ZNS: Enabling In-place Updates and Transparent High Queue-depths

Javier González – Principal Software Engineer
Kanchan Joshi – Staff Engineer
Samsung Electronics
Why do we need a new interface?

- SSDs are already mainstream
  - Great performance (Bandwidth / Latency) – Combination of NAND + NVMe
  - Easy to deploy - Direct replacement for HDDs
  - Acceptable price $/GB

- But, we have 3 recurrent problems:

1. **Log-on-log Problem (WAF + OP)**
   - Remove redundancies
   - Leverage log data structures
   - Remove device data movement (GC)

2. **Multi-tenancy everywhere**
   - Address noisy-neighbor
   - Provide QoS

3. **Cost Gap with HDDs**
   - Higher bit count (QLC)
   - Reduce DRAM
   - Reduce OP & WAF


ZNS Use Cases

Archival

- **Motivation**
  - Facilitate adoption QLC
  - Reduce TCO: ↓WAF, ↓OP, ↓DRAM

- **Adoption**
  - Tiering for cold storage
    - Denmark cold: ZNS SSDs
    - Finland cold: High-capacity SMR HDDs
    - Mars cold: Tape
  - Leveraged zoned ecosystem from SMR

- **Zone Configuration**
  - Large zones
  - Immutable per-zone data
  - Need for large QDs

Log I/O

- **Motivation**
  - Leverage existing flash-friendly data structures
  - Facilitate adoption of QLC + TLC co-existence
  - Reduce TCO: ↓WAF, ↓OP, ↓DRAM

- **Adoption**
  - General storage systems using log-structures
    - Log-structured databases & file systems
  - Adopt zoned ecosystem for direct replacement

- **Zone Configuration**
  - Variable zone sizes supported
    - Small sizes for more control over placement and sched.
    - Large sizes SSD-managed placement
  - Host-side stripping & sched. across domains
  - Immutable per-zone data

I/O Predictability

- **Motivation**
  - Provide QoS for multi-tenant workloads
  - Address Noisy Neighbor & Flatmate problem
  - Facilitate adoption of QLC + TLC co-existence
  - Reduce TCO: ↓WAF, ↓OP, ↓DRAM

- **Adoption**
  - High multi-tenant environments
  - Workloads with strict QoS requirements

- **Zone Configuration**
  - Zones grouped in isolation domains (ID)
  - Zone domain mgmt. in provisioning layer
  - Host-side stripping & sched. across domains
Zoned Namespaces (ZNS)

**Data Placement & Zone Mgmt.**

- **Device-side zone mapping**
  - Different across ZNS products
  - Not yet in NVMe ZNS interface

<table>
<thead>
<tr>
<th>Zone 0</th>
<th>Zone n-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config. A</td>
<td>Zone 0</td>
</tr>
<tr>
<td></td>
<td>Zone k-1</td>
</tr>
<tr>
<td>Config. B</td>
<td>Zone n-k</td>
</tr>
<tr>
<td></td>
<td>Zone n-1</td>
</tr>
<tr>
<td>Config. Z</td>
<td>Die 0</td>
</tr>
</tbody>
</table>

- **Device managed zone state machine**
  - Host-driven transitions (most of the time)

**Write Operation**

- **Write Path** - QD=1 at a zone granularity
- **Append Path** - QD<=X / X = #LBAs in zone

**Spec. Status**

- Technical proposals ratified and published
  - TP 4053: Core ZNS specification
  - TP 4056: Namespace Types
  - TP 4061: Simple Copy

**Link:** [https://nvmexpress.org/developers/nvme-specification/](https://nvmexpress.org/developers/nvme-specification/)
Write Model

### Write @ QD1
- **I/O Path**
  - Traditional write operations
  - Limited to queue-depth 1 per zone
- **Use Cases**
  - Zoned replacement for block SSDs
  - Per-zone write performance is not critical
    - RAID, Block stripping
- **Zone Configuration**
  - Viable for all zone sizes
  - Small zones sizes can leverage zone stripping

![Write Command (QD1)](image)

### Append
- **I/O Path**
  - New I/O command
    - Nameless LBA to a given zone
    - Handle LBA in completion path
    - Queue-depth limited by zone size
- **Use Cases**
  - Applications writing to large zones and able to handle LBA re-mapping in completion path
- **Zone Configuration**
  - More benefit in larger zones

![Append (SQ)](image)

### Zone Stripping
- **I/O Path**
  - Traditional write operations
    - I/Os stripped across a zones / blocks
- **Use Cases**
  - Applications able to manage smaller zones and control / sched. data placement
- **Zone Configuration**
  - Benefit in smaller zones / multiple ZNS SSDs
  - Host sees a zones and zone groups (ID) and manages them, independently of the device

![Stripped Write Commands (QD1 per each zone)](image)

---

2020 Storage Developer Conference. © Samsung Electronics. All Rights Reserved.
Zone Random Write Area (ZRWA)

**Motivation & Operation**

**Motivation**
- Support out-of-order writes to a given zone
  - Enable larger QD in a zone
  - Enable in-place updates

**Mechanism**
- Expose a write buffer in front of a zone to the host
  - Select ZRWA during open operation
  - Sliding window in front of WP
- Operation:
  - Writes are placed into the ZRWA - no write pointer constraint
  - In-place updates allowed in the ZRWA window
  - ZRWA can be committed explicitly using a dedicated command
  - ZRWA can be committed implicitly when writing over sizeof(ZRWA)

**Use Cases**

**Need for larger QDs using traditional writes**
- Applications where mapping takes place in the submission path
  - E.g., RAID
- Operation
  - Hide the write pointer QD=1 constraint on the write path
  - Enable out-of-order writes to a zone
  - Enable larger zone configurations

**Need for in-place updates**
- Applications where small updates around the WP are required before data becomes immutable
  - E.g., metadata updates at specific checkpoints
- Operation
  - Write within ZRWA window: WP + sizeof(ZRWA)
  - Send explicit commit OR write past the ZRWA window (implicit commit)

**Spec. Status**

**Under development in NVMe – ZNS Taskforce**
- Operation and mechanisms in place
- Still subject to changes
- Join the discussions on Tuesdays!
Simple Copy

Motivation & Operation

**Motivation**
- Reduce the overhead added to GC in ZNS
- Garbage Collection on ZNS SSDs using existing mechanisms
  - Read data from device to host, remap and write back to a new zone
  - Creates extra traffic over the fabric
  - Creates extra memory and CPU footprint in the host

**Mechanism**
- Create a new command that offloads data movement to device
  - Source: SGL of LBA ranges (can be different zones)
  - Destination: Single LBA range (single zone)
- Operation
  - Select a number of source zones to garbage collect
  - Select a destination zone for moving data
  - Send a Simple Copy Command

**Use Cases**

**Garbage Collection for ZNS**
- ZNS moves GC to the host due to explicit zone reset
  - GC through Read + Write
  - Creates extra traffic over the fabric
  - Buffer management increases host’s memory and CPU footprint
- Operation
  - Hide the write pointer QD=1 constraint on the write path
  - Enable out-of-order writes to a zone
  - Enable larger zone configurations

**Need for in-place updates**
- Applications where small updates around the WP are required before data becomes immutable
  - E.g., metadata updates at specific checkpoints
- Operation
  - Write within ZRWA window: WP + sizeof(ZRWA)
  - Send explicit commit OR write past the ZRWA window (implicit commit)

**Spec. Status**

**Ratified and published**
- Applies to NVMe 1.4 specification
- TP 4065a – Simple Copy Command
- Link: https://nvmexpress.org/wp-content/uploads/NVM-Express-1.4-Ratified-TPs-1.zip
Linux Ecosystem

- Zoned ecosystem has grown significantly since ZNS publication
- Ecosystem backed by several vendors
- ZNS supported in Linux 5.9

**User-Space**
- Libraries
  - Enable easy adoption of zoned devices
  - Facilitate support for classes of applications
- Management Tools
  - Required for adoption in real deployments
- Test / Evaluation
  - Extend support and test cases for new write model

**Kernel-Space**
- Extend Zoned Block Framework
  - Build on top of infrastructure for SMR HDDs
- Align with I/O model based on Append
  - Keep unified write model for zoned devices
  - Extensions targeting co-existence
- Increased activity since ZNS TP release

**Emulation**
- Facilitate development of SW stack
- Compliance and performance evaluation

---

2020 Storage Developer Conference. © Samsung Electronics. All Rights Reserved.
Linux Kernel: User-append, ZRWA, simple-copy

### Raw block-device interface
- Make ZNS features consumables to zone-aware applications
- Target: User-space FS/DB/SDS which prefer to do things by themselves
- Less abstraction, more control, more flexibility

### Challenge
- How to return “written-location” to user-space
- How to ask for zone-append

### Io_uring
- Pass a pointer along with SQE
- Update the pointer with written-location (before posting CQE)
- Trigger: RWF_APPEND + flag to report offset in directly (pointer)

### Linux AIO
- Use the field “res2” to return
- Trigger: RWF_APPEND + flag to report offset directly

### Block-device
- Send zone-append (instead of write) and return offset to caller

### Setup
- IOCTL (zone-mgmt) to attach/remove ZRWA with a zone

### During I/O
- Travels as regular write

### Zone write-lock avoidance
- Make mq-deadline treat ZRWA-enabled zone as conventional (in-place update and multi-QD write)

### Append
- IOCTL (zone-mgmt) to attach/remove ZRWA with a zone

### During I/O
- Travels as regular write

### Simple-Copy
- Bio/Request with new opcode REQ_OP_COPY
- Expect source-lba lists in payload, and destination-lba as write-location

### Io_uring
- Opcode IORING_OP_COPY, similar to write

---

2020 Storage Developer Conference. © Samsung Electronics. All Rights Reserved.
F2FS: ZRWA & Simple Copy

**F2FS on Zoned Devices**
- Max 6 open zones (Hot/Warm/Cold – Data and Node)
- Allocation/GC unit is “section”- collection of fixed-size segments (2MB each)
- Configurable section size, mapped to device zone
- “Strict” LFS mode: avoid writing to holes

**Challenges**
- F2FS Meta requires in-place update. ZNS does not have conventional-zones. Multi-device setup (CNS + ZNS) is needed.
- Large zones (section) may lead to more movement during GC. May affect user-operations during foreground GC.
- QD1 writes on zone : speed dampeners

**Solutions**
- Reduce on-media writes
  
  ```
  if (ZRWA window (old_blkaddr) == ZRWA window (new_blkaddr))
  Do in-place update;
  ```
- Move some meta to ZNS itself (e.g. checkpoint)
- Higher queue-depth using Append/ZRWA
- Simple-copy: offload GC data-movement to device

---

**F2FS Volume**

- Random writes
- Sequential Writes

**Sections (Data)**
- Cold/Warm/Hot

**Sections (Node)**
- Cold/Warm/Hot

**CP** | **SIT** | **NAT** | **SSA**
--- | --- | --- | ---

**CNS Device**

**ZNS Device**
Linux Kernel: Performance characterization

**ZRWA**

- Linear scalability within ZRWA window
  - Also across several zones
- Makes sense for metadata writes
  - Leverage in-place update at no BW cost
  - 4KB at moderate / high QD

**Simple-Copy**

- Better use of fabric bandwidth
  - Lower latency, higher bandwidth
  - Scales with block size (SGL for source LBA ranges)
- Lower host CPU utilization
  - Offload data movement
  - Limited to I/O submission

---

**Metal (4K, Write, io-uring, QD 16)**

- Bandwidth (MiB/s)
  - Striping (number of zones)
  - Disabling ZRWA shows a decrease in performance.

---

**96 MiB copy operation**

- Time taken (ms/c)
  - Block Size
  - Read + Write and Simple-Copy comparison
Simon Lund: xNVMe: Programming Emerging Storage Interfaces for Productivity and Performance
QEMU Emulated NVMe Device

Klaus Jensen: Reviving The QEMU NVMe Device (from Zero to ZNS)

Status

- **Improved NVMe Device in QEMU**
  - v1.3 support, cleanups & refactoring merged in Q2 2020

- **Submitted & Reviewed (pending upstream merge)**
  - SGLs, multiple namespaces

- **Submitted (under review):**
  - v1.4 support
  - metadata (separate and extended LBAs)
  - DULBE
  - I/O Command Sets & Zoned Namespaces

- **Submitted (pending reviews)**
  - Simple Copy Command

- **Upcoming**
  - Compare, DIF/DIX & Verify, Write Uncorrectable
  - ZRWA
  - NVMe Low-latency Mode (i.e., QEMU null_blk device)
LSM-based Databases – ZNS through xZTL

Enable RocksDB with thin backend: ~1000 LOC

Add zoned logic for LSM DBs in xZTL
- Easy to port other DBs (e.g., Cassandra)
- Transparent support for several ZNS architectures and I/O models
  - Append: Large zones
  - Stripping: Small zones
- Support for all ZNS functionality
  - E.g., ZRWA, Simple Copy
- Easy to enable Vendor / Customer Unique features
  - xZTL: https://github.com/OpenMPDK/xZTL
  - RocksDB w/ xZTL: https://github.com/ivpi/rocksdb

Tight integration with xNVMe
- Leverage changes in application to run on multiple OSs and I/O backends
  - Linux: io_uring, libaio, SPDK, IOCTL
  - FreeBSD: SPDK
  - Windows (ongoing)

RocksDB - Evaluation
- 2X WAF improvement in SW
- 5-15X WAF improvement in total
- ZRWA: In-place updates
- No write BW degradation (within ZRWA window)
Conclusion

- ZNS is the NVMe way of performing host data placement
  - Reduces TCO: ↑ Capacity (QLC), ↓WAF, ↓OP, ↓DRAM
  - Improves I/O determinism: Host orchestrates zone GC

- Use cases are increasing as ZNS SSDs are available
  - From Archival to I/O determinism
    - Different I/O models supported
  - Extensions to ZNS facilitating transition from Open-Channel SSD architectures
    - ZRWA and Simple Copy are just the beginning

- Solid ecosystem in Linux
  - Several vendors contributing and adding new features
  - Core support in Linux kernel: NVMe driver, block layer & file systems

- Cross I/O path support in xNVMe
  - Single API for all I/O backends
  - Support for Linux, FreeBSD and Windows
  - Support for io_uring, libaio, SPDK and IOCTL

- First applications with upstream zoned support ongoing
  - Working on libraries to facilitate support in classes of applications (xZTL → LSM-based DBs)

Talk to us about ZNS! - javier.gonz@samsung.com & joshi.k@samsung.com