ZNS: Avoiding the Block Interface Tax for Flash-based SSDs

Matias Bjørling, Abutalib Aghayev, Hans Holmberg, Aravind Ramesh, Damien Le Moal, Gregory R. Ganger, George Amvrosiadis

July 14-16
USENIX ATC 2021
The Block Interface Tax

For several decades storage software has been built atop the block interface

- Storage represented as an array of fixed-size blocks
- Each block can be read, written, and overwritten atomically
- Adopted for HDDs as well as SSDs

Hard Disk Drive (HDD) → Solid-State Drive (SSD)
The Block Interface Tax

The inherent properties of flash-based SSDs have made the block interface a poor fit
▪ SSDs “append” pages to erase blocks, need to erase whole block before rewriting
▪ Data placement overhead: media over-provisioning (7-28%), higher $cost and lower performance

Block Interface Compatibility increases $cost, hurts performance
Zoned Namespace SSDs

Getting rid of the block interface tax

What if the host could write data onto the flash-based SSD through append-only regions (zones)? → ZNS exposes them!

- No fine-grained data placement in SSDs: **+7-28% capacity, lower $cost, predictable high performance**

Overwriting a block in a zone requires erasing/rewriting entire zone
Zoned Namespace SSDs

Getting rid of the block interface tax
Zoned Namespace SSDs
Getting rid of the block interface tax

The Catch: No overwrites/out-of-order writes allowed under ZNS. Only works if software layers above are modified to support this limitation.

Research opportunity*: Which applications can evolve to use the ZNS interface? How?

*Stavrinos et. al., Don't be a Blockhead: Zoned Namespaces Make Work on Conventional SSDs Obsolete, HotOS, 2021
Evolving towards ZNS SSDs

- ZNS SSDs relinquish GC responsibilities traditionally carried out by the FTL.
- The ZNS interface enables the SSD to translate sequential zone writes onto distinct erase blocks.
- Since random writes are disallowed by the interface, and zones must be explicitly reset by the host, the data placement occurs at the coarse-grained level of zones.
- GC of zones becomes the responsibility of the host.
- Media reliability continues to be the full responsibility of the SSD.
Adoption
Three ways to adopt ZNS SSDs

- Host-side FTL
  - Implement a host-side FTL that exposes the ZNS SSD as a block interface SSD.
  - High system overhead wrt to DRAM and CPU.
  - Enable workloads that specifically require random write characteristics.

- File Systems (f2fs /w zones)
  - Place data onto zones using the file system characteristics
  - Efficient use of resources, as the file system simply places data more efficiently
  - Layer of indirection away from the application, and therefore some inefficient data placement causes host GC.

- End-to-end Data Placement (RocksDB /w ZenFS)
  - Places data onto zones using the application characteristics
  - No indirection overhead cause by FTL data placement nor file system.
  - Highest performance and the lowest write amplification
Enabling the Linux Ecosystem
Adding support for ZNS SSDs

- General Linux Support thru the Zoned Block Device (ZBD) subsystem
- NVMe driver support for zone attributes (e.g., capacity)
- API support for exposing limit of active zones, which depends on device resources
- Linux file system support: extending f2fs to run on ZNS

<table>
<thead>
<tr>
<th></th>
<th>Lines Added</th>
<th>Lines Removed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux Kernel</td>
<td>647</td>
<td>53</td>
</tr>
<tr>
<td>f2fs (kernel)</td>
<td>275</td>
<td>37</td>
</tr>
<tr>
<td>f2fs (tools)</td>
<td>189</td>
<td>15</td>
</tr>
<tr>
<td>fio</td>
<td>342</td>
<td>58</td>
</tr>
<tr>
<td>ZenFS (RocksDB)</td>
<td>3276</td>
<td>2</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>4729</strong></td>
<td><strong>169</strong></td>
</tr>
</tbody>
</table>
ZenFS Architecture
A new storage backend for RocksDB

- Extent-based block-aligned contiguous region of file data
  - Multiple file extents per zone (no spanning)
- Journal data: appended to circular buffer of designated zones
  - Includes WAL data, file identifiers, in-memory allocation structures
  - Buffered writes handled by buffering in memory until flush event
- Zone management
  - User limit for internal fragmentation simplifies file size uncertainty (due to compression, compaction)
  - Write lifetime hints from RocksDB simplify Garbage Collection
  - Limits active zones based on device resources
Evaluation

Apples-to-apples comparison

- Production hardware platform that can expose itself as either a block-interface SSD or a ZNS SSD.

- Methodology
  - Raw I/O performance
  - RocksDB Performance
    - XFS, F2FS (Block)
    - F2FS /w zone support (ZNS)
    - RocksDB /w ZenFS (ZNS)

Feature summary of the evaluated SSDs

<table>
<thead>
<tr>
<th>SSD Interface</th>
<th>Block</th>
<th>Block</th>
<th>Zoned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media Capacity</td>
<td>2TiB</td>
<td>2TiB</td>
<td>2TiB</td>
</tr>
<tr>
<td>Host Capacity</td>
<td>1.92TB</td>
<td>1.6TB</td>
<td>2TB</td>
</tr>
<tr>
<td>Over-provisioning</td>
<td>7%</td>
<td>28%</td>
<td>0%</td>
</tr>
<tr>
<td>Placement Type</td>
<td>None</td>
<td>None</td>
<td>Zones</td>
</tr>
<tr>
<td>Max Active Zones</td>
<td>N/A</td>
<td>N/A</td>
<td>14</td>
</tr>
<tr>
<td>Zone Size</td>
<td>N/A</td>
<td>N/A</td>
<td>2048 MiB</td>
</tr>
<tr>
<td>Zone Capacity</td>
<td>N/A</td>
<td>N/A</td>
<td>1077 MiB</td>
</tr>
</tbody>
</table>
Raw I/O Characteristics
Improving Write Throughput & Read Latency

- ~380MiB/s / 7% OP
- ~600MiB/s / 28% OP
- ~1GiB/s / ZNS SSD

Read Latency scales linearly with write load

ZNS SSD (0% OP)  |  Block SSD (7% OP)  |  Block SSD (28% OP)

© 2021 Western Digital Corporation or its affiliates. All rights reserved.
RocksDB: Writes
Double the throughput over 28% OP SSDs

- XFS and F2FS overprovisioning at 28%
- Fillrandom begins at clean state. Overhead visible when overwriting
- Write Amplification for ZNS is 1.0x
  - XFS at 2.0x and vanilla F2FS at 2.4x
RocksDB: Reads and Writes
Improving Writes and Tail Latencies

- When writes are limited to 20MB/s
  - Only ZNS achieves write goal, others 15% lower

- When writes are not limited
  - ZNS SSD write throughput 2x higher

- RocksDB on ZNS achieves up to 4x lower 99.99th-percentile read latency, 2x write throughput
Summary

- ZNS SSDs enable higher performance and lower-cost-per-byte flash-based SSDs.

- By shifting responsibilities for managing data placement within erase blocks from FTLs to host software, ZNS eliminates the need for fine-grained indirection table, garbage collection, and media over-provisioning.

- We find that the 99.9th-percentile random-read latency for our RocksDB /v ZenFS is at least 2-4x lower on a ZNS SSD compared to a block-interface SSD, and the write throughput is 2x higher.

- All work is upstream and available through the appropriate open-source projects.
Thank You