The Denali Open-Channel Standard: Impact and Future

Javier González

Principal Software Engineer

SSDR R&D Center Leader

SAMSUNG

Open-Channel Solid State Drives

- Open-channel Definition
  - *“Class of Solid State Drives that (i) expose their internal geometry to the host and (ii) allow it to manage part of the FTL responsibilities through a (iii) Read/Write/Erase interface”*
- A bit of history…
  - Term introduced at Baidu paper [1] to optimize KV-store by accessing physical media
  - Pre Open-Channel SSDs implementations in the industry: Fusion I/O and others.
  - Open-Channel 1.2 / Physical Page Addressing (ppa) mode (2014/2015) [2]
  - Open-Channel 2.0 / Pre-Denali release (2016) [2]
  - Alibaba’s Dual-Mode SSD [3]
  - Project Denali (2018) [4]
  - Standardization in NVMe (2019)
- Open-Source Driven
  - Full Linux support & Open Specification

[1] **An efficient design and implementation of LSM-tree based key-value store on open-channel SSD** (Eurosys’14),
Peng Wang, Guangyu Sun, Song Jiang, Jian Ouyang, Shiding Lin, Chen Zhang, and Jason Cong
[2] *lightnvm.io*
[3] **In Pursuit of Optimal Storage Performance: Hardware/Software Co-Design with Dual-Mode SSD** (Alibaba white paper)
Yu Du, Ping Zhou, Shu Li
[4] **Denali: The Next-Generation High-Density Storage Interface** (OCP’18)
Laura Caulfield, Arie van der Hoeven

Open-Channel SSD Design Goals

Data Placement

What?

- Reduce WAF
- Increase SSD life and endurance
- Provide multi-tenant I/O isolation (noisy neighbor)
- Reduce TOC

How?

- Eliminate log-on-log problem (Application - FS - SSD)
- Eliminate device triggered I/O
- Integrate cross-layer GC
- Map tenants to physical PUs
- Reduce OP and device L2P

I/O Scheduling

What?

- Guarantee SLA QoS
- Reduce intra-tenant tail latencies (noisy flatmate)
- Utilize full device bandwidth
- Utilize full device capacity

How?

- Manage state of PU in application I/O sched.
- Control vertical and horizontal PU stripping
  - Vert.: ↑ Isolation / ↓ BW / ↓ Capacity
  - Horiz.: ↓ Isolation / ↑ BW / ↑ Capacity

Open Ecosystem

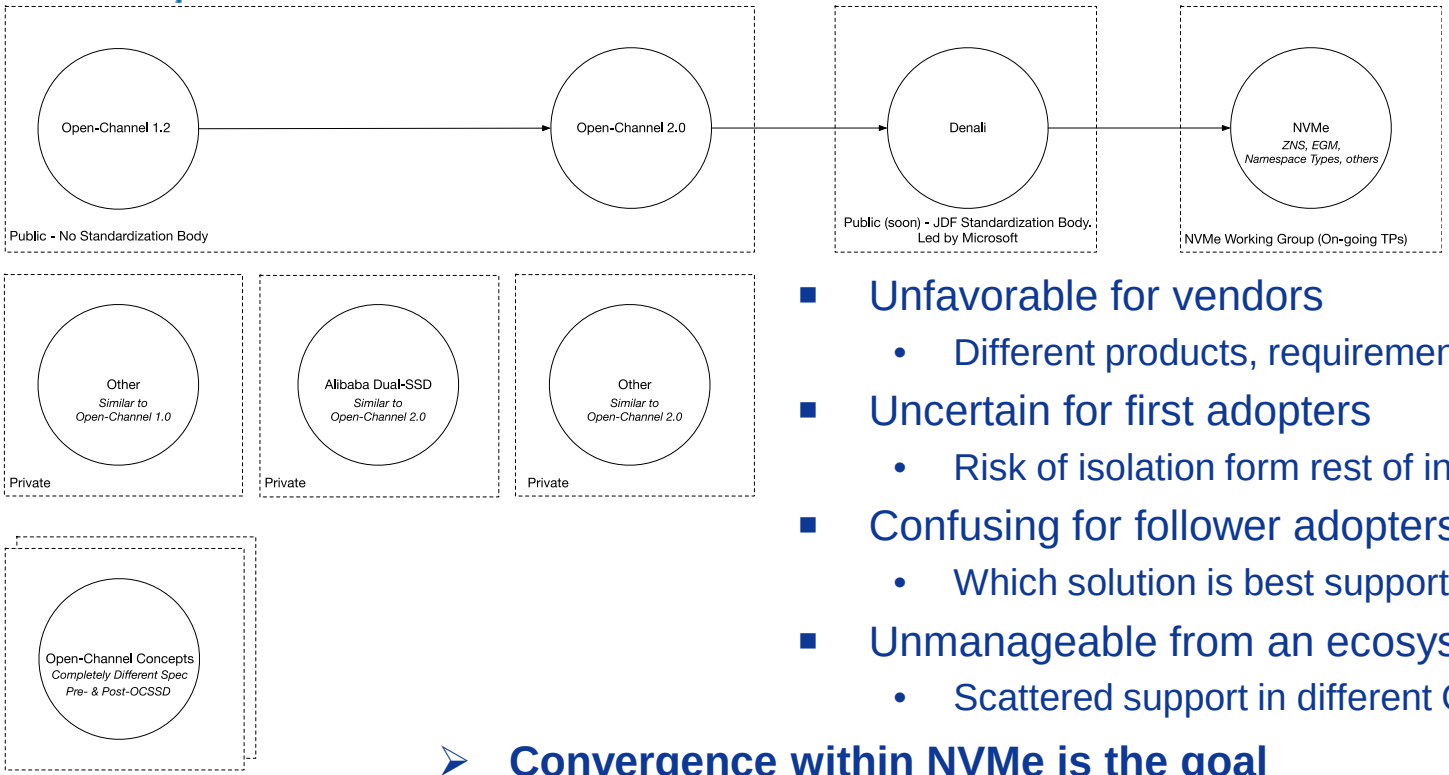
What?

- Facilitate adoption
- Incorporate industry feedback
- Reduce risk for vendor locking

How?

- Public specification development
- Open-source development (LightNVM, pblk, liblightnvm)
- Open-source tools (testing, emulation, management)
- Documentation

Fragmented Adoption



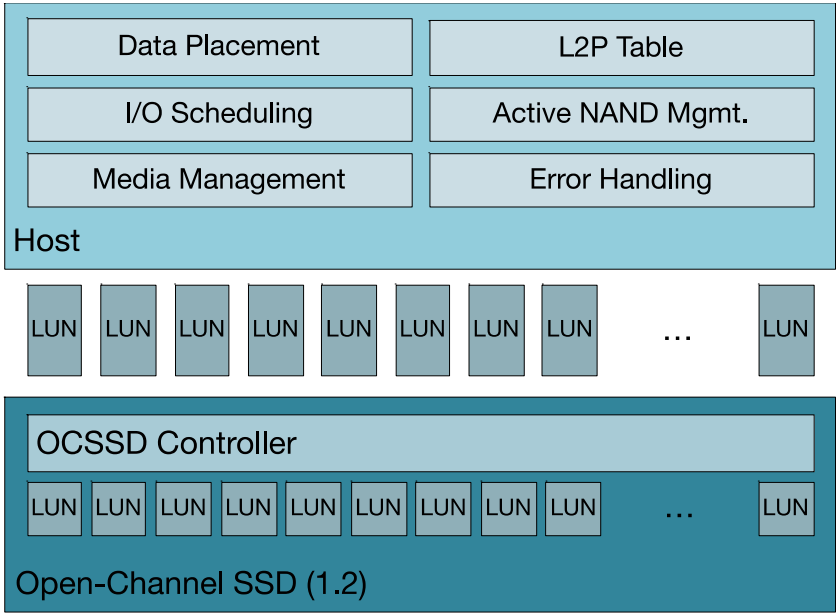
- Unfavorable for vendors
  - Different products, requirements and qualifications
- Uncertain for first adopters
  - Risk of isolation form rest of industry
- Confusing for follower adopters
  - Which solution is best supported?
- Unmanageable from an ecosystem perspective
  - Scattered support in different OS – added complexity

➢ **Convergence within NVMe is the goal**

- Common specification, ecosystem and expectations

Open-Channel SSD 1.2 - Cross-Layer Optimizations

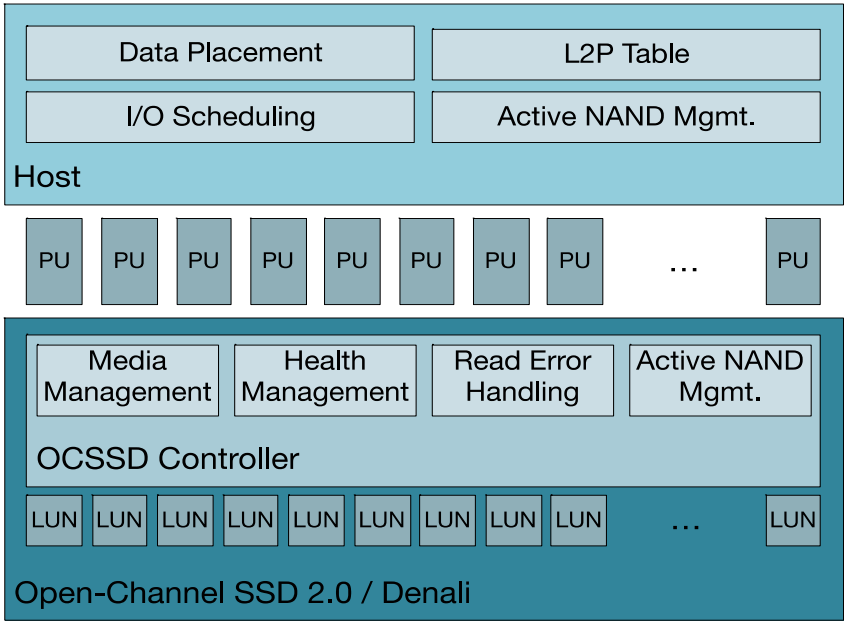
- Map application data structures containing user data to data layout in the SSD
  - Explicitly handle placement, hotness and bandwidth
  - Application-specific data placement
- Implement I/O scheduler based on physical die (LUN) state
  - Control latencies per I/O (noisy neighbor & flatmate)
  - Application-specific schedulers and I/O patterns
- Host-managed NAND
  - Deal with media constraints, care and errors
  - Deal with NAND vendors and generations



➢ ***OCSSD 1.2 enables applications to enforce strict SLAs at the cost of having to (i) build dedicated FTLs for each application, (ii) directly deal with the NAND differences and (iii) use host resources for managing storage while (iv) guaranteeing a sane NAND supply.***

Open-Channel SSD 2.0 / Denali - Commoditization

- Map into logical space (group, PU) instead of a physical space (channel, die)
  - Manage data hotness within the logical space
  - Data placement for classes of applications
- Implement I/O scheduler based on PU state
  - Control media collisions triggered by user I/O
  - Deal with max. tail-latency from by device I/O
- Use “Perfect media”
  - Device deals with media constraints, care and errors
  - Device abstracts NAND vendors and generations
  - AER events for host to deal with device and media states (e.g., *move data due to high ECC*)



➢ ***OCSSD 2.0 / Denali enables the host to control data placement and I/O scheduling while abstracting the media. It trades cross-layer performance for commoditization.***

Project Denali

- Bring industry together to create a specification around Open-Channel concepts
  - Separate specification based on NVMe – goal is to merge in NVMe
  - Focus development for quick adoption
- Main technical contributions
  - Device warranty
    - Dedicated physical access logs in SMART
    - Complete error handling
  - Missing features in Open-Channel public specifications
    - RAID / Parity
    - Integrity / Encryption
    - Random access
    - Reservations
  - Support for newer media

➢ ***Denali addressed this issues within the specification itself, instead of aligning with NVMe concepts. This provided a full solution, but complicated standalone standardization***

Zoned Devices – SMR HDDs

- Shingled Magnetic Recording (SMR) HDDs
  - Sequential-only zones due to physical constrains when increasing density
  - T10 ZBC (SCSI) / T13 ZAC (ATA)
  - Support in existing OSs (e.g., Linux) added in the last years

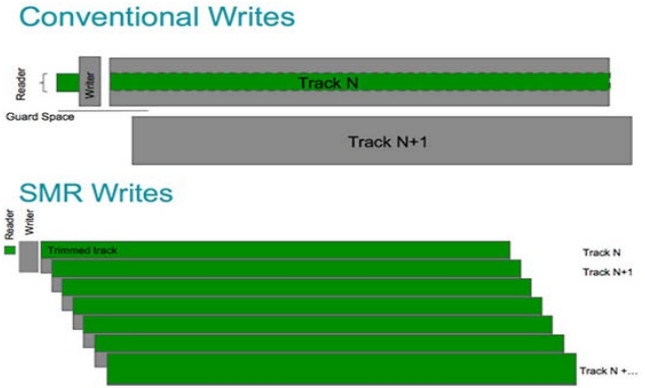
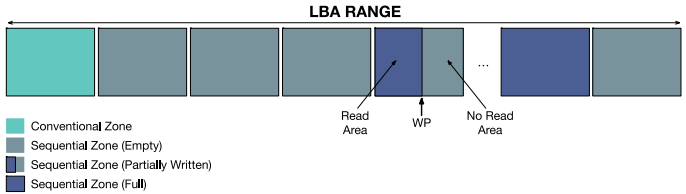
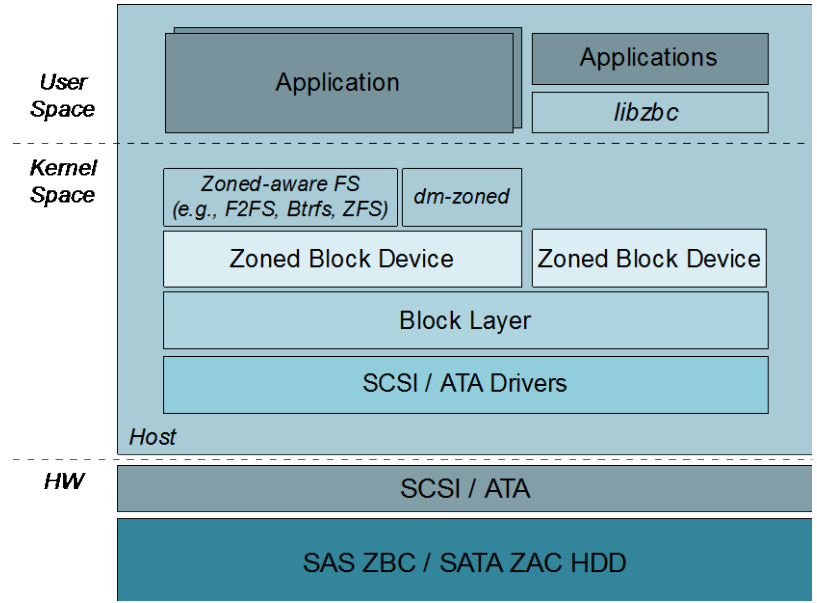
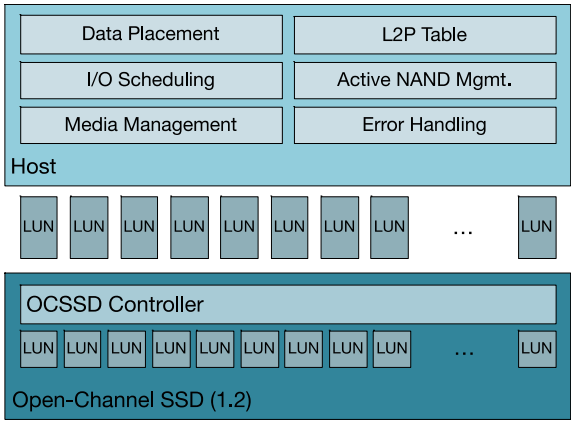


Image Credit: Seagate *(https://www.seagate.com/tech-insights/breaking-areal-density-barriers-with-seagate-smr-master-ti/)*

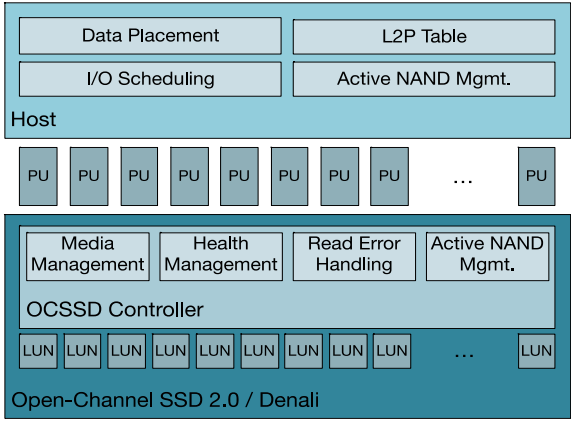




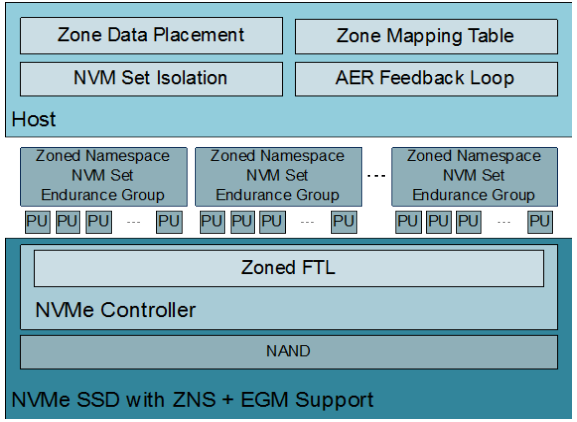
Open-Channel Architecture Evolution



- Full host-based FTL
- Controller for fast I/O path
- Cross-Layer Optimizations
  - Co-design between FTL and applications



- Decoupled FTL
  - Device: Media Management
  - Host: Data Placement & I/O Scheduling
- Controller with more responsibility
- Trade optimizations for commoditization



- Zoned FTL
  - Smaller mapping table
  - Host managed sequentially
  - Good fit for append-only
- Perfect media within zones
  - Device hides media mgmt
- Full commoditization

NVMe Standardization of Open-Channel / Denali Concepts

- Difficult to bring a ”finalized” specification into NVMe for ratification
- Partition Open-Channel / Denali functionality into feature Technical Proposals (TPs)

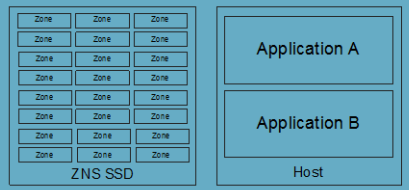
Zoned Namespaces (ZNS)

Benefits

- **Less WAF / Increased life time**: Less extra GC triggered by host LBA updates. Host manages data placement directly in storage stack
- **Reduced tail-latency**: Less device-triggered GC translates into lower chance for unaccounted media collisions
- **Reduced TCO:** Less device DRAM and less need for over-provisioned media

Mechanisms

- Host controls (i) data placement at a zone granularity and (ii) zone state transitions (e.g., open, close, reset)



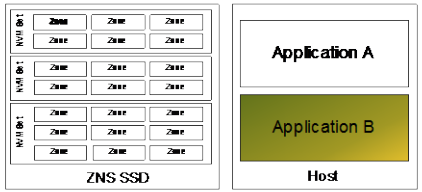
Endurance Group Management

Benefits

- **Multi-tenant I/O Isolation:** Reduce noisy-neighbor problem; i.e., enable the host to reduce media collisions by assigning different Parallel Units (PUs) to different tenants
- **Intra-tenant I/O Isolation:** Reduce noisy-flatmate problem; i.e., enable host to explicitly schedule I/Os from different classes of I/O streams to different PUs

Mechanisms

- Host schedules I/Os to different NVM Sets to provide isolation within and across tenant applications (EG -> NVM Set -> NS)



Other Related TPs

Namespace Types

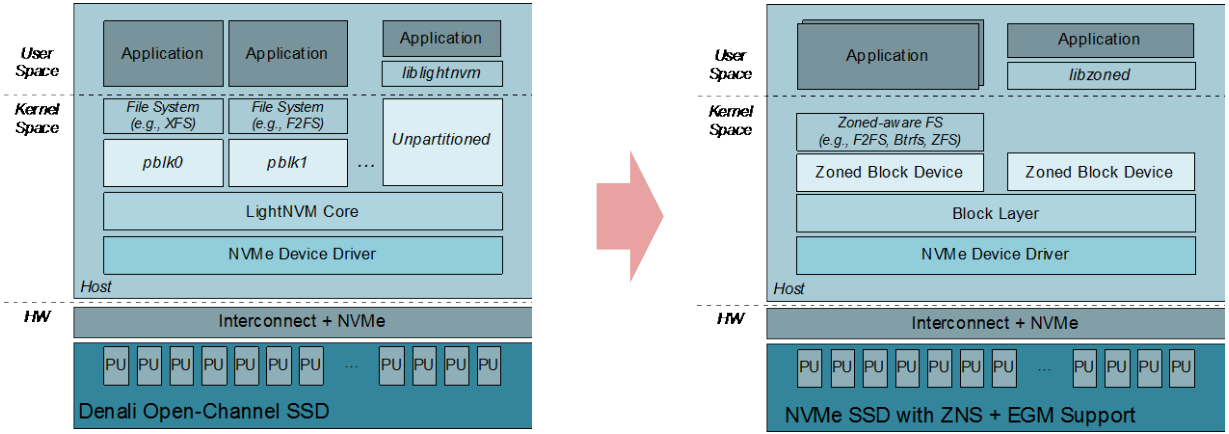
- Mechanism to identify different types of namespaces
- Facilitate namespace type co-existence

Simple Copy Command

- Mechanism to move data internally in the controller between two different LBAs
- Keep the command limited to simple use case

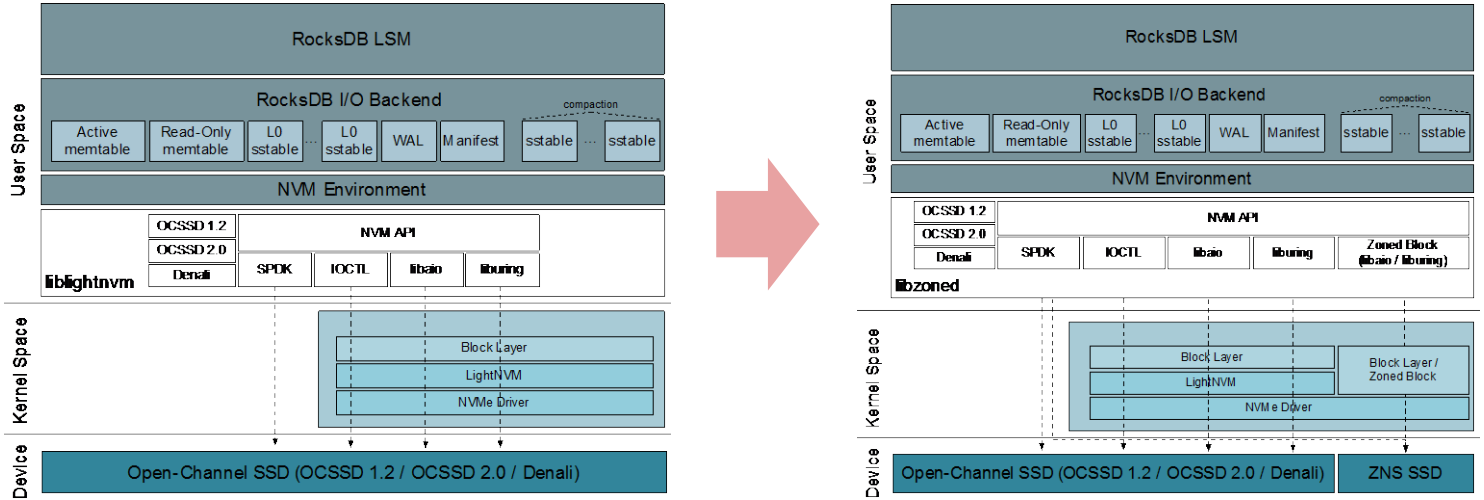
More to come

Ecosystem Transition in Linux



- ZNS leverages more easily the existing ecosystem around NVMe
  - Use all features existing in the block layer and NVMe driver (e.g., multi-queue, hybrid polling)
  - Use existing Zoned Block Device infrastructure built for SMR (ZAC/ZBC) drives
    - Less changes required to natively support log-structured file systems (e.g., F2FS, btrfs, XFS, ZFS on Linux)
  - Expect more contributions than in Open-Channel ecosystem (i.e., LightNVM)
- Enable ZNS support for applications with Open-Chanel backends (e.g., RocksDB)
  - Maintain *liblightnvm* NVM API in *libzoned*

Example: RocksDB OCSSD to ZNS



- OCSSD RocksDB already implements zone abstractions
  - NVM Environment matured after several re-implementations
  - Support for NVM API, which supports other application integrations
  - liblightnvm -> libzoned
    - Generic user-space library for non-block devices with read/write/reset requirements
    - Focus on evolving zoned block framework

Denali Impact & Future

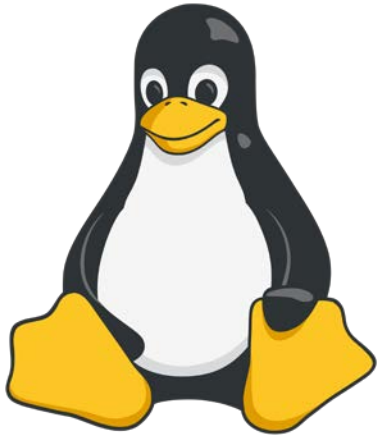
- Denali brought the industry together to standardize Open-Channel concepts
  - First step towards standardization, another step towards commoditization
  - Effort lead by Microsoft and CNEX Labs
  - Representation from vendors, OEMs and users
- Denali succeeded at providing a complete specification
  - Continuation of OCSSD 2.0
  - Applied industry feedback
  - Targeted requirements from cloud and enterprise users
- Denali struggled at standardization within NVMe
  - Difficult to provide a “complete” specification for ratification
  - Unilateral standardization with NVMe dependencies is not easy to adopt
- ➢ NVMe standardization is the next natural step to commoditization
  - Introduce functionality progressively in the form of independent technical proposals (TPs)
  - Align with NVMe concepts that are ratified and deployed
  - Leverage existing ecosystems for easier adoption

Conclusion

- Open-Channel SSDs work at best on full-stack integrations
  - Higher benefits when cross-layer optimizations are possible (OCSSD 1.2)
  - Need to deal with media management and supply
- Open-Channel SSDs have evolved towards commoditization
  - Denali / OCSSD 2.0 trades optimizations for commoditization of the media
  - Strong Linux ecosystem has enabled this evolution through research & experimentation
- Open-Channel fragmentation is a burden for wide adoption
  - Challenges for vendors, users and developers
  - Only Tier 1 customers can afford deployment
  - The lack of a standard challenges the benefits
- The industry efforts in Denali are now put into different TPs in NVMe
  - Zoned Namespaces (ZNS) and Endurance Group Management
- Next tangible steps:
  - TP Ratification
  - ZNS support in Linux & application enablement e.g., RocksDB)

Call for action

- Contributions to open-source ZNS ecosystem
  - Linux kernel support for ZNS in zoned block framework
  - User-space libraries and application support
  - QEMU support
  - Test tools
- Contributions to ZNS standardization in NVMe
  - Co-sponsoring current TPs
  - Working on future TPs
- Talk to us



The Denali Open-Channel Standard: Impact and Future

Javier González

Principal Software Engineer

SSDR R&D Center Leader