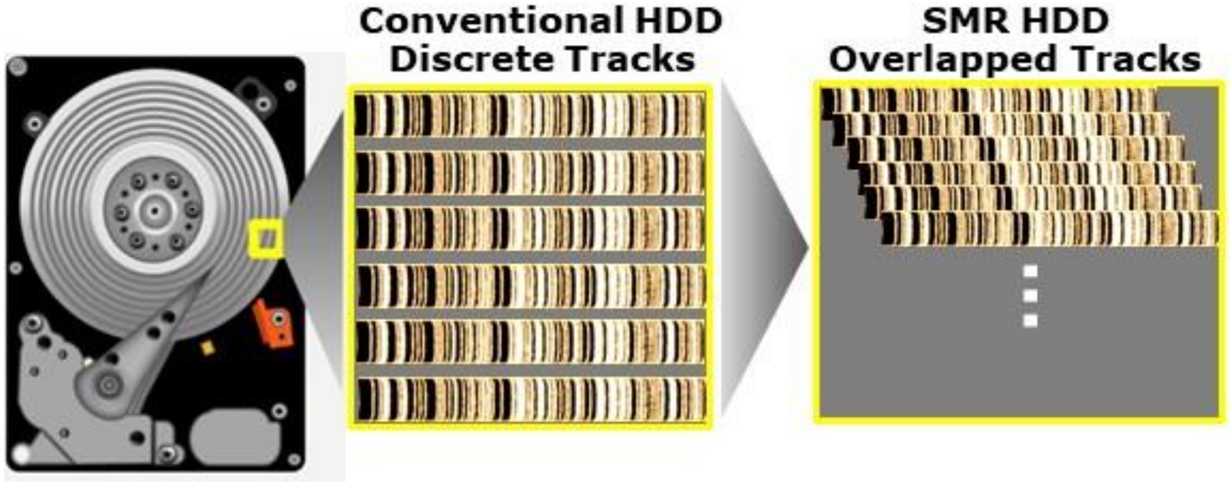
Director, Emerging System Architectures

USENIX Vault - 23rd of February 2020

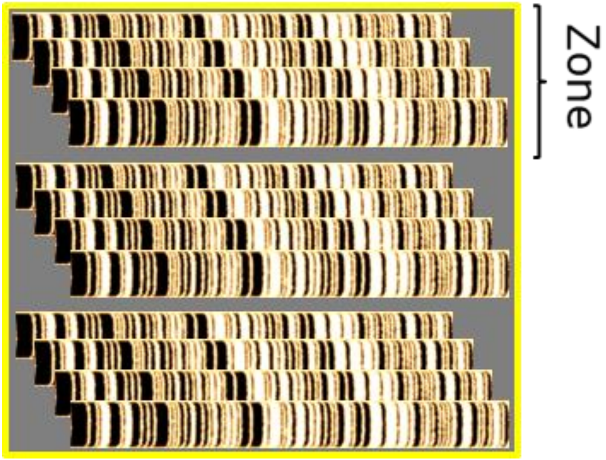
Zoned Block Storage already in HDDs

Take advantage of SMR capacity growth

- SMR (Shingled Magnetic Recording)
  - Enables areal density growth
  - Causes magnetic media to act like flash
    - Data must be erased to be re-written



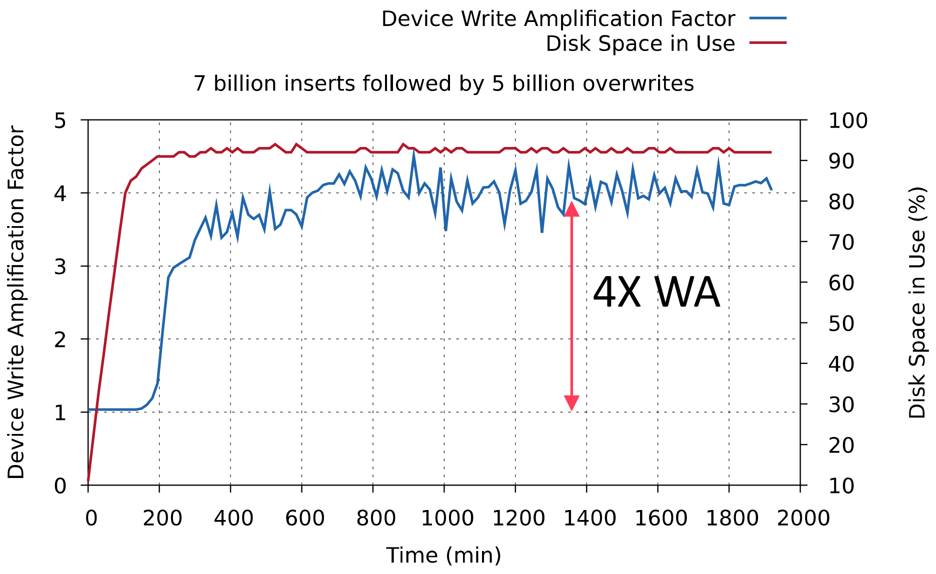
- Zoned Block access for HDDs
  - HDD formatted into fixed sized regions
  - Host/Device enforce sequential writes in LBA space to mitigate RMW effects of SMR
  - Standardized ZAC and ZBC

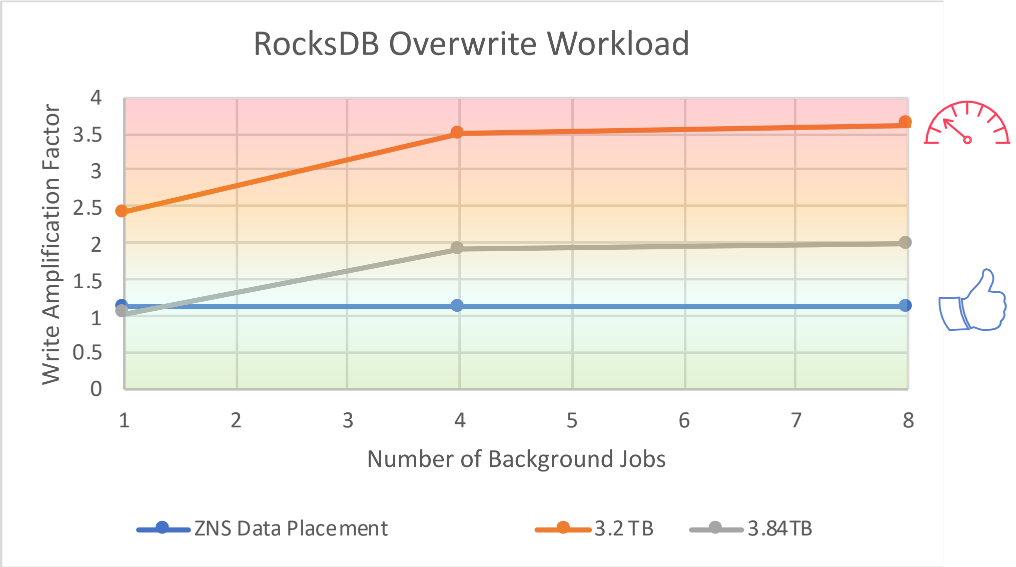


Zones for Solid State Drives

Motivation

- Zones for a typical SSD design provides
  - 20% more storage capacity
    - Media for over-provisioned can be exposed
  - Reduction of write amplification (~1X)
    - The device no longer requires traditional device-side GC
  - Latency improvement as well
    - Lower write amplification equals better QoS!





Zoned Namespaces Overview

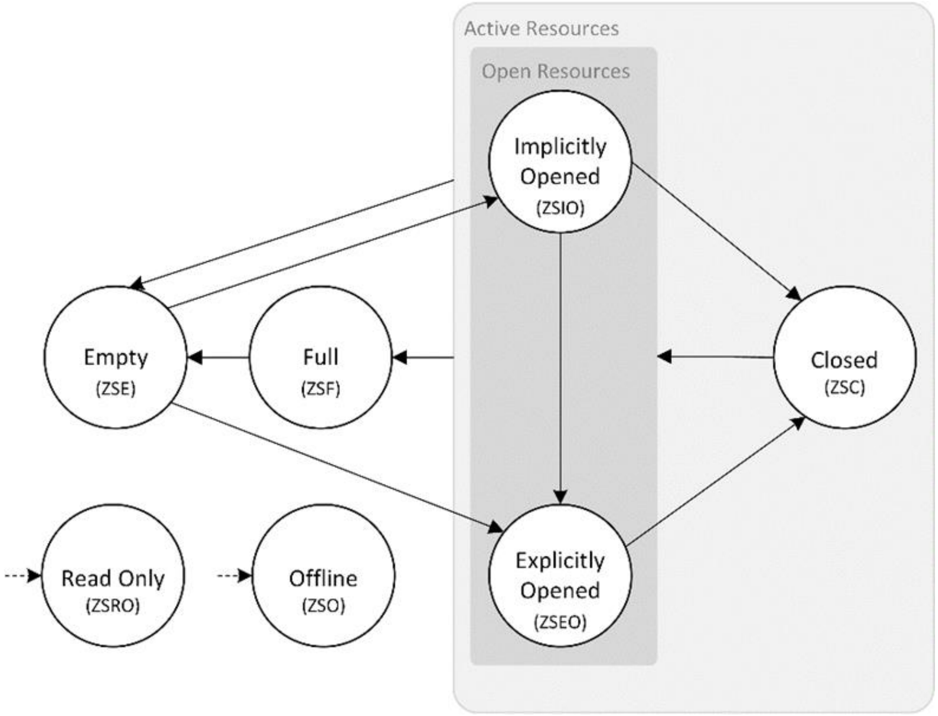
Standardization in the NVMe$^{\text{TM}}$ working group

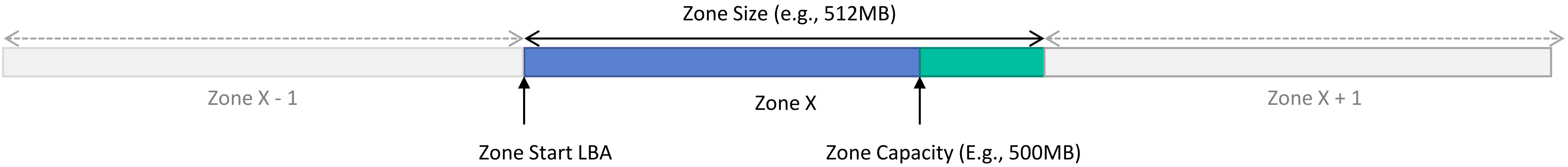
- Inherits the NVM Command Set
  - i.e., Read/Write/Flush commands are available
- Namespace divided into fixed sized Zones
  - Sequential Write Required is only zone type supported for now
- Aligned to host-managed ZAC/ZBC model, with some SSD optimizations
  - Zone Capacity (Fixed Zone Sizes)
  - Zone Descriptors
  - Zone Append
- Soon to be published

Zoned Namespaces Specification

Host-Managed Zoned Block Devices

- 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(NEW)
  - Zone Size is fixed
  - Zone Capacity is the writeable area within a zone

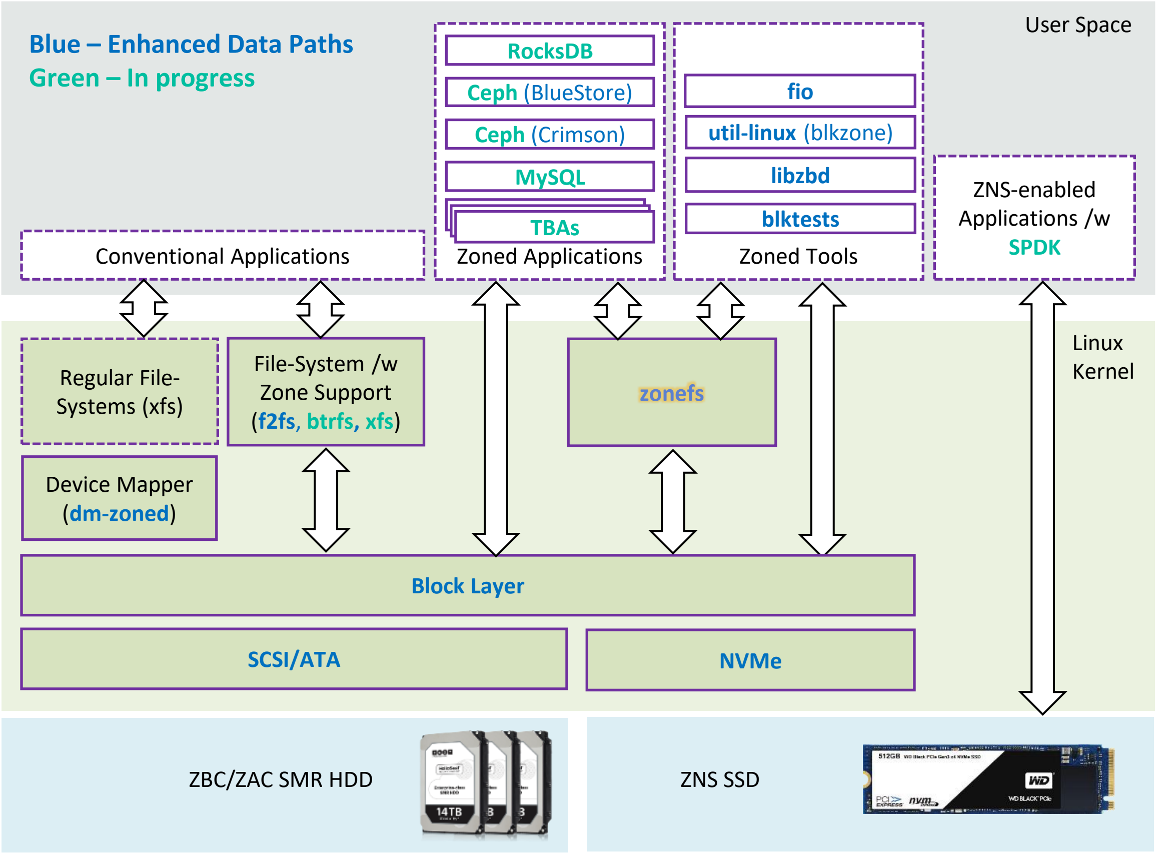




Linux Zones Software Eco-system

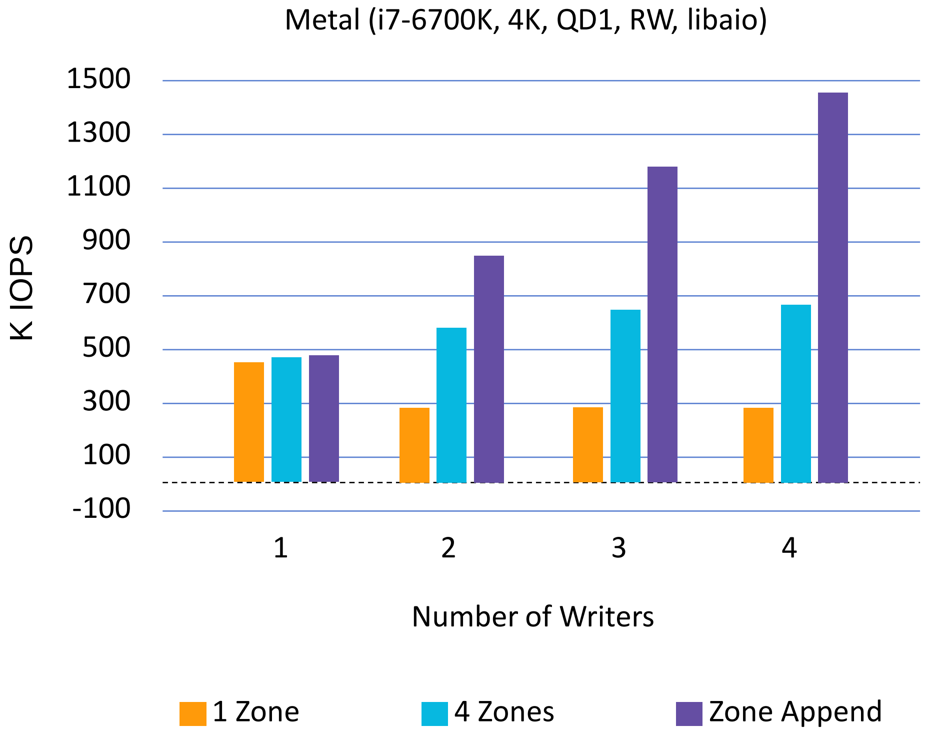
Builds upon the existing zoned (SMR HDDs) software support

- Mature storage stack for zoned block device through enablement of SMR HDDs:
  - Linux kernel 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 many of the largest consumers of storage
- Latest News
  - ZoneFS – New ~~kid~~file-system on the block!
  - Btrfs – Zone support in progress
- Upcoming
  - Base Zoned Namespaces Support
    - Zone Capacity + NVMe device driver support
  - Zone Append command
  - XFS, RocksDB, Ceph, MySQL, and "TBA's"



What is Zone Append

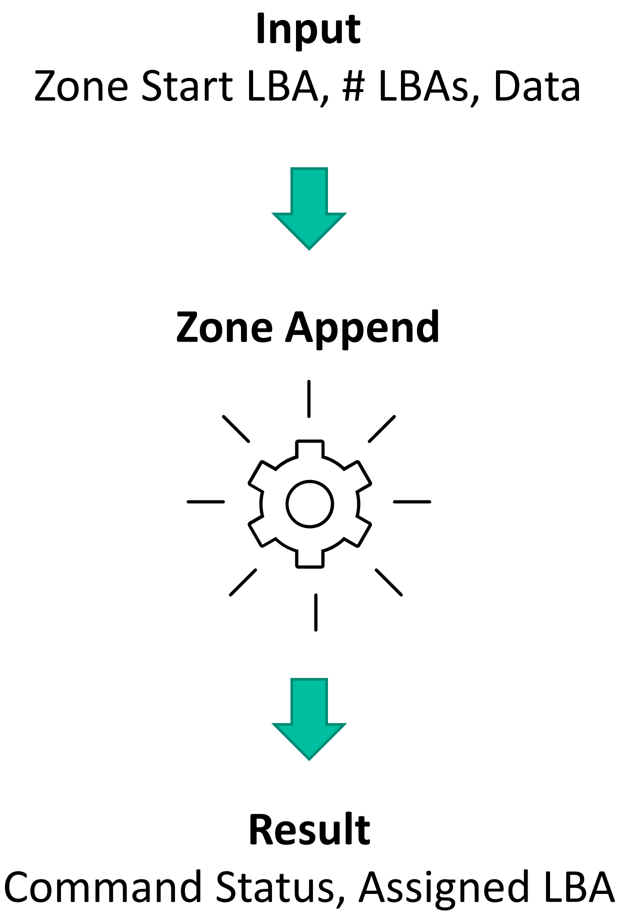
- Sequential Writes equals Strict Write Ordering
  - Limits write performance, increases host overhead
- Low scalability with multiple writers to a zone
  - One writer per zone -> Good performance
  - Multiple writers per zone -> Lock contention
- Can improve by writing multiple Zones, but performance is limited
- Zone Append to the Rescue
  - Append data to a zone with implicit write pointer
  - Drive returns LBA where data was written in zone



What is Zone Append?

What makes it powerful?

- Zone Append is like a block allocator
  - It's 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 that 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.
  - Co-design
    - 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.



Input

Zone Start LBA, # LBAs, Data

Zone Append

Result

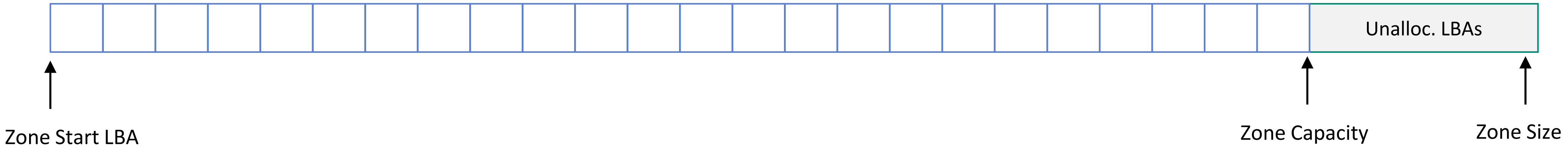
Command Status, Assigned LBA

What is Zone Append?

Example

- Zone Size: 32 LBAs
- Zone Capacity: 24 LBAs

Cmd #	Starting LBA	# LBAs	Assigned LBA	Write Pointer	Write Pointer (After Cmd)
0	LBA 0 (ZSLBA)	1 (4K)	0	0 (Zone to Open)	1
1	LBA 0 (ZSLBA)	2 (8K)	1	1	3
2	LBA 0 (ZSLBA)	5 (20K)	3	3	8
3	LBA 0 (ZSLBA)	8 (32K)	8	8	16
4	LBA 0 (ZSLBA)	1 (4K)	16	16	17
5	LBA 0 (ZSLBA)	5 (20K)	17	17	22
6	LBA 0 (ZSLBA)	2 (8K)	22	22	24 (Zone to Full)



Western Digital.

Zone Append using nvme-cli

Report Zones

```
silverwolf@ZNS-IS-AWESOME:~/git/nvme-cli$ sudo ./nvme zone-report --help
Usage: nvme zone-report <device> [OPTIONS]

Retrieve zones from a specific device in binary or human readable format

Options:
  [  --namespace-id=<NUM>, -n <NUM> ]   --- Desired namespace
  [  --slba=<NUM>, -s <NUM> ]           --- Start LBA of the zone
  [  --nr_zones=<NUM>, -z <NUM> ]       --- Maximum number of zones to be
                                            reported
  ...
```

Zone Append

```
silverwolf@ZNS-IS-AWESOME:~/git/nvme-cli$ sudo ./nvme zone-append --help
Usage: nvme zone-append <device> [OPTIONS]

The zone append command is used to write to a zone using the slba of the
zone, and the write will be appended from the write pointer of the zone

Options:
  [  --slba=<NUM>, -s <NUM> ]           --- starting lba of the zone
  [  --data=<STR>, -d <STR> ]           --- File containing data to write
  [  --metadata=<STR>, -M <STR> ]       --- File with metadata to be written
  [  --limited-retry, -l ]              --- limit media access attempts
  ...
```

Zone 0 State

```
nvme zone-report -s0 -z 1 /dev/nvme0n1
Zones reported: 1
SLBA: 0x0 WP: 0x0 Cap: 0xNA State: EMPTY Type: SEQWRITE_REQ
```

Append 1 LBA to Zone 0

```
nvme zone-append -s 0 -d ../onelba /dev/nvme0n1
zone-append: Success
```

Zone 0 State

```
nvme zone-report -s0 -z 1 /dev/nvme0n1
Zones reported: 1
SLBA: 0x0 WP: 0x1 Cap: 0xNA State: IMP_OPENED Type: SEQWRITE_REQ
```

Append 8 LBAs to Zone 0

```
nvme zone-append -s 0 -d ../eightlbas /dev/nvme0n1
zone-append: Success
```

Zone 0 State

```
nvme zone-report -s0 -z 1 /dev/nvme0n1
Zones reported: 1
SLBA: 0x0 WP: 0x9 Cap: 0xNA State: IMP_OPENED Type: SEQWRITE_REQ
```

Zone append example usage

Pseudo code for a block allocator

```
u16 zone_append(u64 *lba, char *data, u16 num_lbas);

int write_and_map_block(u64 zone_start, struct block *block) {
        u64 lba = zone_start;
        u16 status;

        status = zone_append(&lba, block->data, block->num_lbas);
        if (status != NVME_STS_OK)
            return -ZONE_APPEND_ERROR;

        /* The data was persisted and written
         * lba has been updated to reflect the start address
         */
        map_chunk(lba, block->id);

        return 0;
}
```

Zone Append APIs

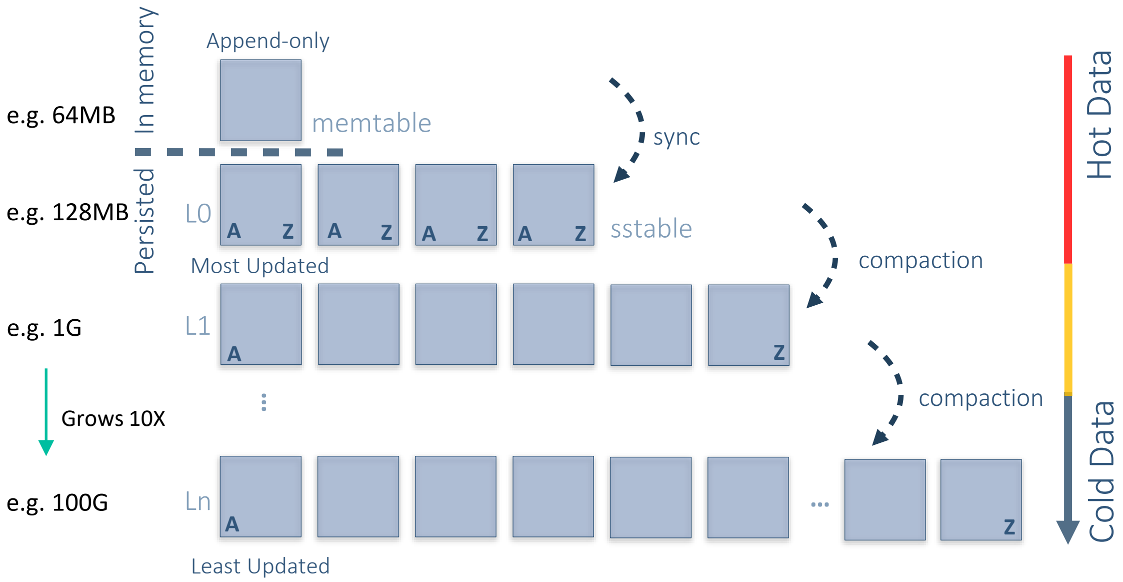
Linux Kernel internal

```
bio = bio_alloc(GFP_KERNEL, (len + PAGE_SIZE - 1) >> PAGE_SHIFT);
bio->bi_opf = REQ_OP_ZONE_APPEND;
bio_set_dev(bio, bdev);
bio->bi_iter.bi_sector = zone;

while (len > 0) {
       u64 count = min_t(u64, len, PAGE_SIZE);

       bio_add_page(bio, page, count, 0);
       len -= count;
}

ret = submit_bio_wait(bio);

if (!ret)
        printk(“Sectpr assigned %ld\n", bio->bi_iter.bi_sector);
```

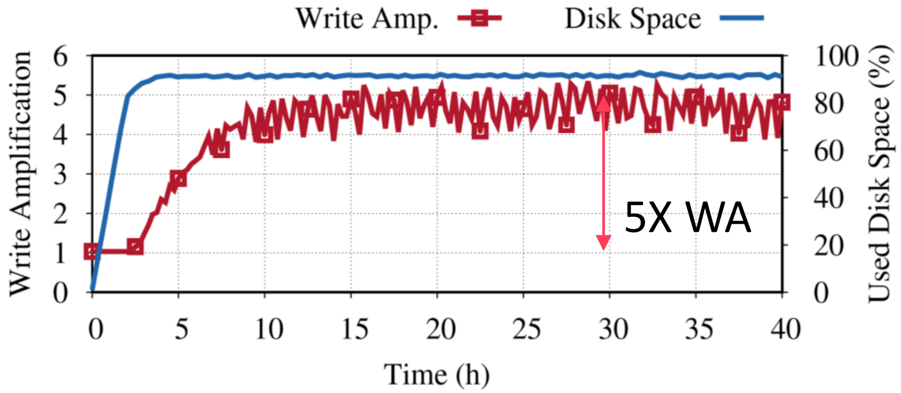
Zone Append Use-Cases: RocksDB + ZNS

- Key-value store where keys and values are arbitrary byte streams.
- Zoned Namespaces Support
  - ZEnv, a new zoned storage environment/back end is being developed to enable ZNS with Zone Append:
    - Provides end-to-end integration with zoned block devices
    - Provides a simplified file system interface and maps RocksDB files to a set of extents on a zoned block device

Based on Log-Structured Merge (LSM) Tree data structure

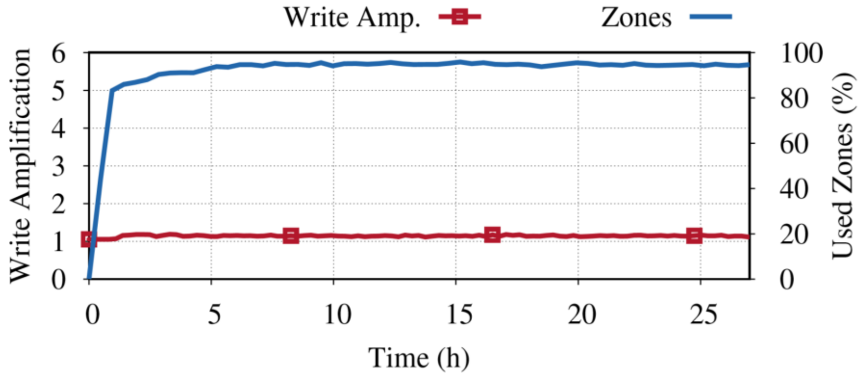


Workload: Fill 7B keys, Overwrite 7B keys



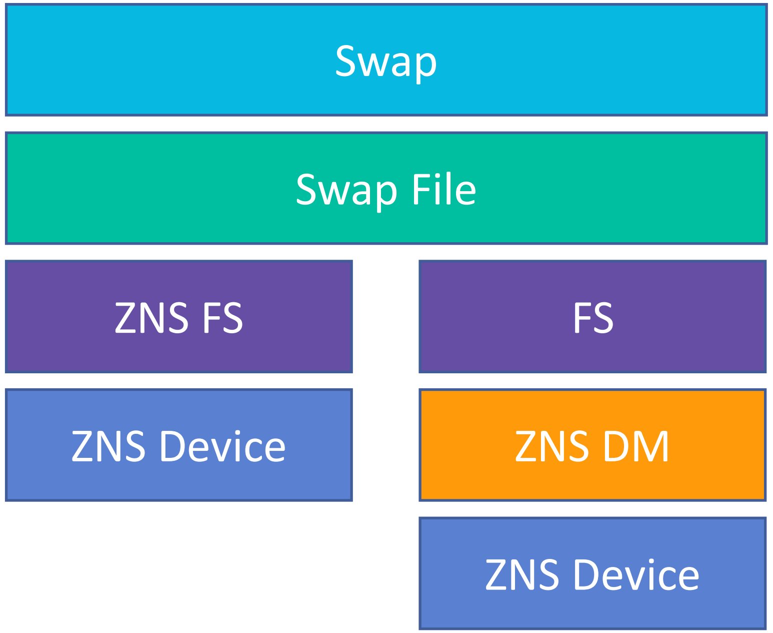
5x

Write Throughput

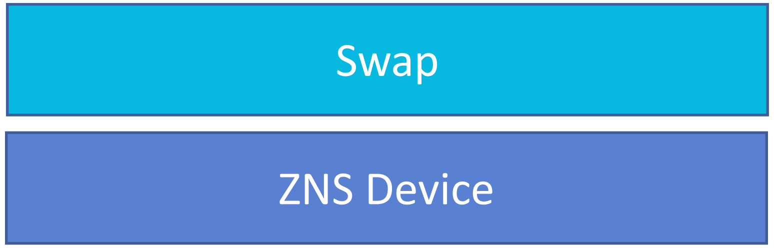


ZNS + Append + SWAP

- Layered Approach
  - Defer ZNS handling to FS or DM layers
  - Works, but duplicated MD between SWAP and ZNS handling layer



- Direct ZNS Append Support
  - No MD duplication
    - Swap maintains pte to swap location on device
  - Append Support
    - May enable performance optimizations based on drive selecting swap location



Accelerating Distributed Storage

Uncoordinated writes to Zones

Allocation of Zones

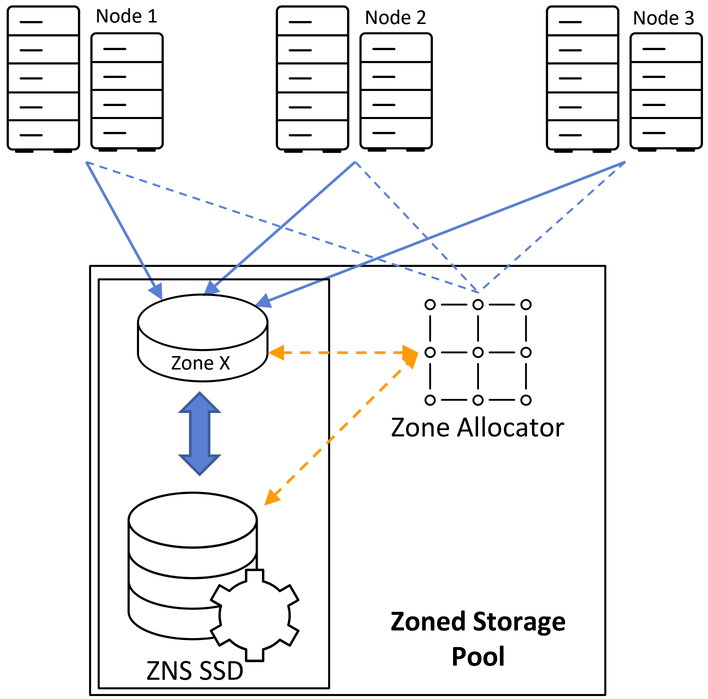
1. Node X requests a writeable zone
2. Zone Allocator returns Zone X
3. Node Y requests a writeable zone
4. Zone Allocator return Zone X

…

Request New Zone when Old is Full

1. Node X requests a writeable zone
2. Zone Allocator returns Zone X
3. Node X writes to zone…
4. Note X retrieves a Write error (Zone is Full)
5. Node X requests a new writeable zone.

Result: Reduces 4K block allocation decisions to 1/Zone Capacity



```
# 3 Processes, One SSD
DistributedAppend -n 3 -wr 10 /dev/nvme0n1
Creating 3 processes:
Wait between writes: 10ms
-------------------------
P1: Request new zone: Zone 0 returned.
P2: Request new zone: Zone 0 returned.
P1: Writing 4K to zone – Assigned LBA: 0
P2: Writing 4K to zone – Assigned LBA: 1
P3: Request new new: Zone 0 returned.
P3: Writing 4K to zone – Assigned LBA: 2
P2: Writing 4K to zone – Assigned LBA: 3
P1: Writing 4K to zone – Assigned LBA: 4
…
P3: Writing 4K to zone – Write error.
P3: Request new zone: Zone 1 returned.
P2: Writing 4K to zone – Write error.
P2: Request new zone: Zone 1 returned.
```

P1 LBAs: Zone 0 [0,4,…]

P2 LBAs: Zone 0 [1,3,…]

P3 LBAs: Zone 0 [2,…]

Western Digital.

Western Digital®