CNEXLABS

Denali Open-Channel SSDs

Flash Memory Summit 2018 – Architecture Track

Javier González

Open-Channel SSDs

- **Definition:** *A class of Solid State Drives that expose (some of) their geometry to the host and allow it to control (part of) its internals through a richer R/W/E interface.*
  - The design space is very large, from pure physical access to restricted access rules.
- **Objective:** Move storage closer to the application – host data placement & scheduling
  - Reduce Write Amplification Factor (WAF) *and* Space Amplification Factor (SAF)
  - Make media parallelism available to the host – device provides isolation across parallel units

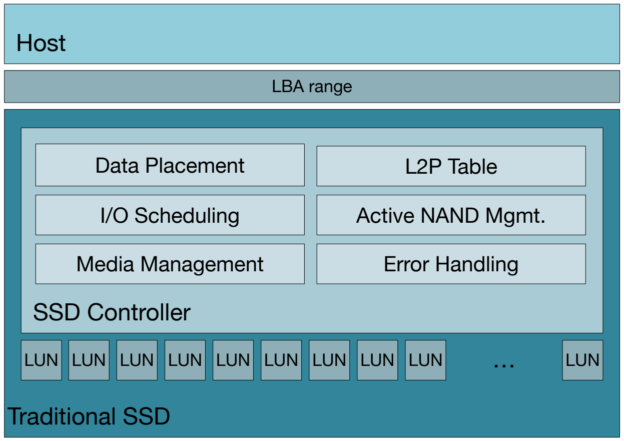
- Term introduced at *Baidu paper* [1] to optimize their KV-store by accessing physical media.
- Different types of OCSSDs implemented in the industry: Fusion I/O, Violin Memory and others.
- Several specifications available:
  - Open-Channel SSD 1.2 and 2.0 [2] (Support in Linux Kernel through LightNVM since 4.4 and 4.17)
  - Alibaba's Dual-Mode SSD [3]

[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

Denali Open-Channel SSDs

- Industry standard for a *specific* point in the Open-Channel SSD design space
  - **Goal:** Minimize host & device changes across NAND generations to speed up new media adoption
  - Device manages the physical media: retention, ECC, access policies, program patterns, etc.
  - Host controls data placement and I/O scheduling
  - Media access is abstracted through access rules – align with zoned devices
  - Feedback loop for device -> host communication
- Several companies involved in the Joint Development Forum (JDF). Representation from:
  - NAND, controller and SSD vendors
  - Cloud Service Providers (CSPs)
  - Enterprise storage
- Start point at Open-Channel 2.0 specification
  - Continue work done from 1.2 to 2.0
  - Address issues / limitations in 2.0
  - Incorporate feedback from the JDF for industry-wide standard
  - Pave the way for incorporating in NVMe standard

Design Space



Host

LBA range

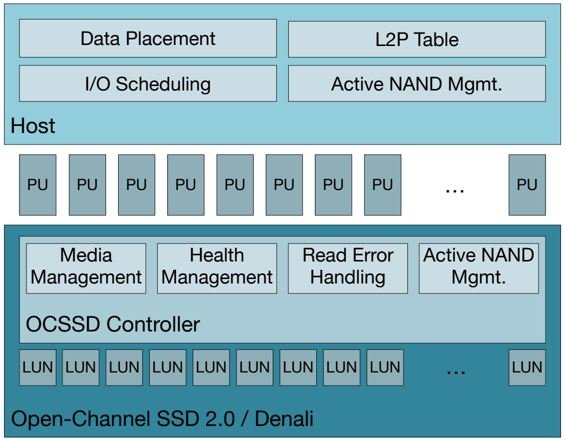
Data Placement
L2P Table
I/O Scheduling
Active NAND Mgmt.
Media Management
Error Handling

SSD Controller

LUN LUN LUN LUN LUN LUN LUN LUN LUN … LUN

Traditional SSD

- Mimic HDDs and maintain block abstraction for adoption
- HDDs already moving towards zone-based block devices (SMR)



Data Placement
L2P Table
I/O Scheduling
Active NAND Mgmt.

Host

PU PU PU PU PU PU PU PU … PU

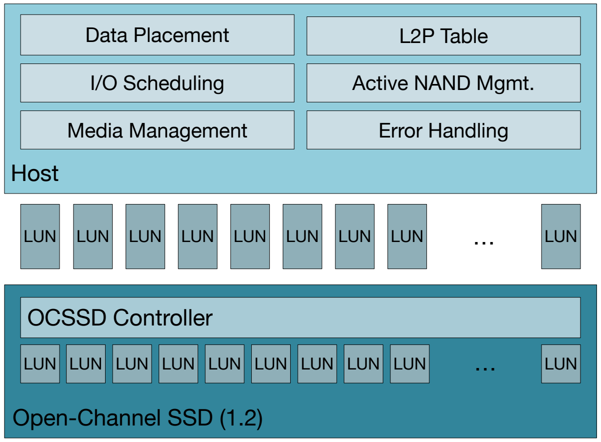
Media Management
Health Management
Read Error Handling
Active NAND Mgmt.

OCSSD Controller

LUN LUN LUN LUN LUN LUN LUN LUN LUN … LUN

Open-Channel SSD 2.0 / Denali

- Simplify device by removing mapping layer – maintain media
- Remove block abstraction to enable host place and schedule
- Host rules similar to zone-devices



Data Placement
L2P Table
I/O Scheduling
Active NAND Mgmt.
Media Management
Error Handling

Host

LUN LUN LUN LUN LUN LUN LUN LUN … LUN

OCSSD Controller

LUN LUN LUN LUN LUN LUN LUN LUN LUN … LUN

Open-Channel SSD (1.2)

- Very simple device
- Ideal for fast prototyping
- Productize single NAND / vendor SSD

SSD Design Space (increasing host responsibilities)

CNEXLABS

Nomenclature

- **Geometry**:
  - *Logical Block:* Minimum addressable unit for reads.
  - *Chunk:* Collection of logical blocks. Minimum addressable unit for resets (erases)
  - *Parallel Unit (PU):* Collection of Chunks that must be accessed serialized. Typically, a device counts on tens / hundreds of these parallel units
  - *Group:* Collection of PUs that share the same transfer bus on the device.
- **Tenant**: Logical construction to represent a collection of chunks that are used by the host for a given purpose. This abstraction allows to manage WAF, SAF, isolation and parallelism
  - A device can be partitioned into several workload-optimized tenants.
  - When a tenant is composed by a whole PU, the tenant is guaranteed I/O isolation – aggregated bandwidth does not have an impact on per-tenant latencies.
  - When none of the tenants share chunks, very little WAF is generated - applications good at managing sequential writes and hot / cold data separation.
  - Per-tenant over-provision allows to minimize SAF – better $$ per TB.

Key Challenges*

* The Denali spec. is not finalized

Device Warranty

- Without any form of protection, the host can create hot spots and wear them out very quickly
- Current warranty metrics do not apply since the host manages data placement
  - DWPD and TBW are device-wide
  - Devices are optimized for a given workload – different over-provision, DWPD, etc.
  - Current reliability / endurance standards are device wide (e.g., JEDEC)

- **Path**: Host is king, but it must behave!
  - Reuse today's warranty concepts, but at a lower granularity (chunk)
  - Add protection mechanisms for misbehaved hosts (e.g., prevent double erases*)
  - Report wear unevenness (over given threshold) to the host through feedback loop*
  - Maintain a dedicated log for warranty purposes. Vendor ground truth
  - Incorporate Denali to current reliability / endurance standards
  - Create tools to verify host's and device's behavior

** Present in Open-Channel 2.0*

Reservations

- The host can organize parallel units (PUs) into different tenants to provide I/O isolation
- Tenants are not visible to the device
  - Feedback loop can only report wear unevenness at a chunk granularity
  - Uneven wear across tenants is expected
  - Device-side per-tenant manipulation is not possible (e.g., RAID)

- **Path:** Incorporate reservations
  - Allow host to create / delete tenants that are visible to the device
    - Host is aware of media boundaries: Group, PU and chunk
  - Enable per-tenant feedback notifications
    - Different urgency and scope
  - Narrow down device responsibilities for different types of tenants
  - Support for *vertical, horizontal* and *hybrid* configurations: manage isolation vs. capacity
  - NVMe alignment: Endurance groups & NVM Sets

Device side RAID

- RAID is strongly recommended for some NAND by the vendors, independently of used ECC
  - Prevent *infant block death*
  - Reach acceptable PE Cycle count
- RAID has purposely being left out of available specifications due to complexity
- Different needs
  - CSPs deals with single device errors at higher level: erasure coding, software RAID, etc.
  - Enterprise require per device guarantees: RAID within drives and across drives

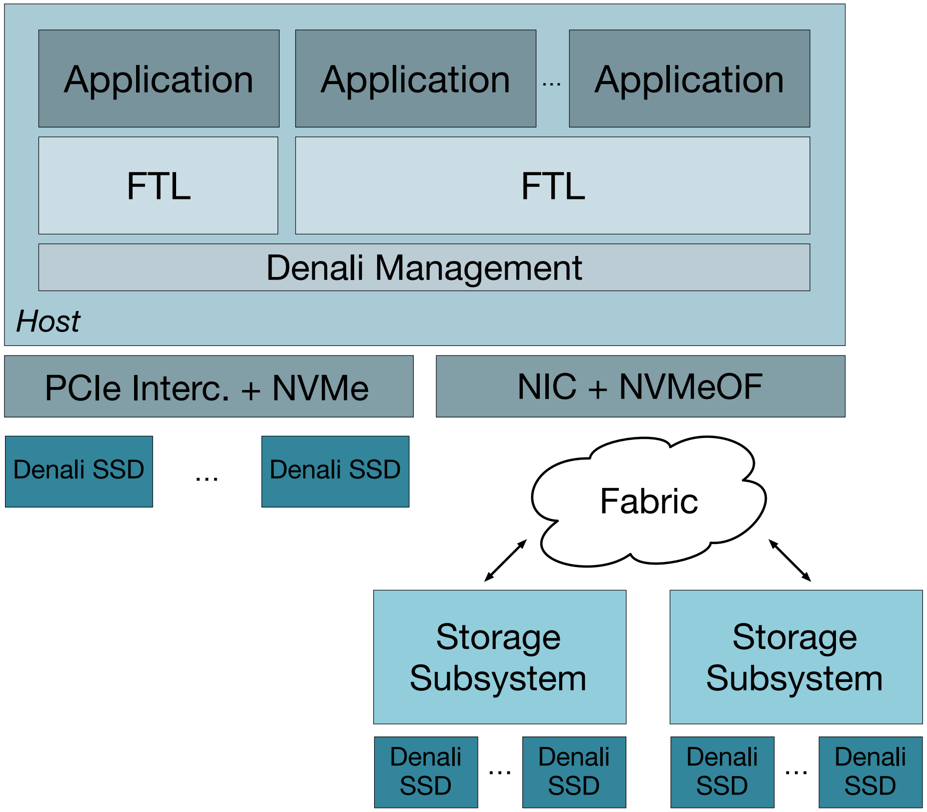
- **Path:** Support RAID as part of reservation interface
  - Denali should address cloud and enterprise; leaving RAID undefined (i.e., VU) will create fragmentation
  - RAID schemes should be reported by the device – flexibility and controlled complexity
  - Enabled in Denali as part of the reservations
    - *create_tenant* takes PUs and raid scheme as parameters
  - User-specific parity schemes can be accelerated other places (in a few slides)

NVMe Alignment

- The ultimate purpose is to standardize the core ideas of Denali OC in NVMe
  - It is necessary for wide industry adoption
  - Helps reducing fragmentation
  - Makes further integration in the storage stack easier
  - Changes are expected

Architecture

Architecture – Host Management



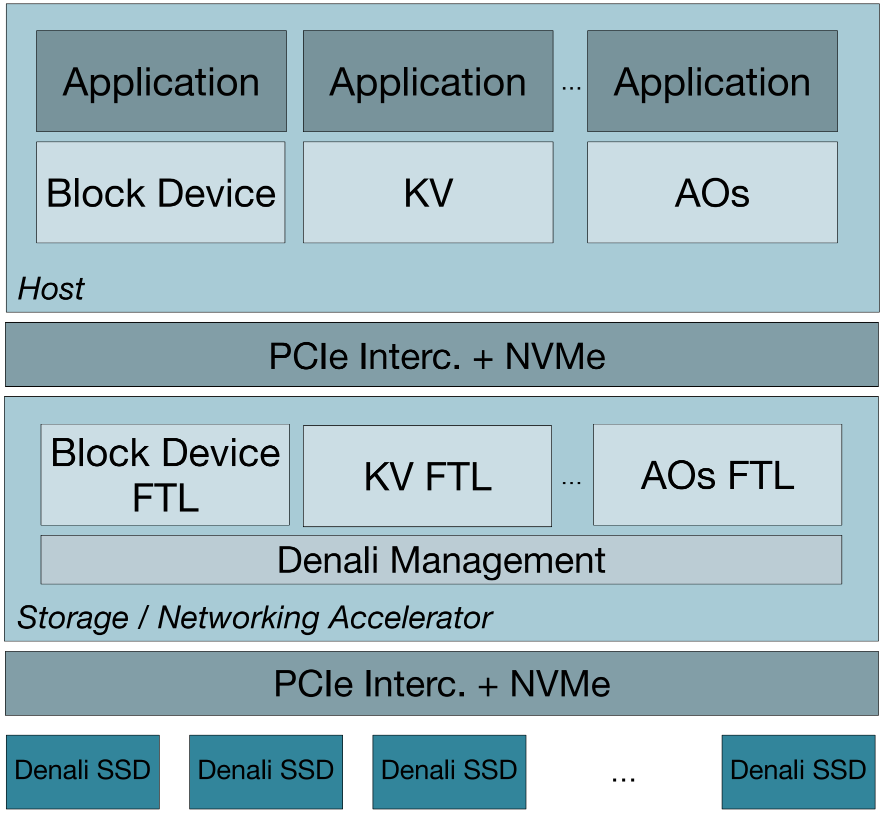
- Host manages storage entirely
  - Use of CPU and memory
  - Need for dedicated drivers *
- Devices expose Denali interface
- Need for Denali-OF for disaggregation

** Support in Linux through LightNVM subsystem (from kernel 4.4)*

Adoption Challenges & Insights

- Service providers are not willing to dedicate extra host resources for managing storage
  - Resources are *billable* to the application
  - FTLs are expensive in terms of both CPU and memory
  - ➢ **Denali must enable both (i) lightweight host and (ii) FTL offload**
- Most applications can excel with simpler abstractions than *raw* media access
  - Error path is the critical path – raw media does fail
  - ➢ **Denali already abstracts media. More complex constructions can be built on top**
- Considerable gains by moving computation and transfers closer to the data while maintaining a simple interface at the host (i.e., computational storage)
  - RAID, erasure coding, deduplication, compression, etc.
  - Analytics, calculations, DB operations, etc.
  - Peer-to-peer PCIe, RDMA
  - ➢ **Denali is a flexible storage offload for different classes of applications**
- Disaggregation must be simple
  - Management + transparent I/O path
  - ➢ **Denali can build on top of work done on NVMeOF**

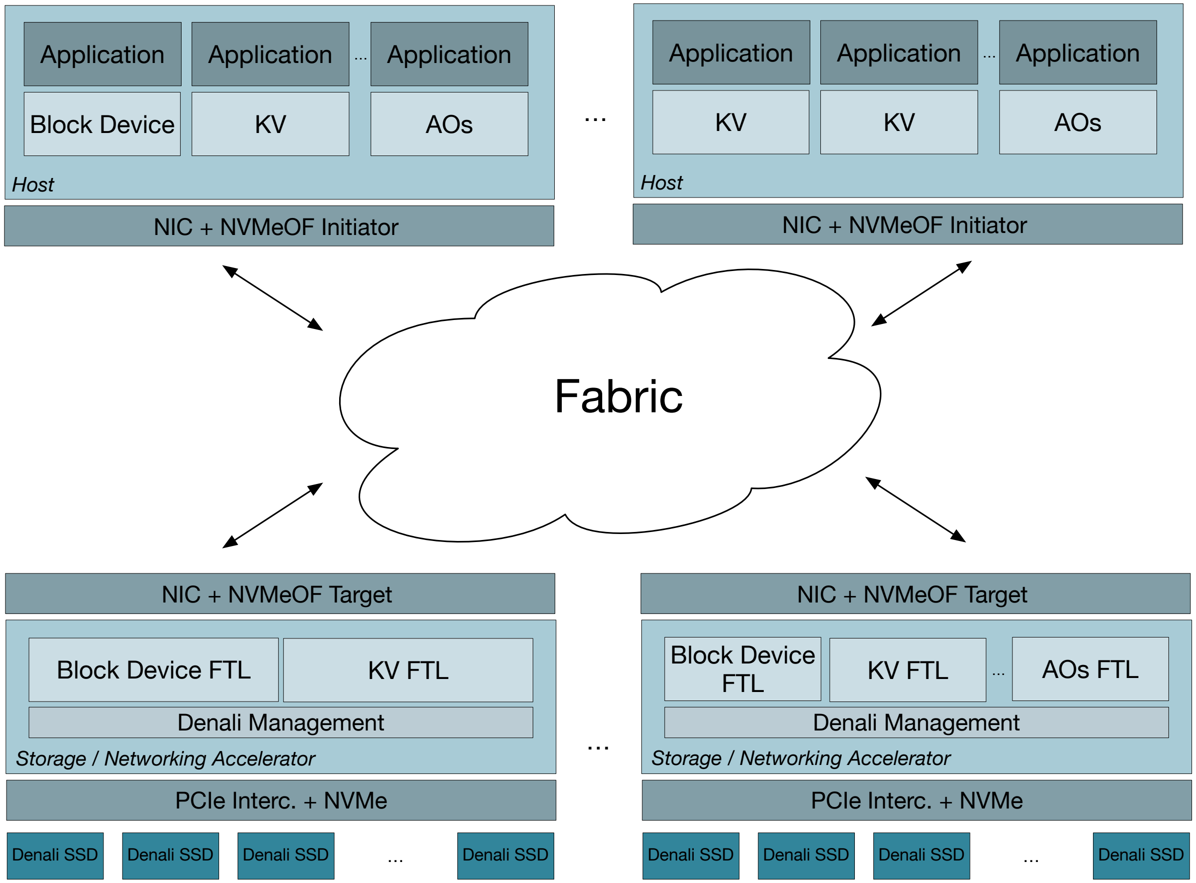
Architecture – Offloaded Management



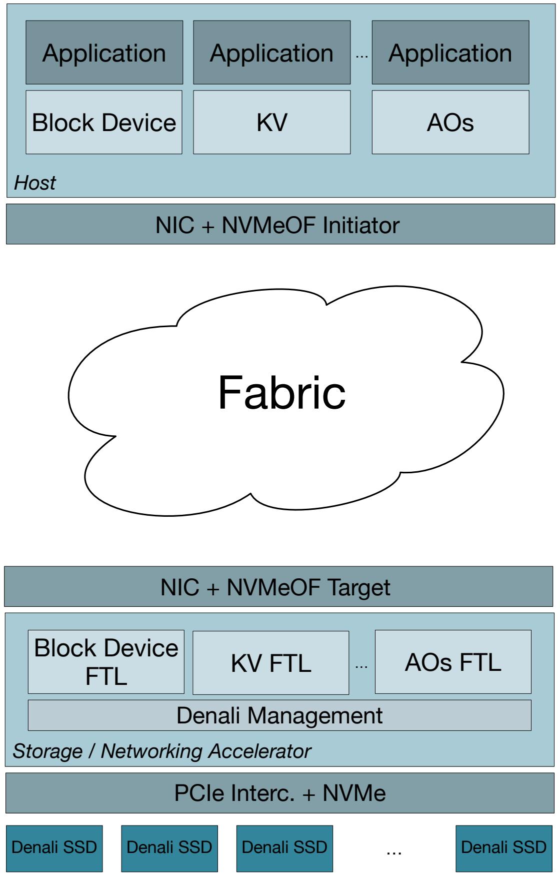
- Host uses a lightweight API
  - Traditional block devices
  - Key-Value stores
  - Dedicated APIs, e.g., append-only streams (AOs)
- Aggregator manages storage
  - *Billable* CPU and memory not shared with the application
  - No need for Denali OC support in host's OS
  - Allows to offload computation closer to the data on parts owned by the service provider
  - Vendor Unique commands allow to use NVMe
- PCIe attached allows fast adoption for dedicated storage APIs on local storage
- Fabrics attached enables typical NVMeOF disaggregation (next slide)

CNEXLABS

Architecture – Offloaded Management - OF

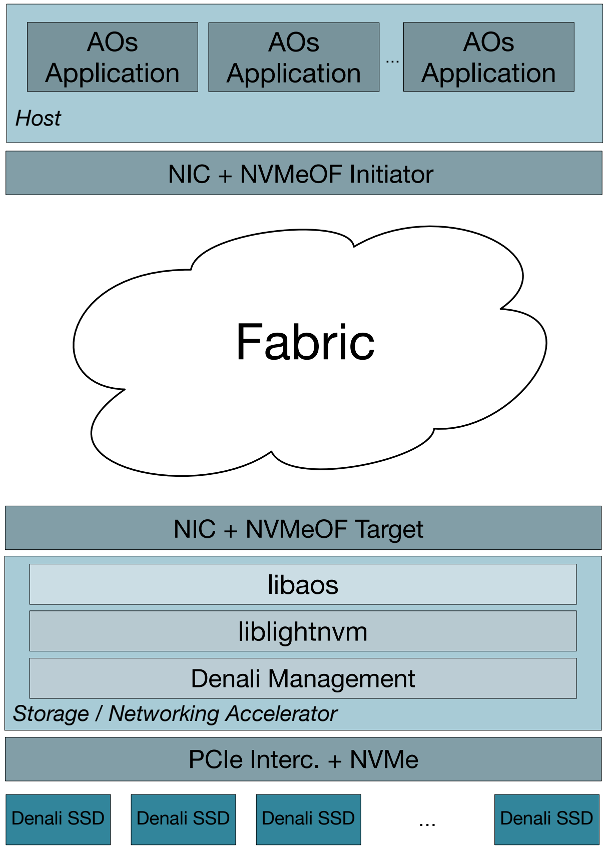


Back to Software APIs



- Storage protocols increase in complexity by adding vendor-, user-dedicated extensions
  - Solve a problem under a set of assumptions
  - Require dedicated support from SSD vendors
  - Once standardized, need to carry it forever
  - Once deployed, it cannot be *easily* modified
- Denali OC allows for these extensions to become APIs
  - Easy to prototype, test, deploy and modify
  - Much shorter deployment time (years to weeks)
  - Still allow for multiple sources
  - NVMe can be used as protocol through VU commands (which remain between host and accelerator)
  - Storage accelerators simplify this further by encapsulating Denali OC-specific management, leaving host untouched
    - Offloads, HW accelerations, user-specific secret sauce, etc.
- Denali **is not** the *ultimate* spec. of specs., but it can help to reduce workload-specific extensions in NVMe

Case Study: Append-Only Streams (Demo)



- Append-Only Streams
  - Spec. developed by Microsoft to (i) minimize WAF, (ii) reduce device DRAM and (iii) speed up denser NAND adoption.
  - Extend the concept of directives (i.e., *I/O streams)*
    - Write sequential following rules, read random
    - Minimize data movement during GC
    - Management interface: per-stream properties (e.g., isolation)
- *libaos*
  - Built entirely on top of *liblightnvm* (< 1500 LOC)
  - Run transparently on OCSSD 1.2, 2.0 and Denali
- Accelerator
  - Hides *libaos* implementation details
  - Does not require SSDs to support AOs natively
  - AOs spec. can change and *libaos* can be improved through software update
    - No need to replace any piece of HW
    - Avoid new SKUs and drive re-qualification

CNEXLABS Microsoft

Conclusions

- Different types of Open-Channel SSDs are gaining significant traction
  - Increasing demand both by CSPs and enterprise
  - Different flavors available
  - Different vendors with SDKs and prototypes
  - ➢ Denali JDF is a step towards reducing fragmentation – NVMe is the goal
- Accelerator platforms for storage offload are a good fit for Denali OC adoption
  - Do not spend host resources managing storage
  - Facilitate offloads and provide them with a richer storage interface
  - Leverage existing ecosystem
  - Help to keep a lightweight NVMe protocol – promote the use of software APIs
- Growing ecosystem with more companies contributing
  - CNEX Labs, Western Digital, Intel, Red Hat and others
  - lightnvm.io: documentation, research articles, github repositories and examples

CNEX Labs, Inc.

Visit us at the CNEX Labs room. Request demo: *info@cnexlabs.com*

Denali:

- Denali-OF Tenant Isolation with Broadcom’s Stingray
- Denali-OF Tenant Isolation with Mellanox’s Bluefield
- Append-Only streams library (*libaos)* on Denali using *liblightnvm*