Zoned Namespaces

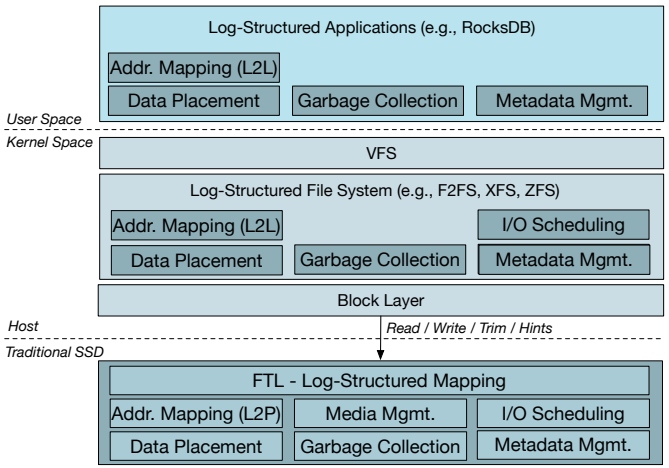
Use Cases, Standard and Linux Ecosystem

SDC SNIA EMEA 20 | Javier González – Principal Software Engineer / R&D Center Lead

Why do we need a new interface?

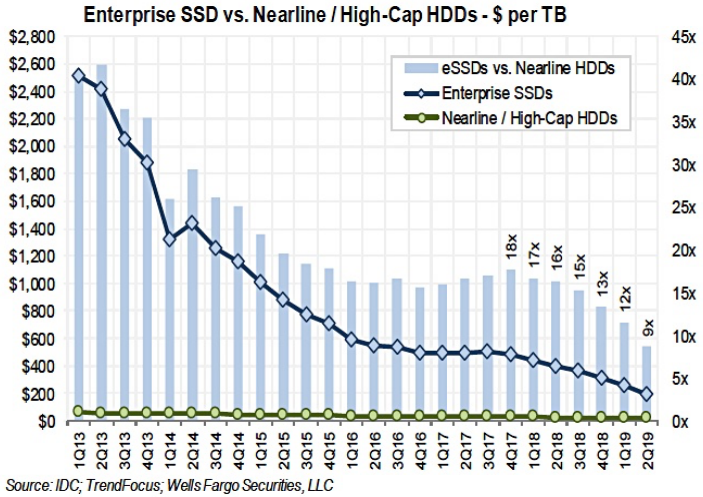
- **SSDs are already mainstream**
  - Great performance (Bandwidth / Latency) – Combination of NAND + NVMe
  - Easy to deploy - Direct replacement for HDDs
  - Acceptable price $/GB
- **But, we have 3 recurrent problems:**

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



1.J. Yang, N. Plasson, G. Gillis, N. Talagala, and S. Sundararaman. Don't stack your
2.log on my log. In 2nd Workshop on Interactions of NVM/Flash with Operating
3.Systems and Workloads (INFLOW), 2014.

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



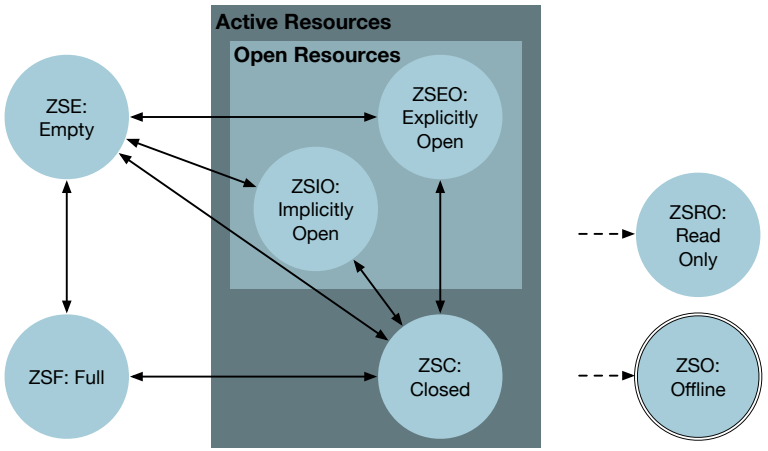
Source: IDC; TrendFocus; Wells Fargo Securities, LLC

https://blocksandfiles.com/2019/08/28/nearline-disk-drives-ssd-attack/

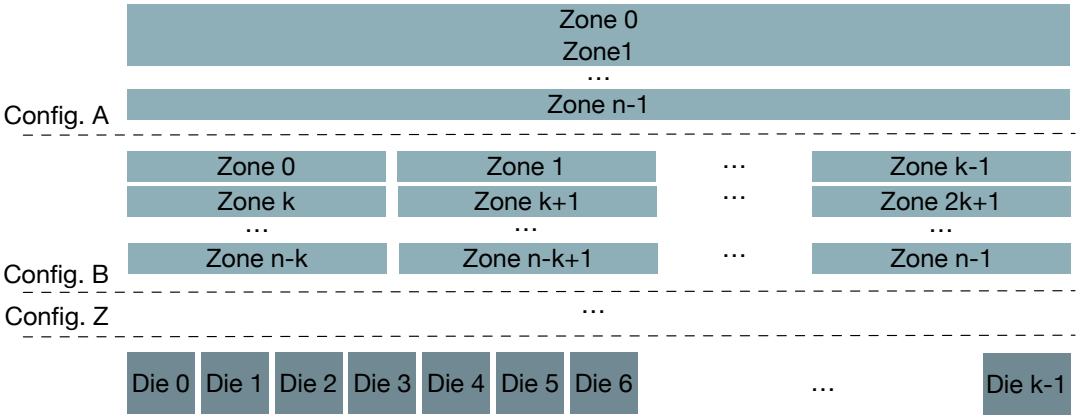
3. Multi-tenancy everywhere
- Address noisy-neighbor
- Provide QoS

ZNS: Basic Concepts

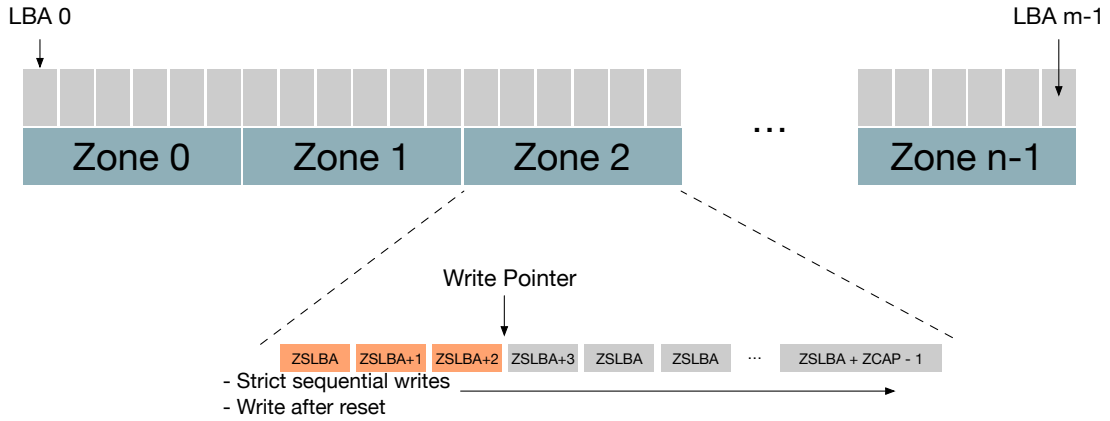
- **Divide LBA address space in fixed size zones**
  - Write Constraints
    - Write strictly sequentially within a zone (Write Pointer)
    - Explicitly reset write pointer
  - No zone mapping constraints
    - Zone physical location transparent to host
    - Open design space for different configuration
- **Zone state machine**
  - Zone transitions managed by host
    - Direct commands or boundaries
  - Zone transitions triggered by device
    - Notification to host through a exception



Zone mapping in device



LBA mapping in zone



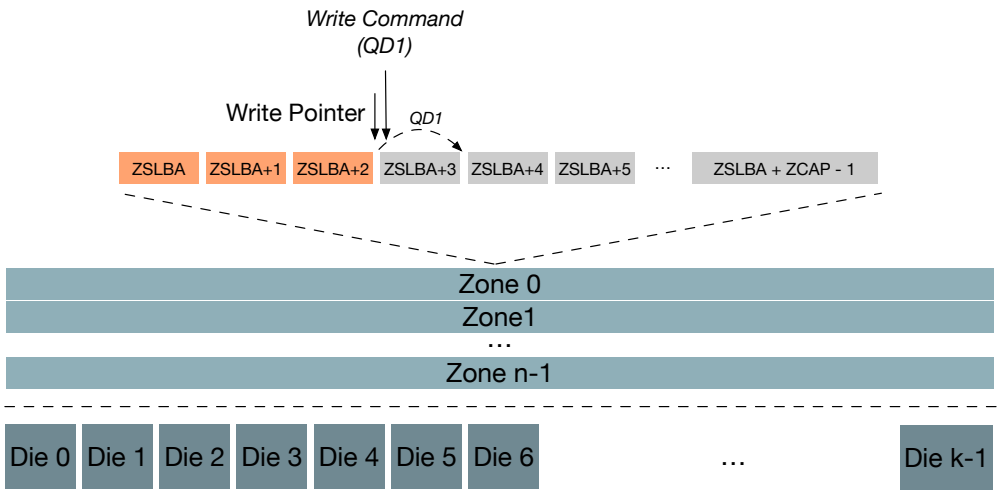
ZNS: Advanced Concepts

Append Command

- Implementation of nameless write [1]
- Benefits
  - Increase QD in a single zone to improve parallelism
- Host changes
  - Changes needed to block allocator in application / file system
  - Full reimplementation of LBA-based features in submission path (e.g., snapshotting)

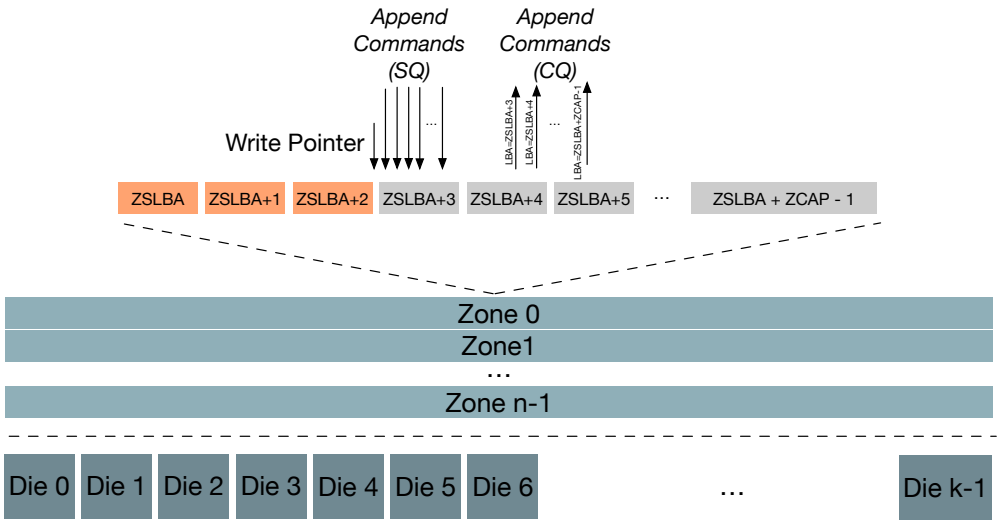
Write Path

- Required QD=1
  - Reminder: No ordering guarantees in NVMe



Append Path

- QD<=X / X = #LBAs in zone

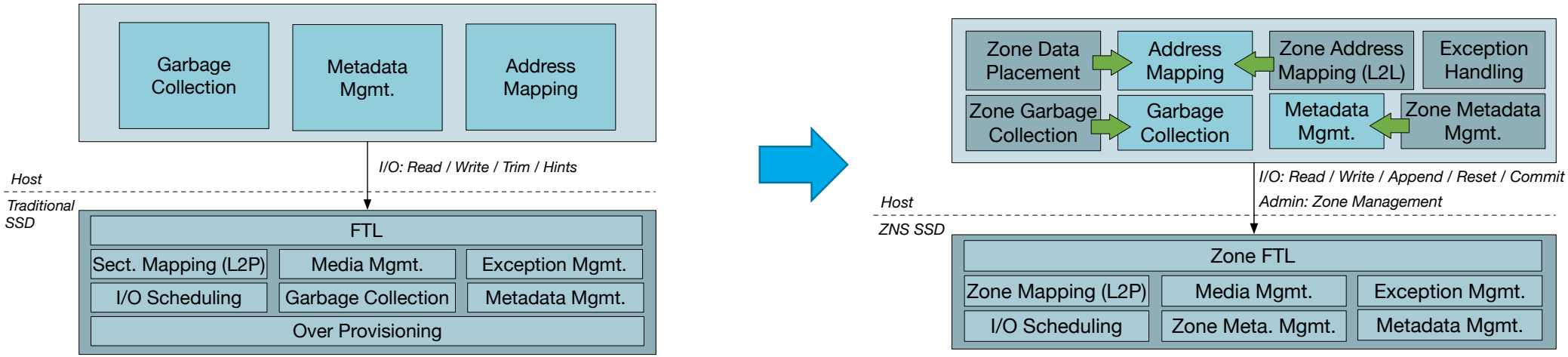


[1] Y. Zhang, L. P. Arulraj, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. De-indirection for flash-based SSDs with nameless writes. FAST, 2012.

ZNS: Host Responsibilities

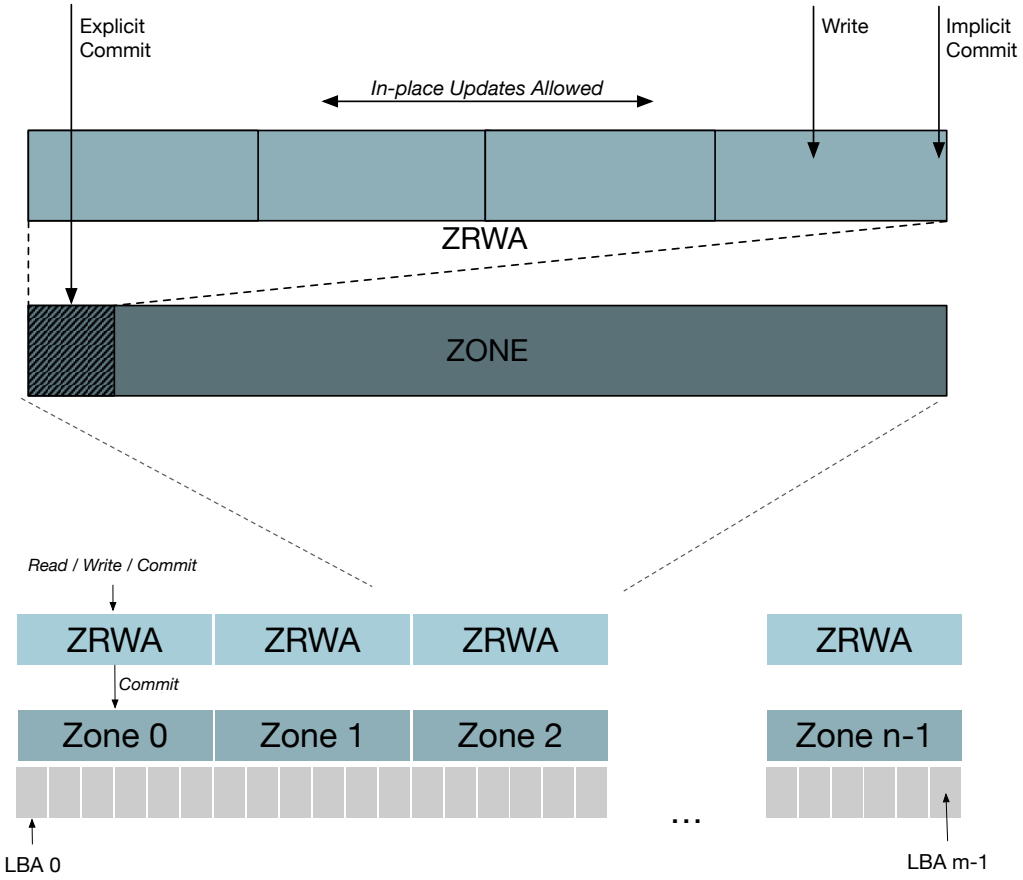
ZNS requires changes to the host

- First step: Changes to file systems
  - Advantages in log-structured file systems (e.g., F2FS, XFS, ZFS)
  - Still best-effort from application perspective
  - First important step for adoption
- Second step: Changes to applications
  - Only make sense to log-structure applications (e.g., LSM databases such as RocksDB)
  - Real benefits in terms of WAF, OP and data movement (GC)
  - Need a way to minimize application changes! (Spoiler Alert: xNVMe)



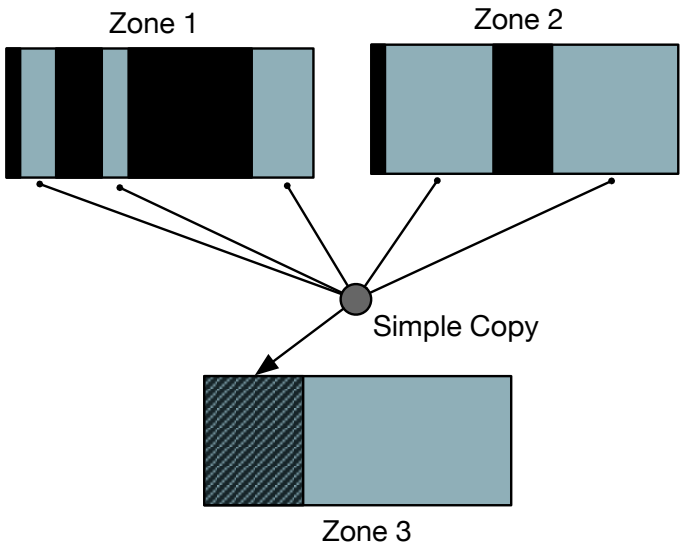
ZNS Extensions: Zone Random Write Area (ZRWA)

- **Concept**
  - Expose the write buffer in front of a zone to the host
- **Benefits**
  - Enable in-place updates within ZRWA
    - Reduce WAF due to metadata updates
    - Aligns with metadata layout of many file systems
    - Simplifies recovery (not changes to metadata layout)
  - Allows writes at QD > 1
    - No need changes to host stack (as opposed to append)
    - ZRWA size balanced with expected performance
- **Operation**
  - Writes are placed into the ZRWA
    - No write pointer constraint
  - In-place updates allowed in the ZRWA window
  - ZRWA can be committed explicitly
    - Through dedicated command
  - ZRWA can be committed implicitly
    - When writing over sizeof(ZRWA)



ZNS Extensions: Simple Copy

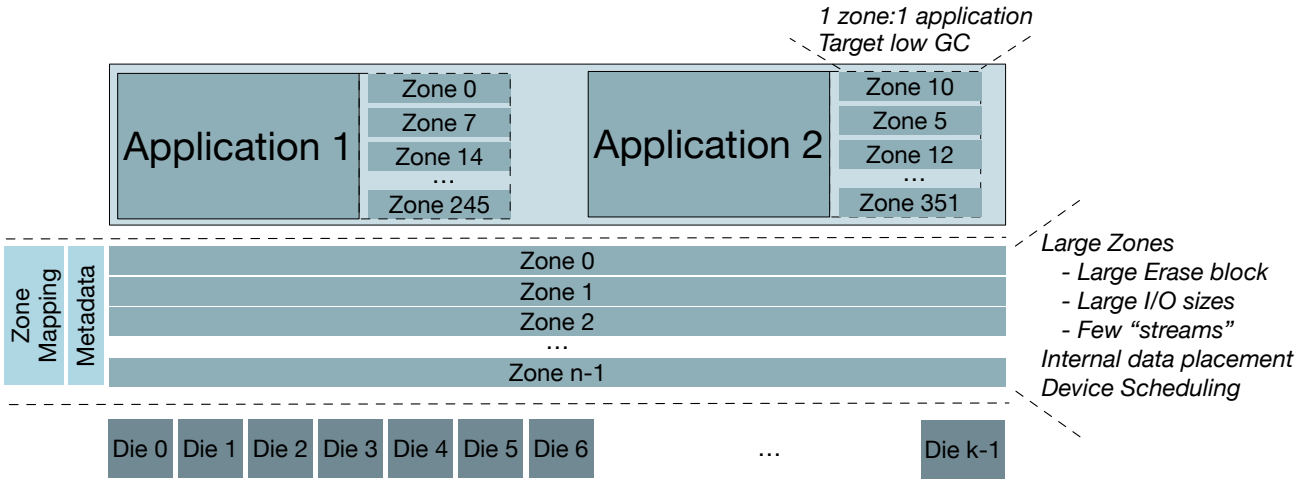
- **Concept**
  - Offload copy operation to SSD
  - Simple copy, because SCSI XCOPY become to complex
- **Benefits**
  - Data is moved by the SSD controller
    - No data movement through interconnect (e.g., PCIe)
    - No CPU cycles used for data movement
  - Specially relevant for ZNS – host GC
- **Operation**
  - Host forms and submits SC command
    - Select a list of source LBA ranges
    - Select a destination with a single LBA + length
  - SSD moves data from sources to destination
    - Error model complexity depending on scope
- **Scope**
  - Under discussion in NVMe



ZNS: Archival Use Case

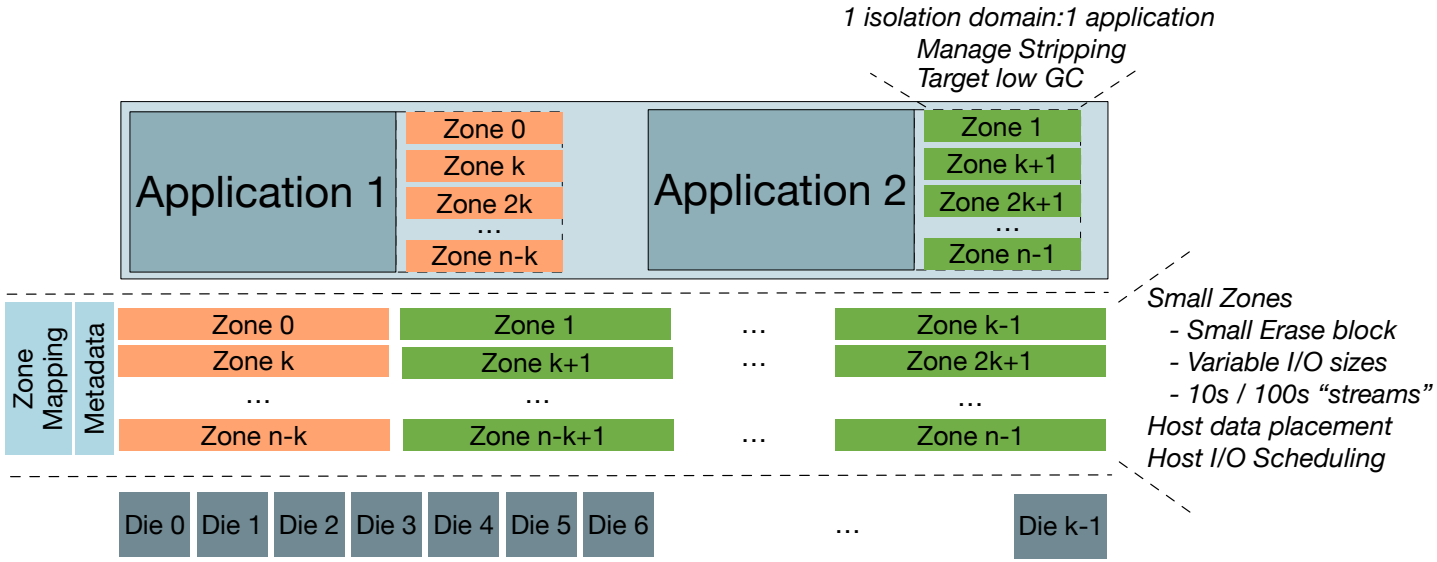
- **Goal: Reduce TCO**
  - Reduce WAF
  - Reduce OP
  - Reduce DRAM usage
  - Facilitate consumption of QLC
- **Adoption**
  - Tiering of cold storage
    - Denmark cold: ZNS SSDs
    - Finland cold: High-capacity SMR HDDs
    - Mars cold: Tape
  - Leverage SMR ecosystem
    - Build on top of same abstractions
    - Expand use-cases

- **Configuration optimized for cold data**
  - Large zones
  - Semi-immutable per-zone data (all / nothing)
  - Minimize metadata and mapping tables
  - Need for large QDs (Append or ZRWA)
- **Architecture properties**
  - Host GC → Less WAF and OP
  - Zone Mapping table → Less DRAM (break 1GB / 1TB)



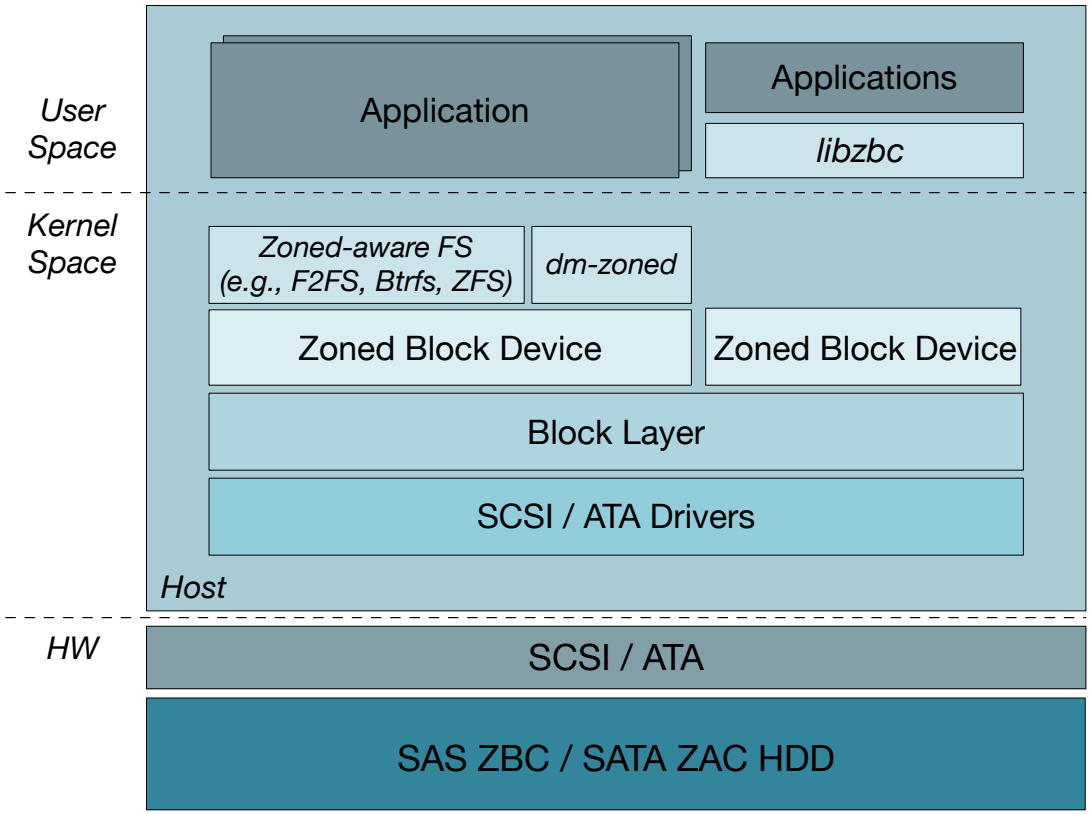
ZNS: I/O Predictability Use Case

- **Goal: Support multi-tenant workloads**
  - Inherit archival benefits:
    - ↓WAF, ↓OP, and ↓DRAM
  - Support QoS policies
    - Provide isolation properties across zones
    - Enable host I/O scheduling
    - Allow more control over I/O submission / completion
- **Configuration optimized for multi-tenancy**
  - Zones grouped in “isolation domains”
  - Host data placement / stripping
  - Host I/O scheduling
- **Inherit archival architecture properties**
  - Host GC → Less WAF and OP
  - Zone Mapping table → Less DRAM (break 1GB / 1TB)



Linux Stack: Zoned Block Framework

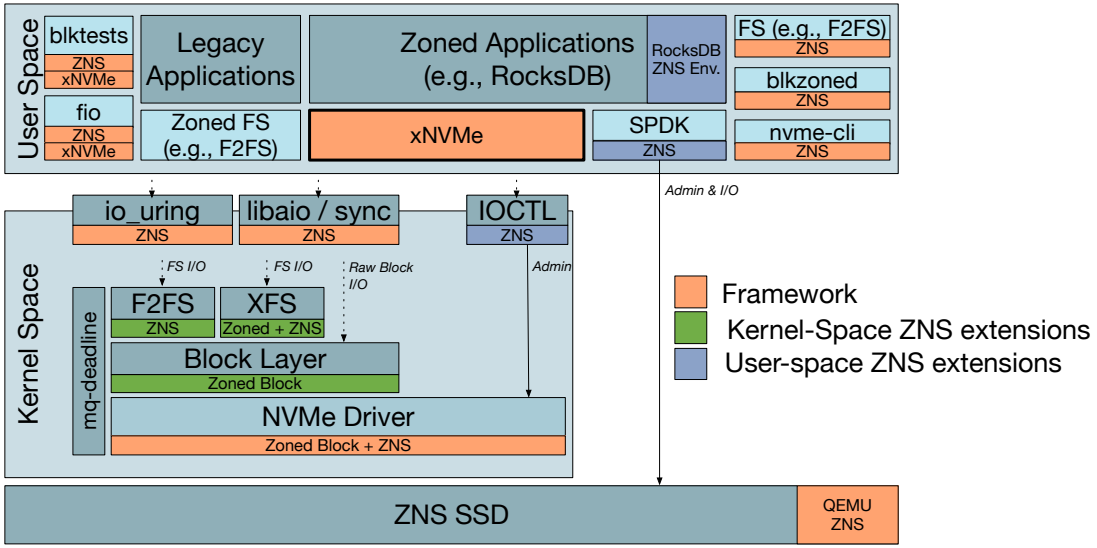
- **Purpose**
  - Built for SMR drivers
  - Add support for write constraints
  - Expose to applications through block layer
- **Kernel Status**
  - Merged in 4.9
  - Block / drivers
    - Support in block layer
    - Support in SCSI / ATA
    - Support in null_blk
  - File systems
    - F2FS
    - Btrfs (patches posted)
    - ZFS / XFS (ongoing)
  - Libraries
    - libzbc
  - Tools
    - Utils-linux, fio



Linux Stack: Adding ZNS support

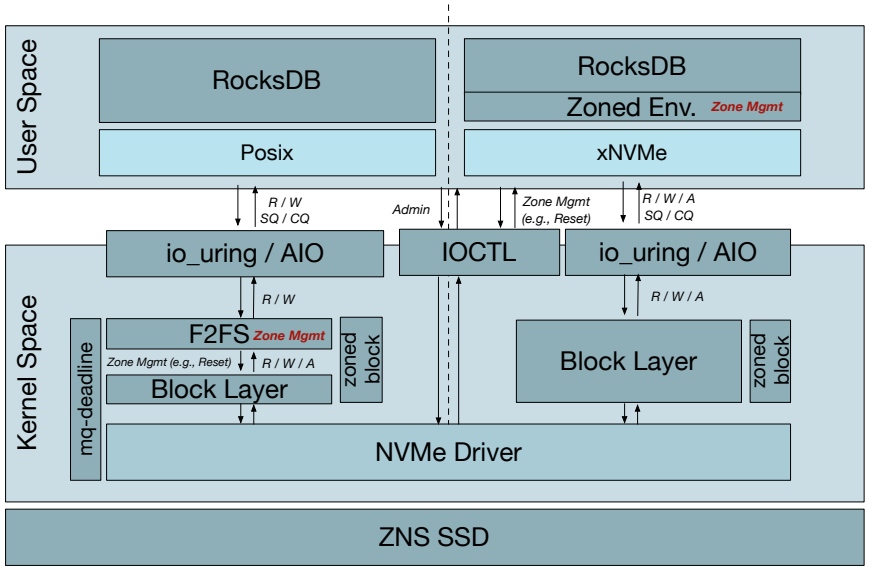
ZNS support in Zoned Block

- Add support ZNS-specific features (kernel & tools)
  - Append, ZRWA and Simple Copy
  - Async() path for new admin commands
- Relax assumptions from SMR
  - Conventional zones
  - Zone sizes, state transitions, etc.



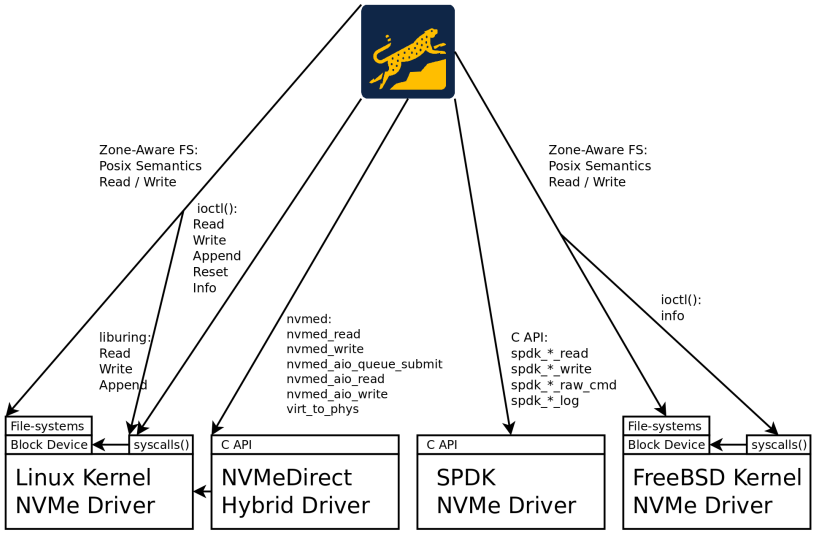
I/O paths

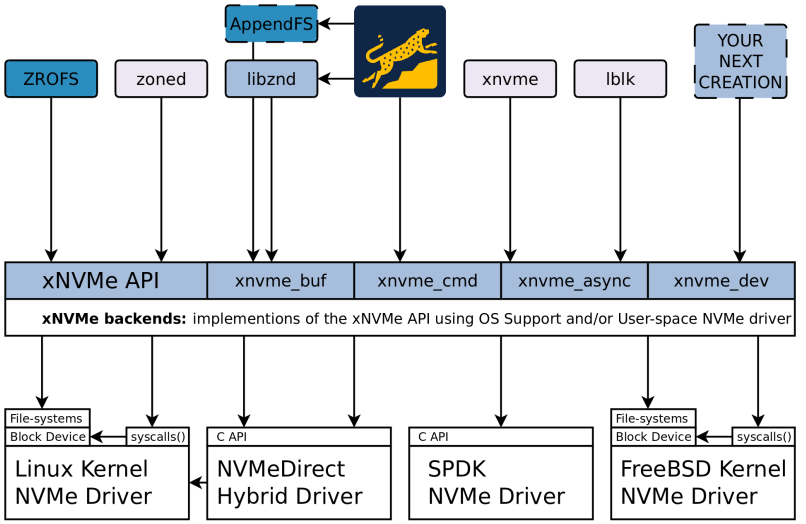
- Using zoned FS (e.g., F2FS)
  - No changes to application
- Using zoned application
  - Support in application backend
  - Explicit zone management
    - Data placement, sched., zone mgmt.



xNVMe: Supporting non-block Applications

- **xNVMe: Library to enable non-block applications across I/O backends**
  - Common API and helpers
  - Linux kernel path: psync, libaio, io_uring
  - SPDK: User-space in Linux and FreeBSD
  - Windows: Future work – new backend required
- **Benefits**
  - Portable API across I/O backends and OSs (minimal benefit for block devices)
  - Common API for new I/O interfaces (Re-write application backend only once)
  - File-system semantics on raw devices (block & non-block)
  - No performance impact





Takeaways

- **ZNS is a new interface in NVMe**
  - Builds on top of Namespace Types (e.g., KV)
  - Driven by vendors and early customers
- **ZNS targets 2 main use-cases**
  - Archival (original use case)
  - I/O Predictability (ZNS extensions)
- **Host needs changes**
  - Basic functionality builds on top of existing zoned block framework
    - Respect write constraints
    - Follow zone state machine: Reset->Open->Closed->Full / ->Offline / ->ReadOnly
  - Append needs major changes to block allocators in I/O path
    - Might work for some file systems / applications, but not for all
  - ZRWA gives an alternative to append
    - Same functionality, not changes to block allocator in I/O path
- **We are building open ecosystem**
  - Linux stack to be released on TP ratification
    - xNVMe, Linux Kernel, tools (e.g., nvme-cli), fio
    - RocksDB Backend on xNVMe
  - Sponsoring TPs for ZNS extensions in NVMe

THE NEXT CREATION STARTS HERE

Placing **memory** at the forefront of future innovation and creative IT life