SPDK, NVME-OF Acceleration

Sasha Kotchubievsky, Oren Duer| Mellanox Technologies

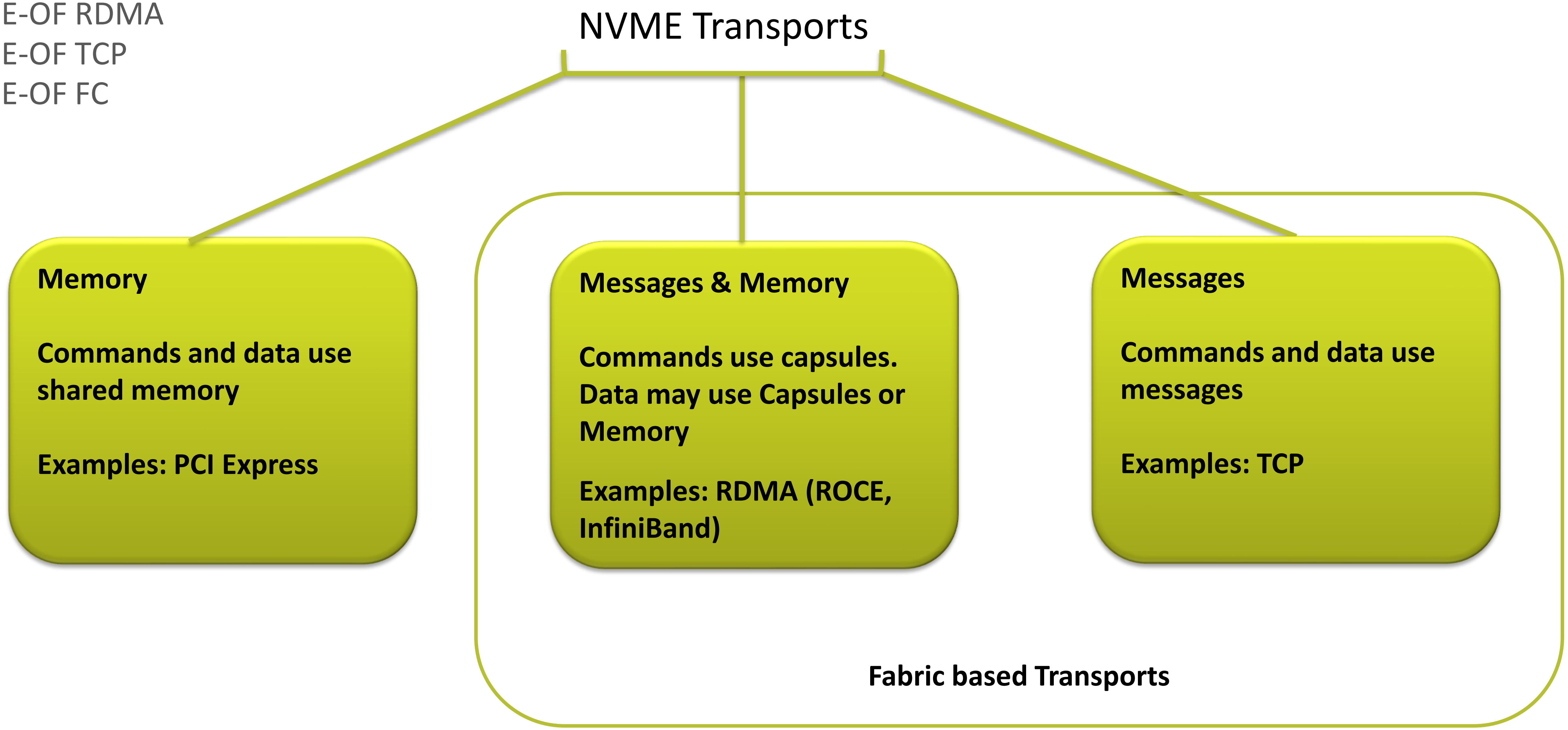
September, 2019

Agenda

- Background
- Low-level optimizations in NVME-OF RDMA transport
- Data protection in RDMA transport
- Advanced hardware accelerations in network layer

NVME and NVME-OF

- **NVMe is designed to work over a PCIe bus**
- The **NVMe over Fabrics** is the protocol used for transferring NVMe storage commands between the client nodes over storage fabric
  - NVME-OF RDMA
  - NVME-OF TCP
  - NVME-OF FC



NVME Transports

Memory

Commands and data use shared memory

Examples: PCI Express

Messages & Memory

Commands use capsules. Data may use Capsules or Memory

Examples: RDMA (ROCE, InfiniBand)

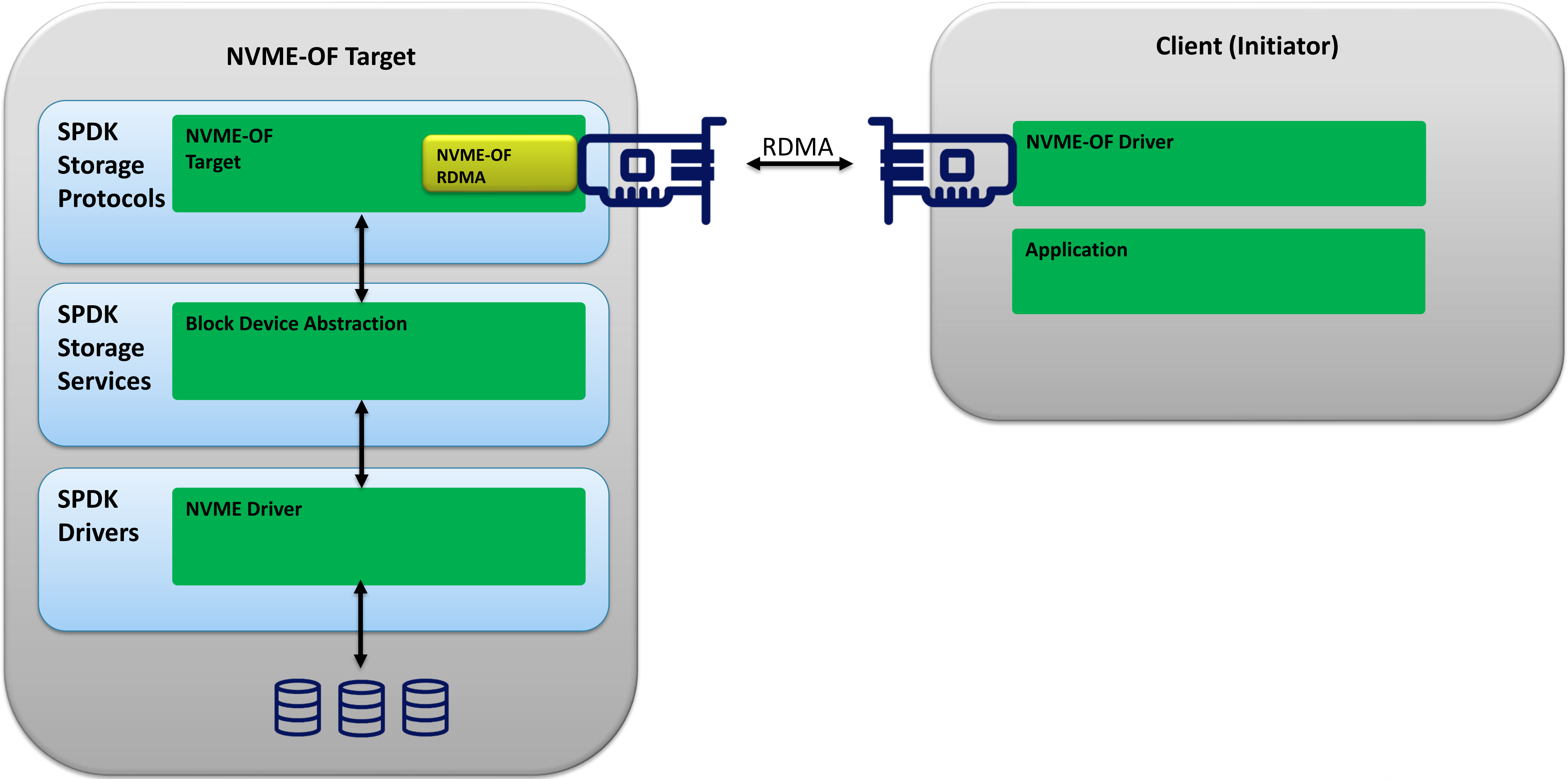
Messages

Commands and data use messages

Examples: TCP

Fabric based Transports

SPDK. NVME-OF Abstraction



NVME-OF RDMA Optimizations

NVME-OF RDMA. Performance optimizations

- Scope
  - NVME-OF Target on x86
  - NVME-OF Target on ARM
  - NVME-OF Target forwards IO to backend target
- Network cards
  - “ConnectX-5”
  - “BlueField”

RDMA. Selective signaling

- “Selective signaling” reduces PCIe bandwidth and CPU usage by eliminating DMA completion
- In IO Read flow, RDMA_WRITE is followed by RDMA_SEND
  - Completion for RDMA_WRITE can be skipped
- Developed by Alexey Marchuk, Mellanox: https://review.gerrithub.io/c/spdk/spdk/+/456469
  - Available in SPDK v19.07
- “Selective signaling” increases IOPs in “randread”
  - ARM up to 15%

RDMA. Work request batching

- “Work request batching” reduces CPU use and PCIe bandwidth by using single MMIO operation (“Doorbell”) for multiple requests
- The default approach for WQE (work request element) transferring requires separate MMIO for each WQE
- WQE batching improve:
  - IO Read flow: RDMA_WRITE is followed by RDMA_SEND
  - “Heavy” loads (high queue depth): NVME-OF Target needs to submit multiple RDMA operations
  - Multi –element SGL: Each element needs own RDMA operation
- Developed by:
  - Seth Howell, “Intel”: https://review.gerrithub.io/c/spdk/spdk/+/449265 - NVME-OF Target
    - Available in SPDK v19.07 . Requires applying fix : https://review.gerrithub.io/c/spdk/spdk/+/466029
  - Evgenene Kotchetov, “Mellanox” : https://review.gerrithub.io/c/spdk/spdk/+/462585/ - NVME-OF Initiator
- Preliminary results:
  - ARM: randread (queue depth 64) up to 5%, randwrite (queue depth 64) up to 12% increase in IOPs

RDMA. Work request’s payload inlining

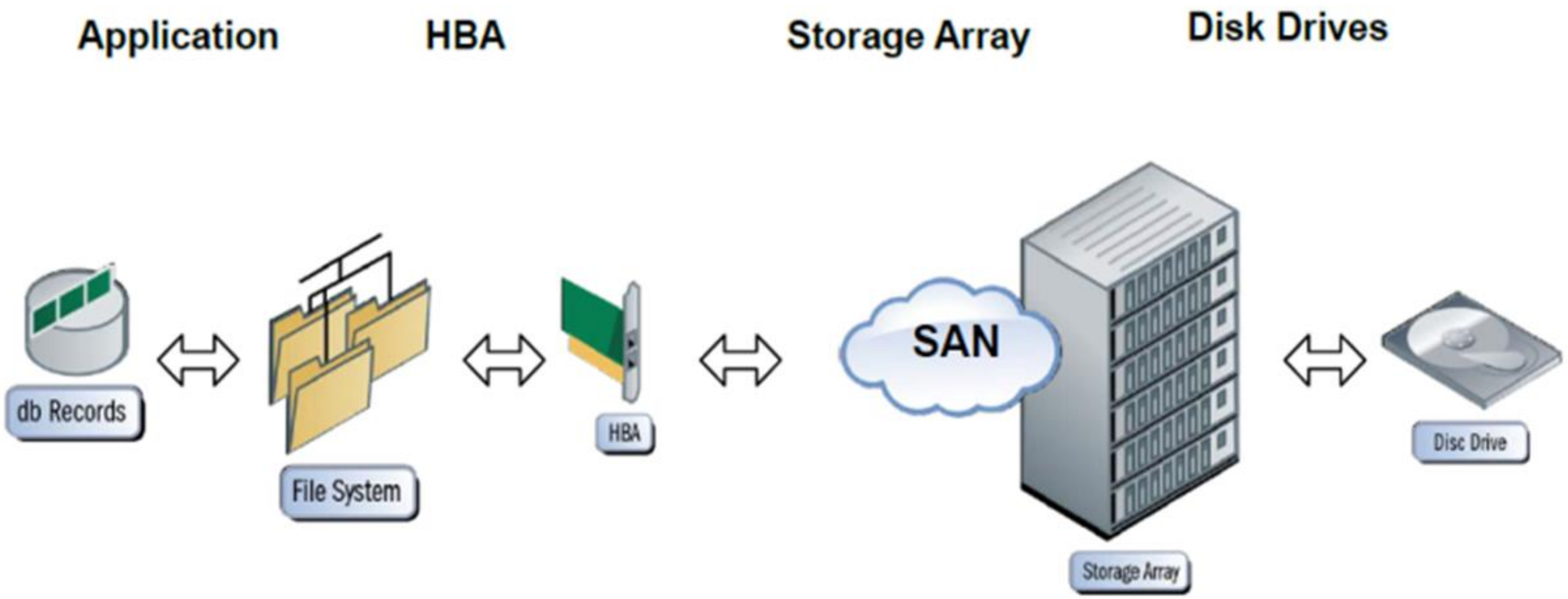
- Payload inlining reduces PCIe bandwidth by eliminating DMA read for payload
- Small payloads up to a few hundred of bytes can be encapsulated into WQE
- Payload inlining can be used for NVME-OF response
  - Capsule size is 16 bytes
  - The feature is under development (Alexey Marchuk, “Mellanox”): https://github.com/Mellanox/spdk/commit/8682d067e5ab9470fb3596db0c47411c974ac47f

NVME-OF RDMA Data protection

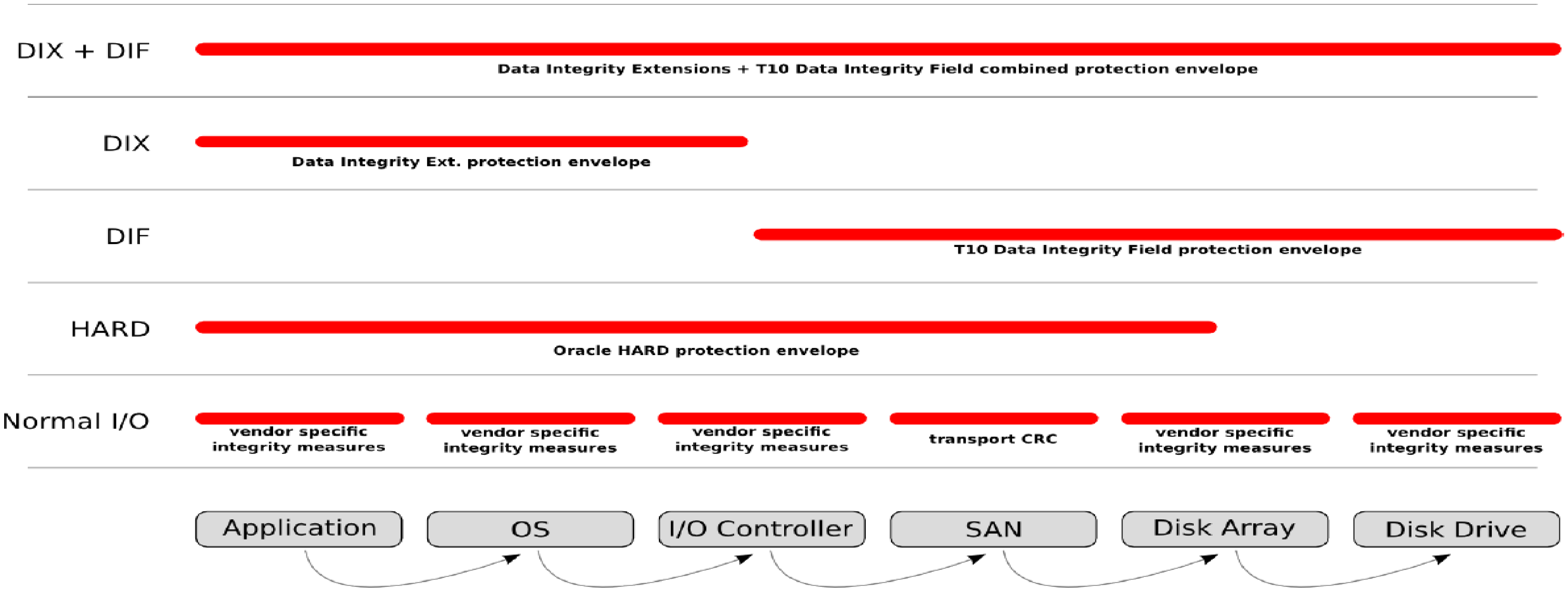
Data Corruption

- IS A DISSASTER !!
- Backups may have bad data
- Downtime/Corruptions may be fatal to a company
- It is better to Not Return any data, than return a wrong one
- Occur as a result of bugs, both SW and HW (drivers, HBAs, Disks, Arrays)
- Common failures:
  - Write incorrect data to the storage device – may take months for recognition
  - Misdirected writes

Error can happen in every entity in the IO path:



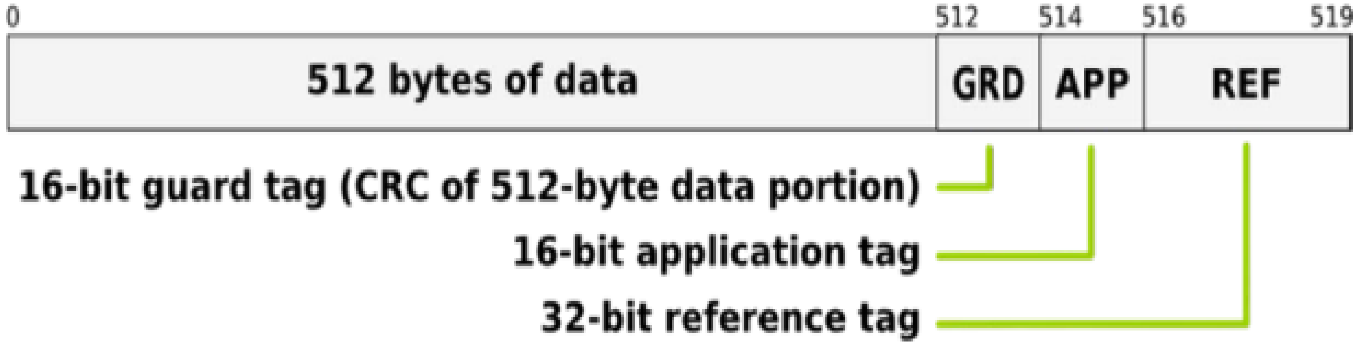
I/O path entities



DIX + DIF
Data Integrity Extensions + T10 Data Integrity Field combined protection envelope
DIX
Data Integrity Ext. protection envelope
DIF
T10 Data Integrity Field protection envelope
HARD
Oracle HARD protection envelope
Normal I/O
vendor specific integrity measures
vendor specific integrity measures
vendor specific integrity measures
transport CRC
vendor specific integrity measures
vendor specific integrity measures
Application
OS
I/O Controller
SAN
Disk Array
Disk Drive

*based on Martin K. Peterson slide

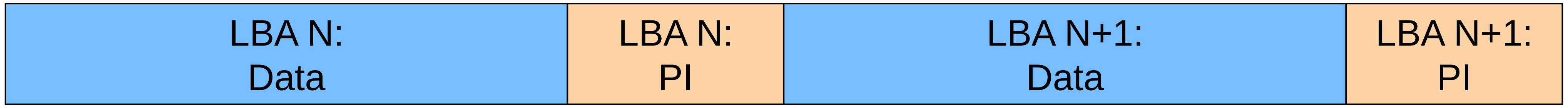
Model



- 8 byte of integrity tuple per sector
- Guard tag:
  - Per request property
  - Protects the data portion of the sector
  - On the Wire – CRC using well-defined polynomial
  - OS – usually use cheaper IP checksum algorithm (may use CRC)
  - I/O controller should convert between types, if needed
- Application tag:
  - Opaque
  - Free usage by application
- Reference tag:
  - Protect against misdirected writes
  - Type 1 - 32 LSbits of the LBA are used as base tag and incremented with each segment
  - Type 2 - 32 LSbits of the LBA used as base tag, can be anything for the rest
  - Type 3 – Only Guard tag is checked

NVMEoF – Metadata Handling

- Two possibilities for MetaData layout
  - Interleaved: Each data block is appended with 8byte integrity payload.
  - Not supported by Linux for local devices



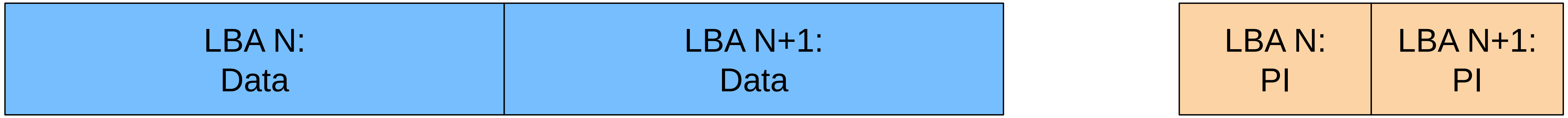
LBA N: Data

LBA N: PI

LBA N+1: Data

LBA N+1: PI

  - Separate: Integrity payload fields lie in a separate buffer from the data.
    - Not supported in Fabrics by definition of the spec (not enough space in the SQE for metadata pointer)



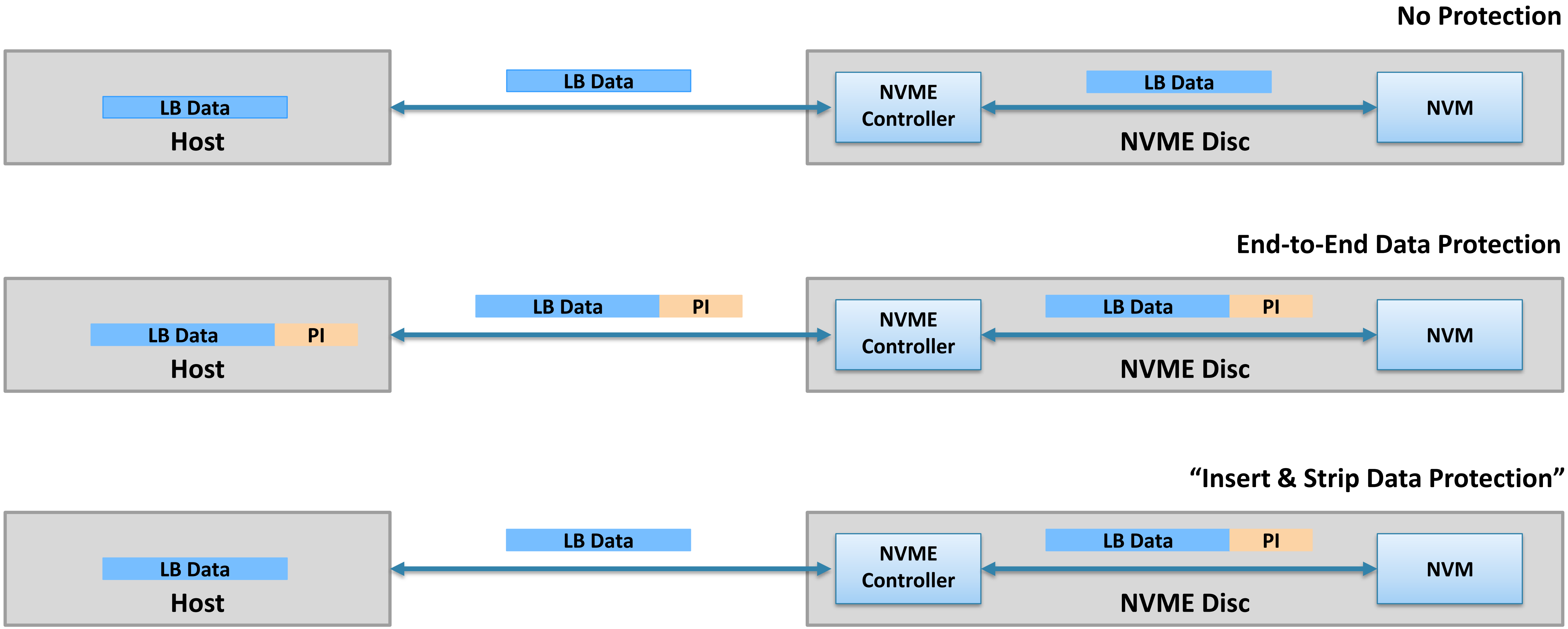
LBA N: Data

LBA N+1: Data

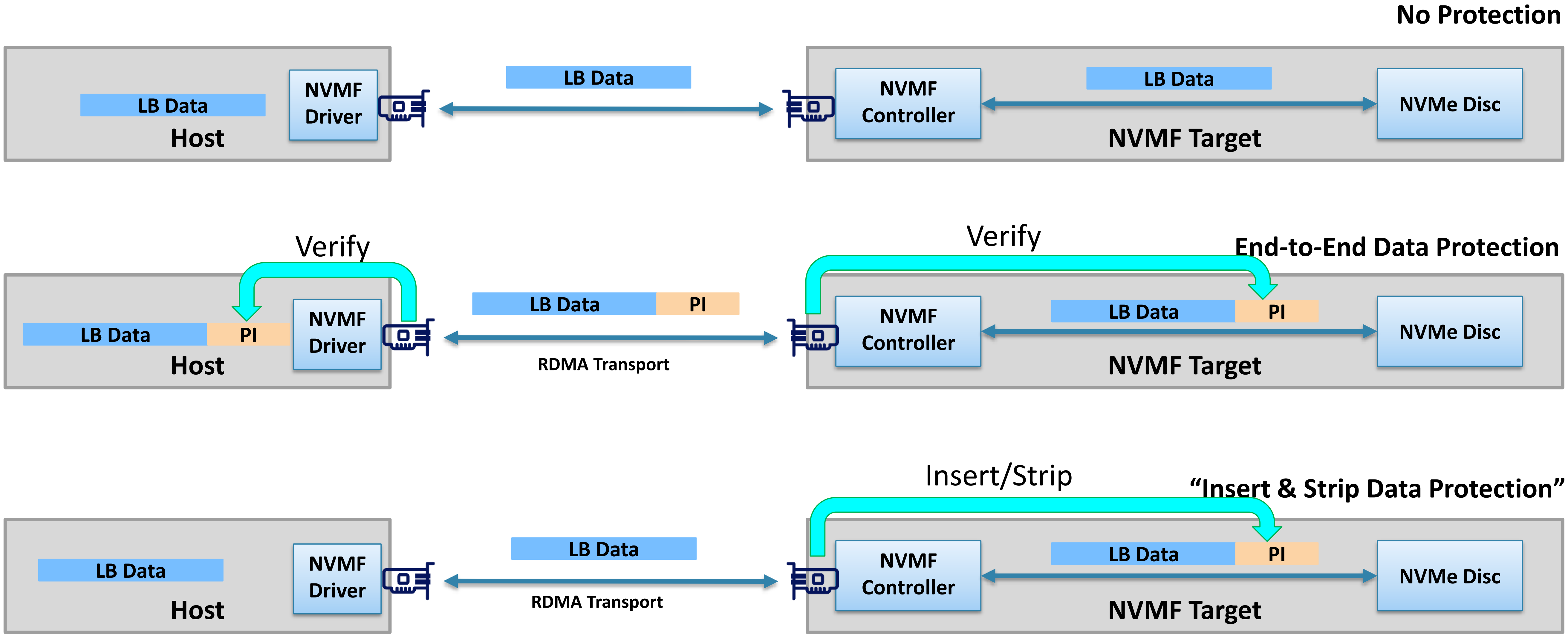
LBA N: PI

LBA N+1: PI

NVME-PCI. Data protection



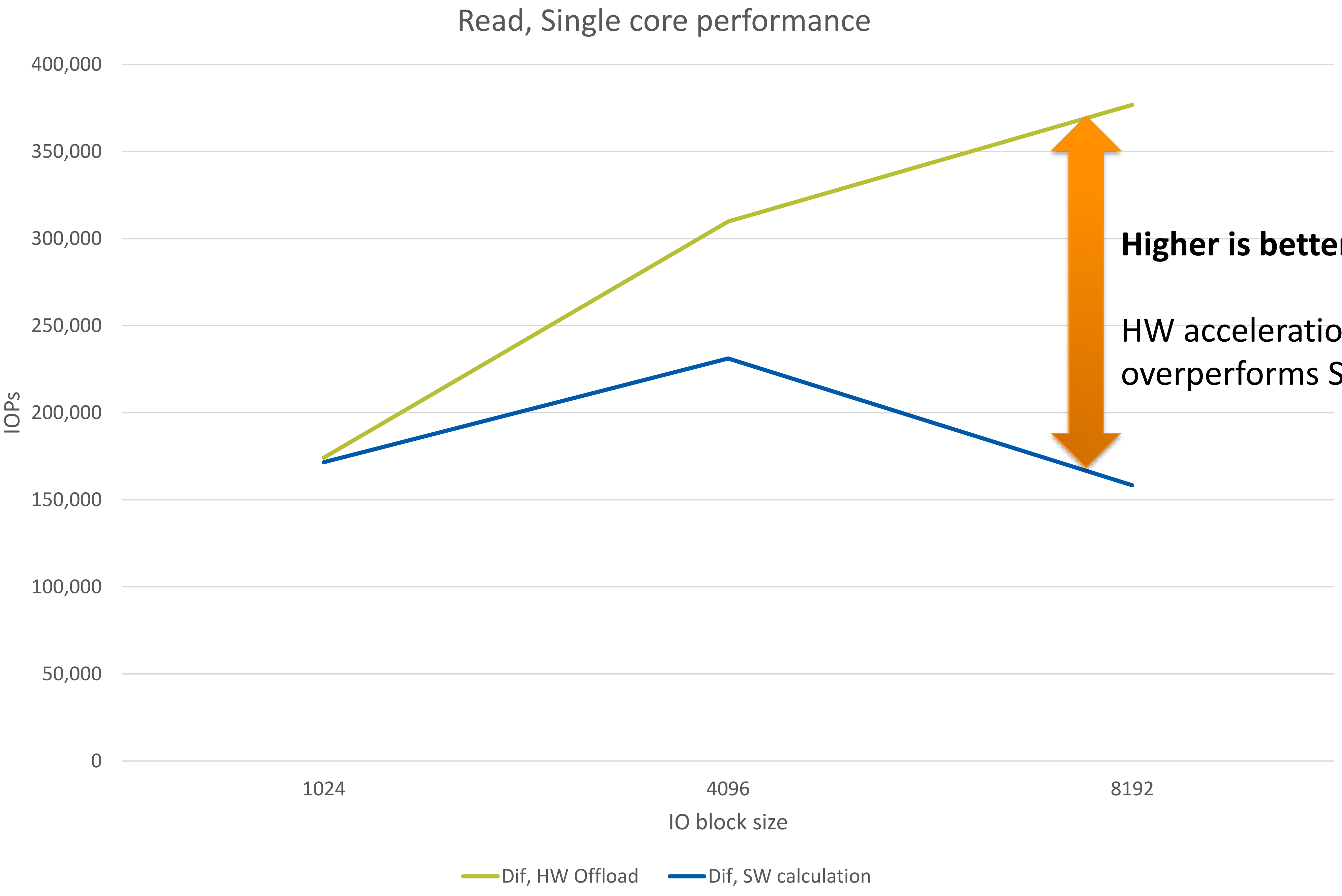
NVME-OF. Data protection



SPDK. DIF “Insert & Strip” mode

- DIF “Insert & Strip” mode in TCP Transport
  - Shuhei Matsumoto, “Hitachi” : https://review.gerrithub.io/c/spdk/spdk/+/456452 - SW implementation
    - Available in SPDK v19.07
- DIF “Insert & Strip” mode in RDMA Transport
  - Aleksey Marchuk, Evgeny Kochetov, “Mellanox” : https://review.gerrithub.io/c/spdk/spdk/+/465248 - SW implementation
  - HW accelerated mode is under development : https://github.com/EugeneKochetov/spdk/tree/nvmf_rdma_sig_offload

DIF “Insert & Strip” mode. SW vs HW

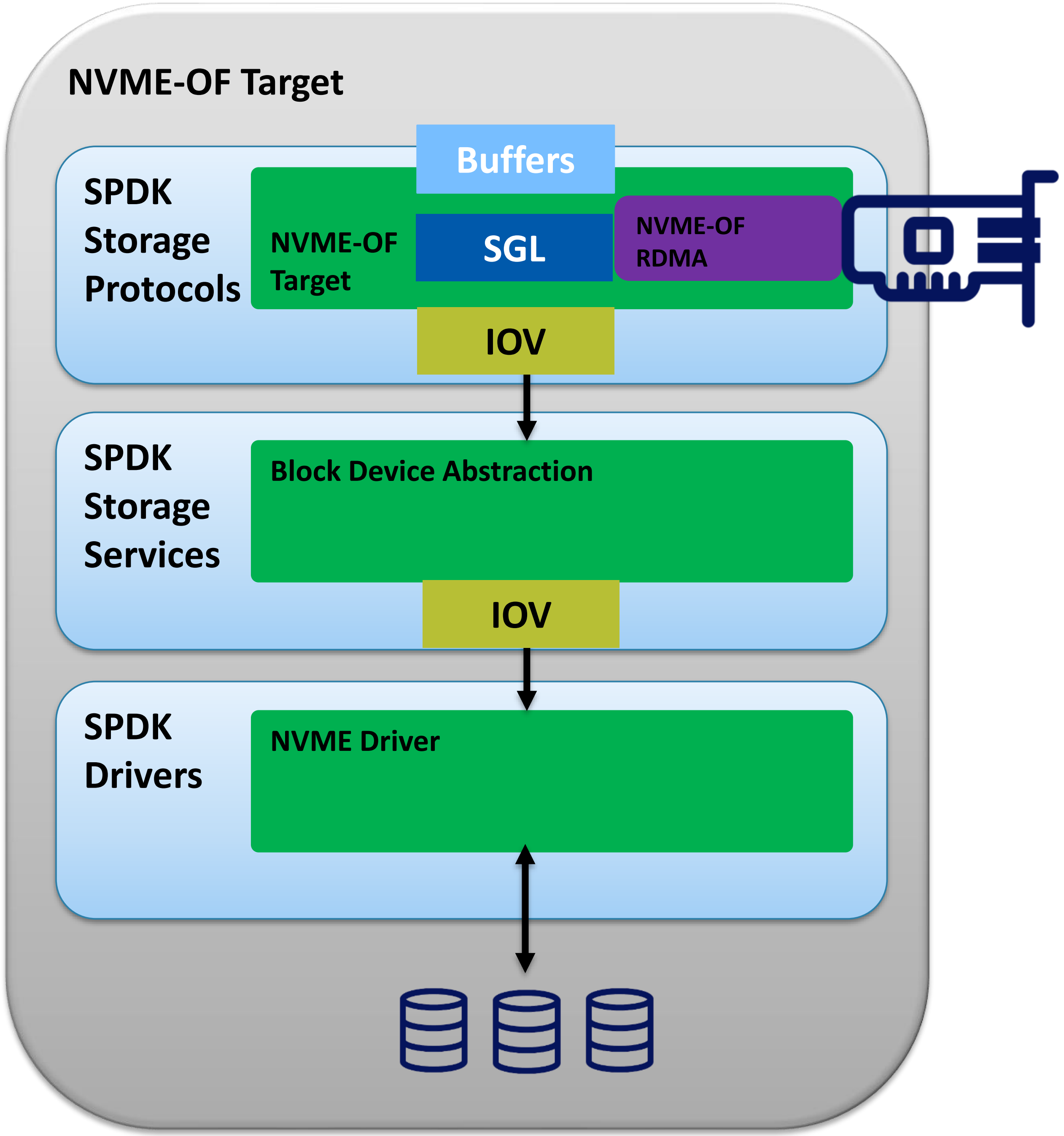


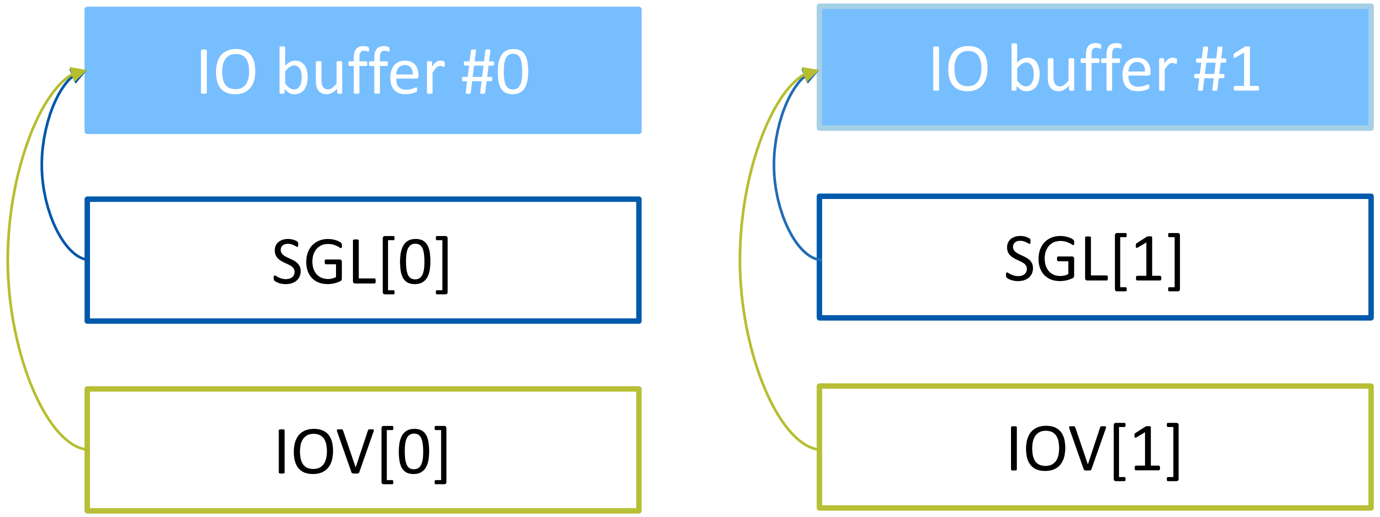
Higher is better

HW acceleration for DIF data protection overperforms SW by 200%

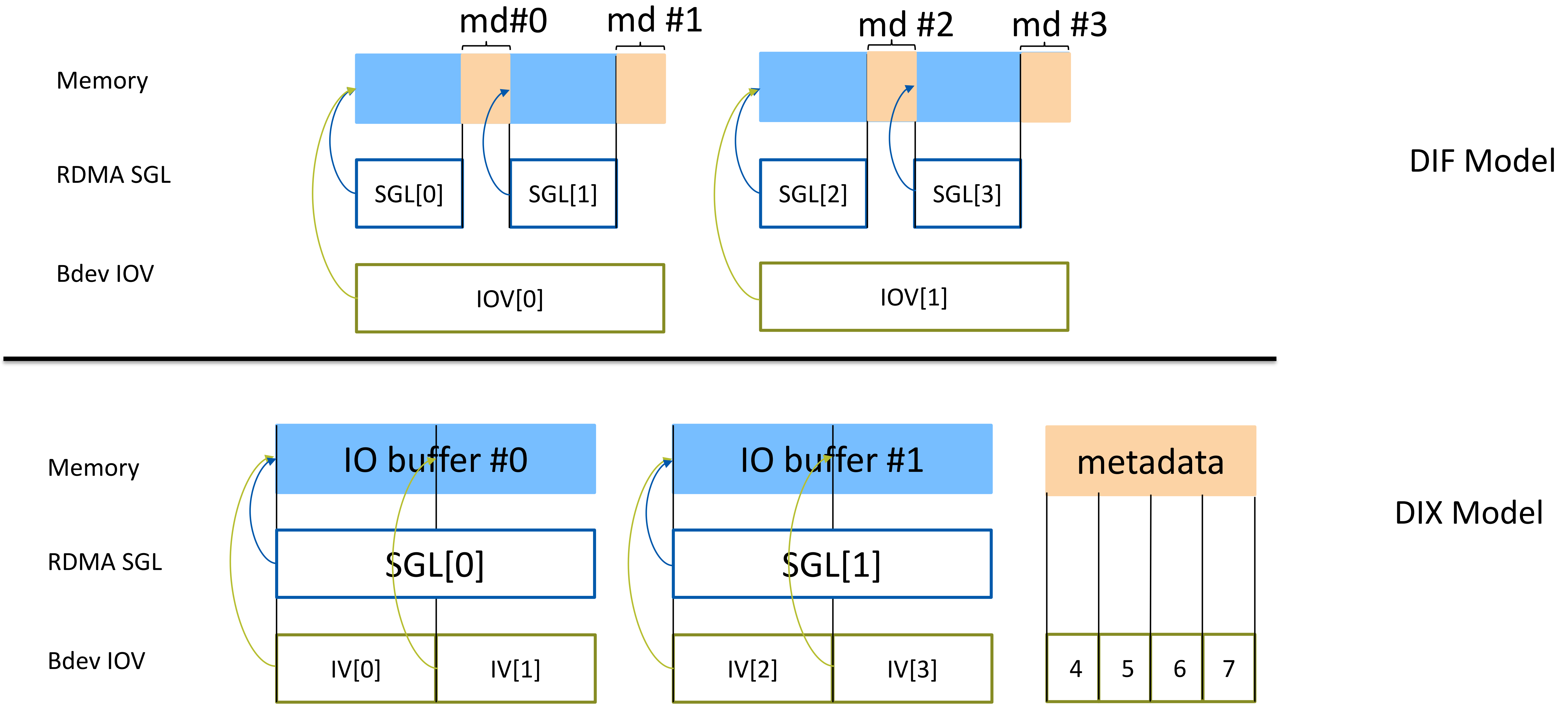
Queue depth: 32
Block size: 512+8
Disk: Samsung PM1725b
Platform: x86

SPDK. Memory management in NVME-OF RDMA





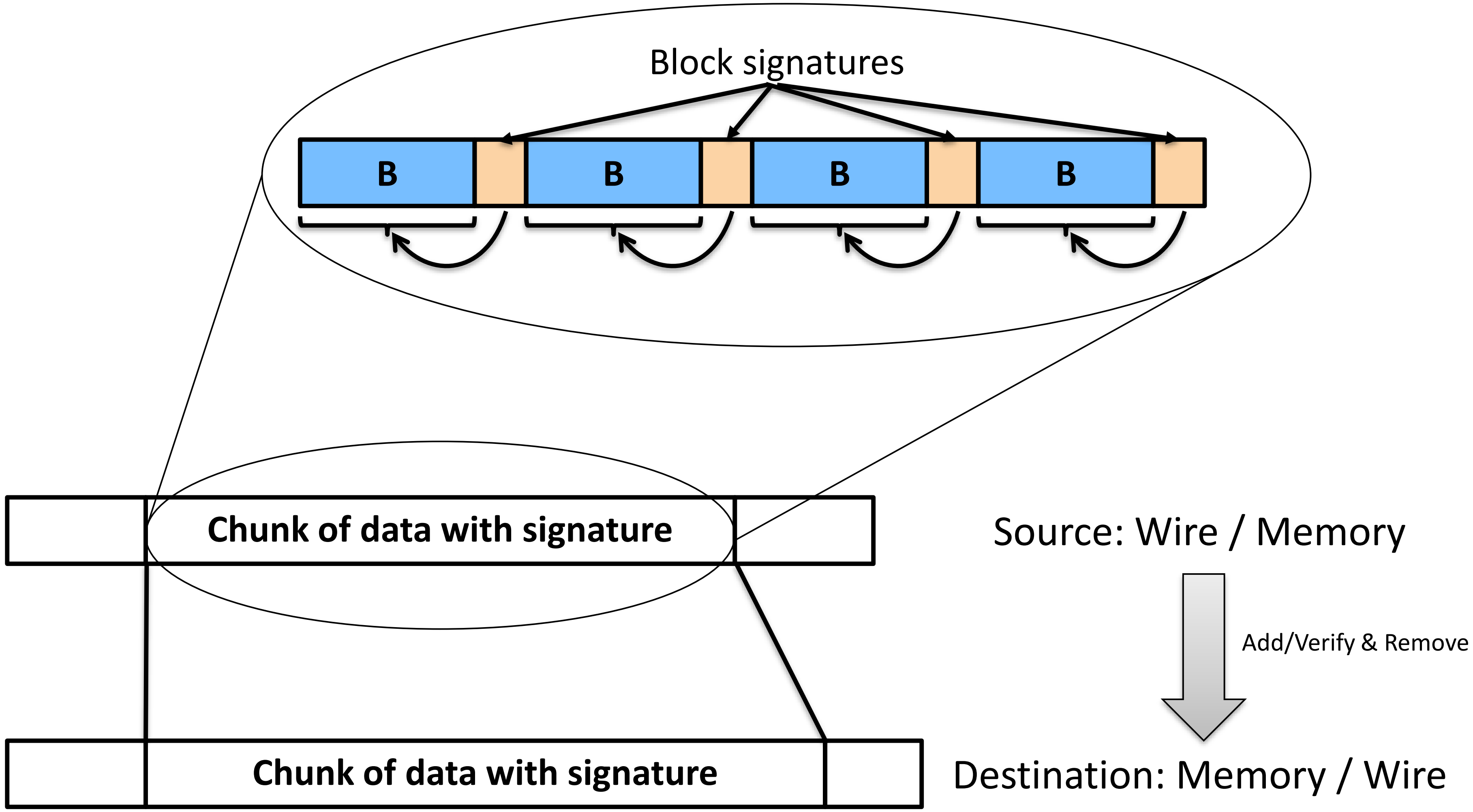
NVME-OF RDMA. Metadata placement



NVME-OF RDMA. Metadata placement

- “DIF” model increases number of SGL elements in RDMA layer
- “DIX” model increases number of IOV elements transferred to bdev layer
- In performance testing “DIF” model overperforms “DIX”
- “DIF” model is chosen as default option
  - Multi-element SGL will be can be replaced by UMR (“User memory region”)
- “DIX” model is used for “in-capsule” data

HW acceleration for “DIF”


Block signatures
B
B
B
B
Chunk of data with signature
Source: Wire / Memory
Add/Verify & Remove
Chunk of data with signature
Destination: Memory / Wire

“User space” API for “DIF”

- Signature operation is executed at data moving between two Signature Domains
  - Wire Domain
  - Memory Domain
- Signature Operations
  - Add
  - Verify
  - Verify & Remove
- Signature types
  - Repeating block signature. All blocks must have equal size
  - Transaction signature are used for protecting entire transaction
  - Variable block signature covers data of any size
- Using “indirect” memory referencing, both DIF and DIX modes are supported
- Planned to be submitted to “upstream” (rdma-core) in 2019

HW acceleration for data protection. Summary

- HW acceleration for guard tag calculation by NIC demonstrates advantage over SW implementation
- Roadmap:
  - User-spaces API for “DIF” manipulation. Submitting to “upstream”
  - HW acceleration for “Insert & strip” mode in SPDK’s implementation for NVME-OF target
  - HW acceleration for Data Integrity Field generation in SPDK’s initiator
  - Verifying DIF in network layer (RDMA) in “initiator” and “target” sides

Advanced hardware accelerations

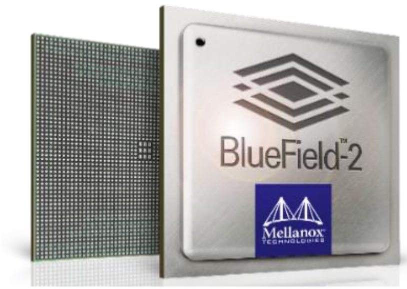
BlueField-2

Superior Storage Performance

- 8 Arm® A72 CPUs @ 2GHz-2.5GHz
- Dual 100Gb/s or Single 200Gb/s ports
- 16 lanes of PCIe Gen4.0
- Up to 5.4M IOPs @ 4KB
- Lowest latency

Storage Accelerations

- NVMe-oF offloads
- NVMe-oF SPDK offload
- RAID, Erasure Coding, CRC32, CRC64 and T10-Diff



Storage Security

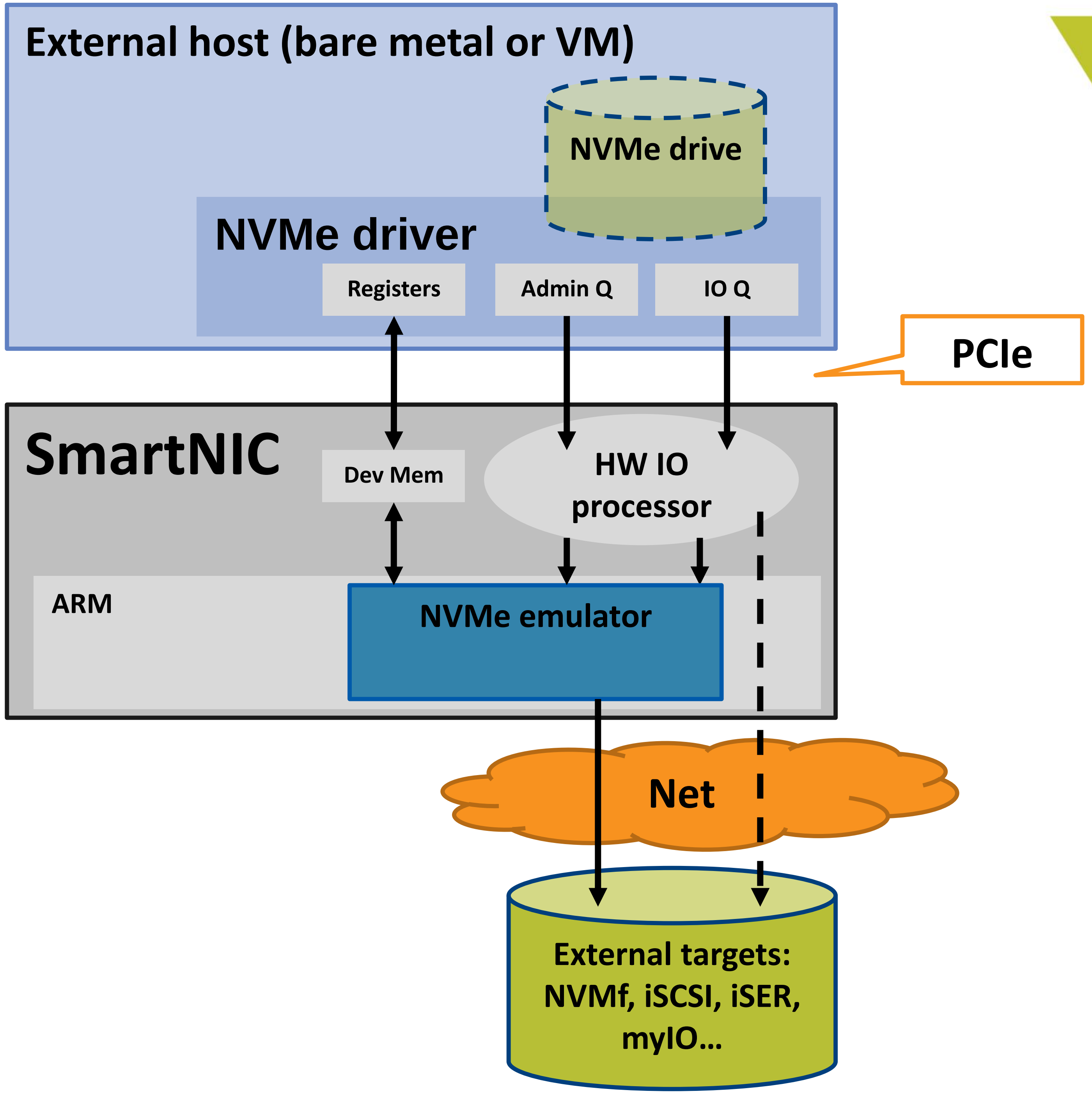
- Data-at-Rest AES-XTS encryption
- Authentication/Authorization services
- Encryption and decryption of data to/from storage
- Protection between users

Unique Features

- Data (De)Compression
- NVMe SNAP™
- Deduplication

NVMe SNAP

- Emulate locally attached PCIe NVMe drive
- Unmodified NVMe driver on host
- NVMe queues serviced in ARM
  - Then go to network
  - Admin Queue, IO Queues
- Optional: IO path skips ARM
  - Protocol conversion on IO processor
  - Must be simple enough
  - Must be RDMA
  - For example: NVMe-oF
  - Lose IOP-level software manipulation option
  - Admin queue still in ARM



SPDK as NVMe emulators standard framework

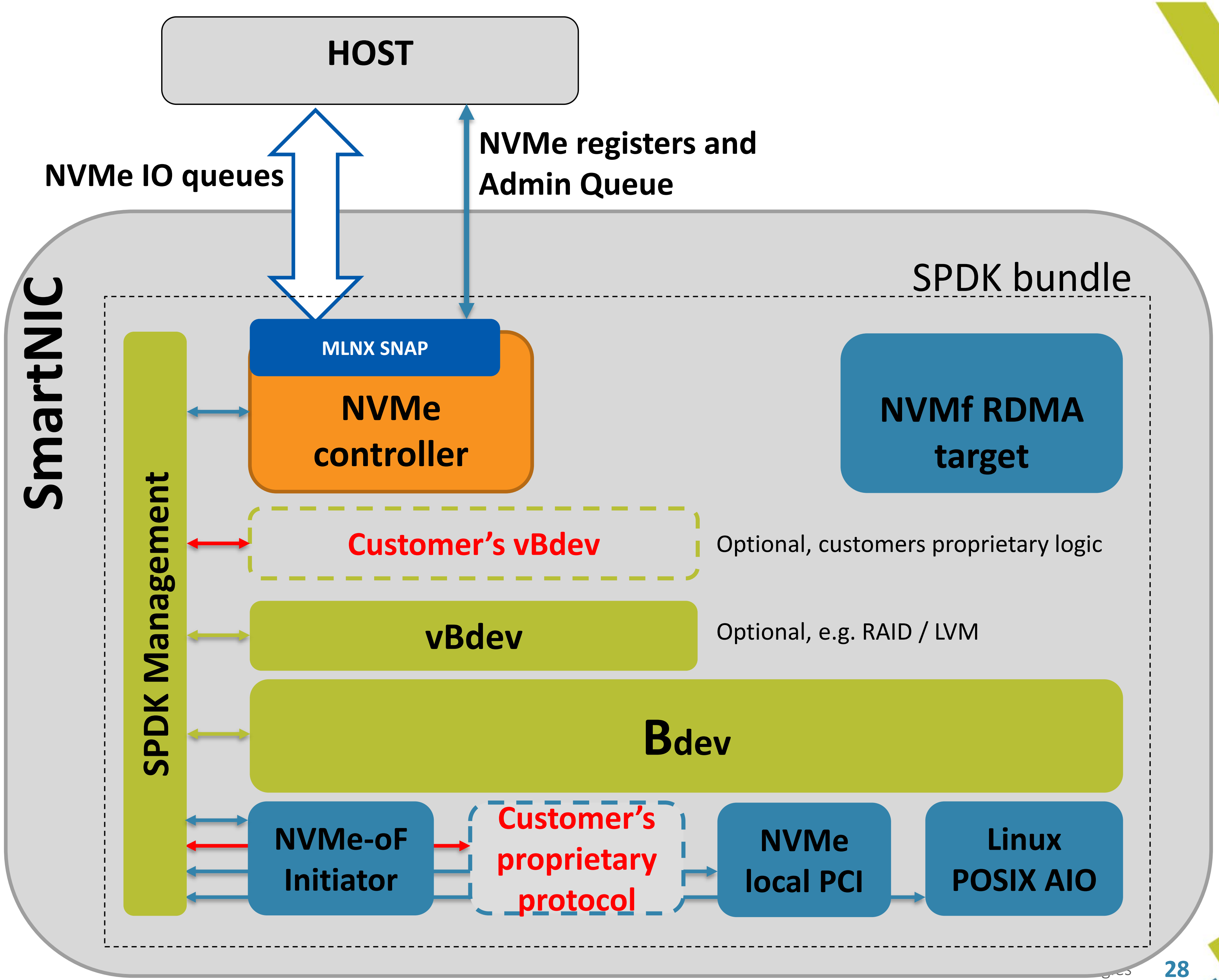
NVMe controller

- New: NVMe controller
  - NVMe device-side registers
  - NVMe device-side admin commands
  - NVMe device-side IO commands

- Vendor specific library
  - Bind to host NVMe device emulation

- Shared code and .h files
  - With NVMe driver
  - With NVMf target

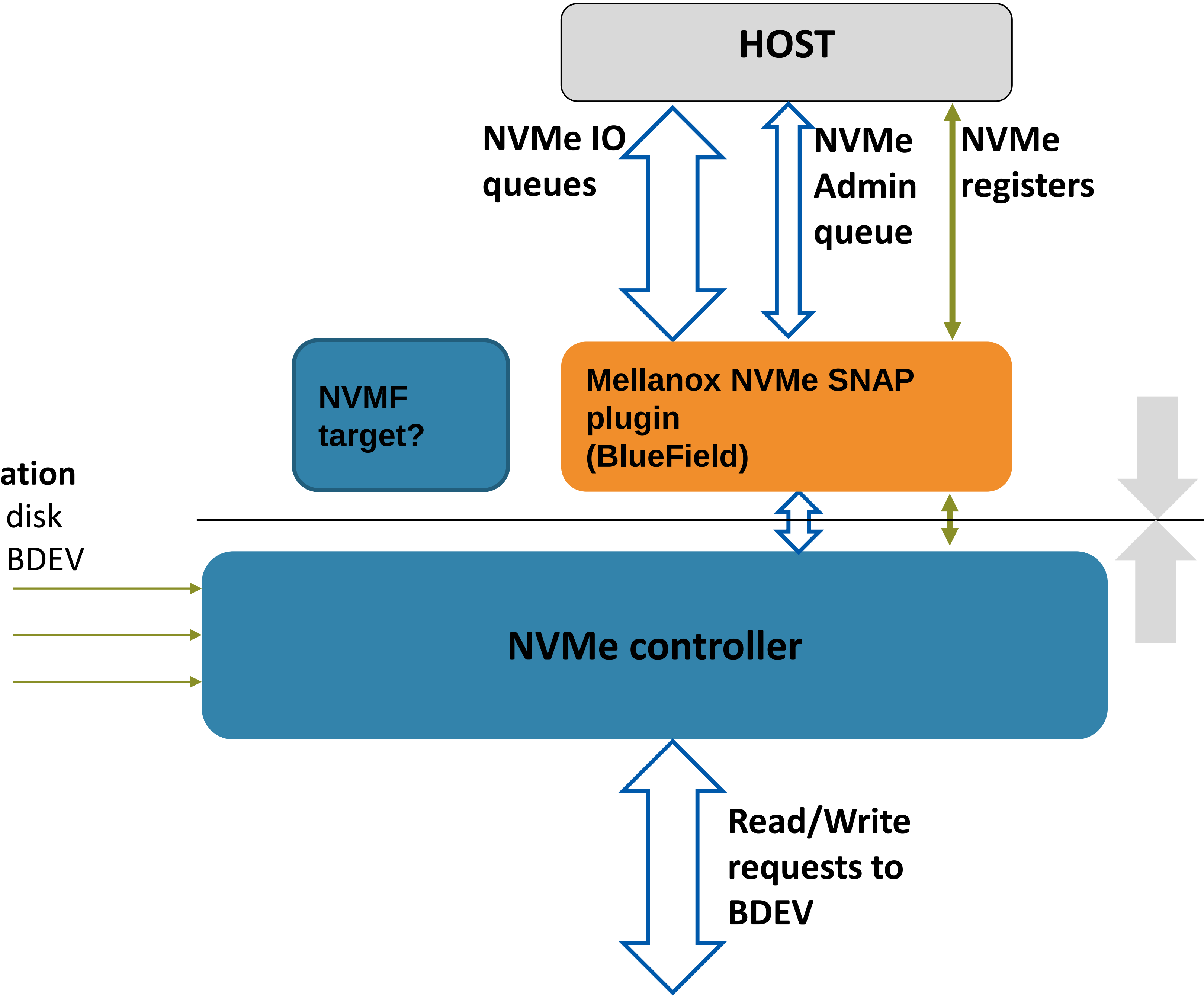
- Configuration is similar to NVMf target
  - Subsystem == emulated NVMe drive
  - Bind BDEVs as Namespaces



SPDK NVMe Controller

Zoom in

Target-like configuration
Subsystem = NVMe disk
Namespace map to BDEV



Register NVMe emulation driver
Register NVMe emulated disk
NVMe regs change
Admin request
IO request

NVMe controller provider API

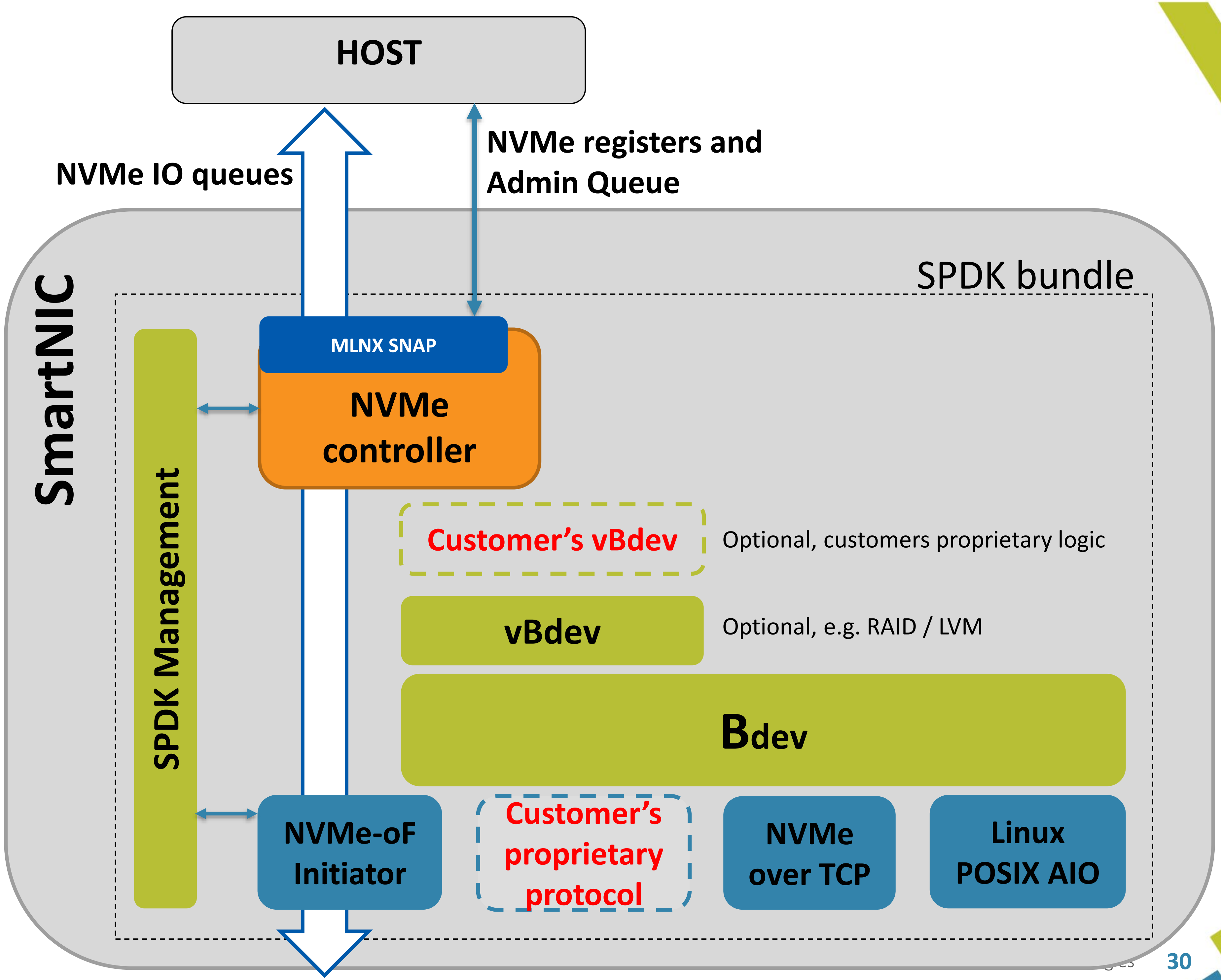
NVMe regs change
Create admin queue
Admin response
Create IO queue
IO response

NVMe Controller full-path offload

- NVMe SNAP to NVMf initiator offload

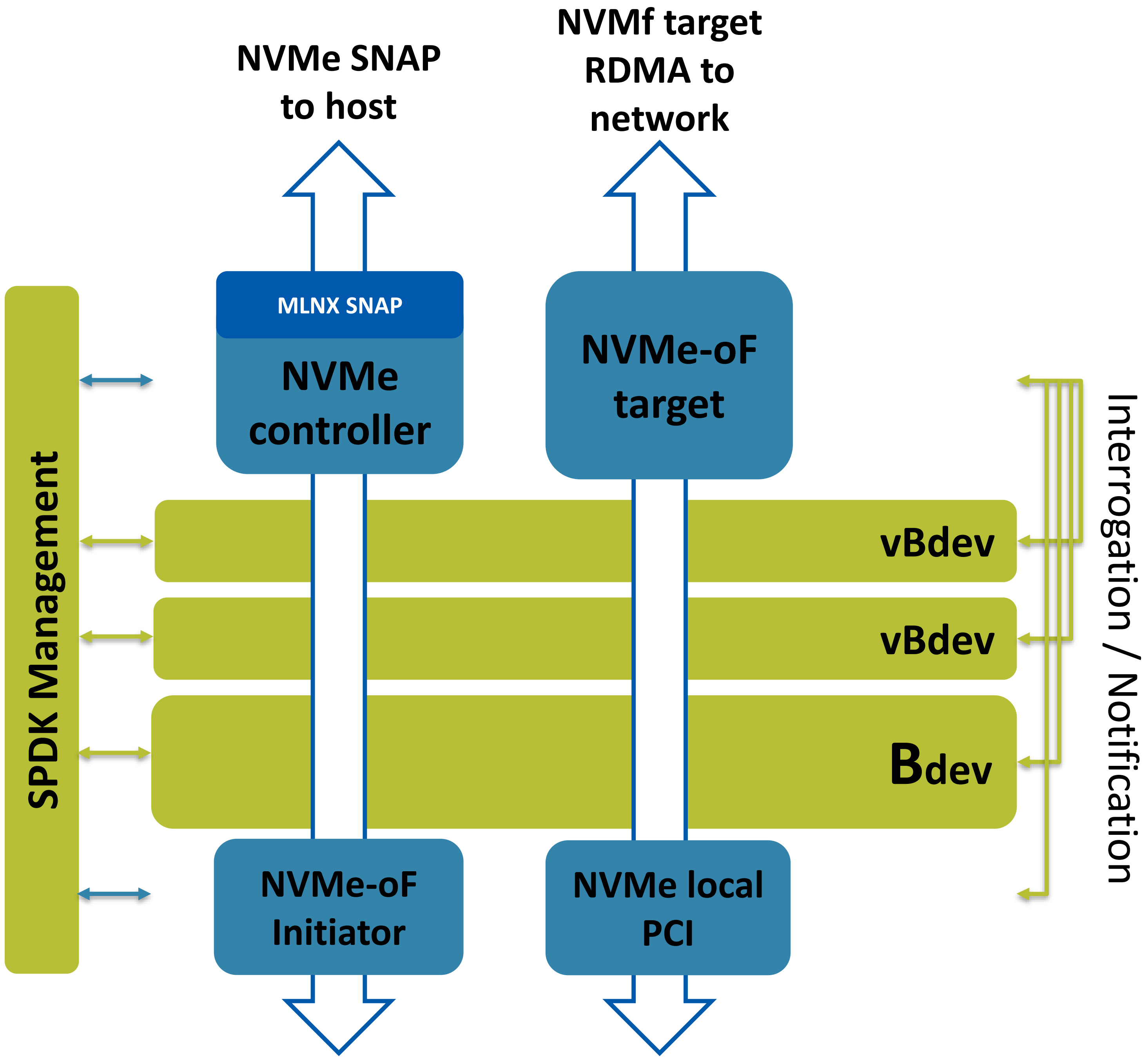
- Per emulated device configuration
  - Don’t offload
  - Always offload
    - Fail configuration if not possible
  - Best effort offload
    - Offload if possible, software path if not

- Best performance!
  - For simple use cases



SPDK in-network offloads

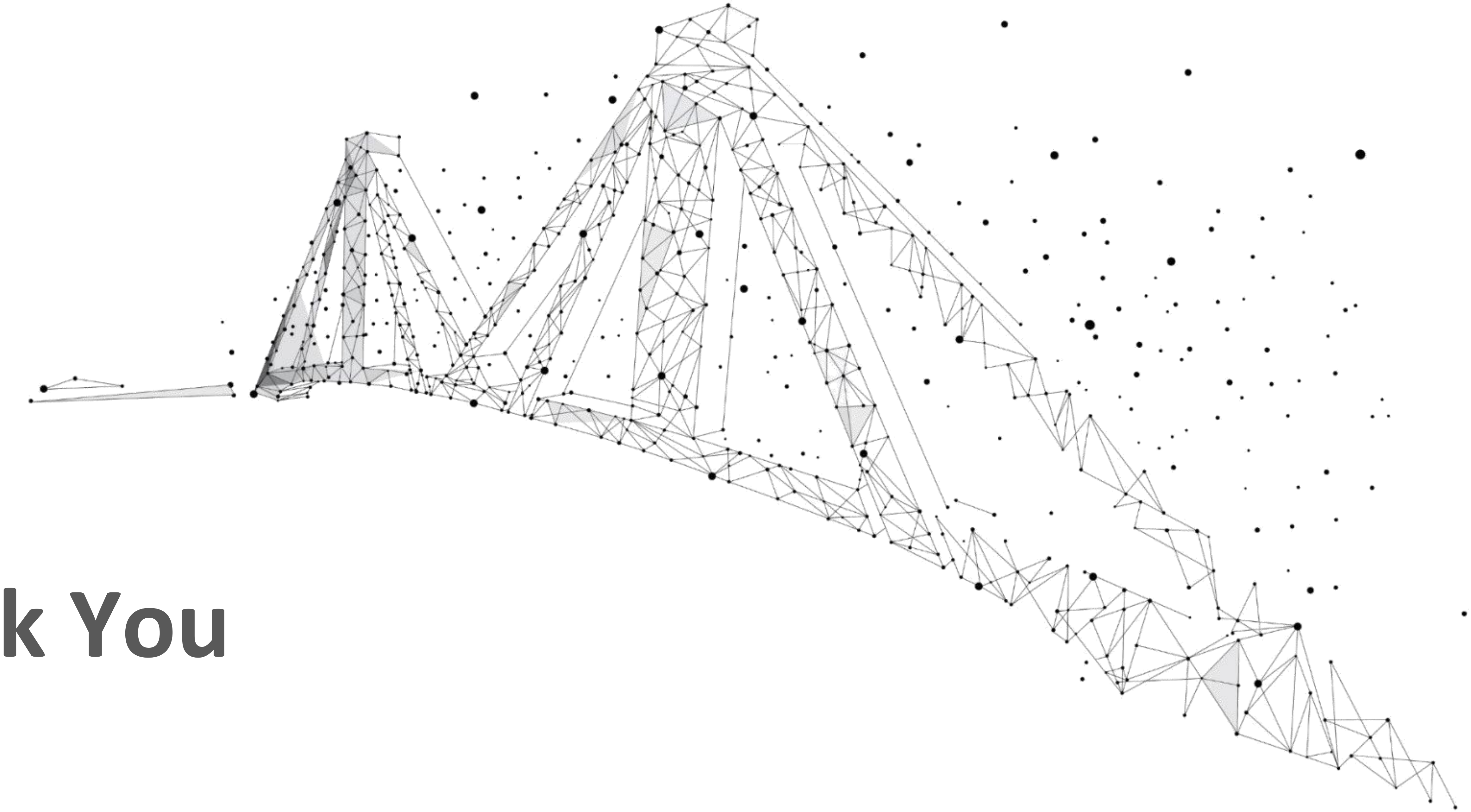
- vs. local mem-to-mem offloads
- Upper application configured to use a **bdev**
  - NVMe controller for SNAP
  - NVMe-oF target
- Interrogate vbdevs/bdevs chain
  - Identify the kind of bdev (NVMf, iSCSI, Crypto…)
  - Get configuration / create resources
  - If vbdev, get next (v)bdev(s), repeat
- Can the full flow and configuration be offloaded?
  - If yes – allow offload, configure device
  - If no – continue in software
- Notification for runtime changes in configs
  - Thin provisioning new chunk mapped
  - Volume resized



Mellanox® TECHNOLOGIES

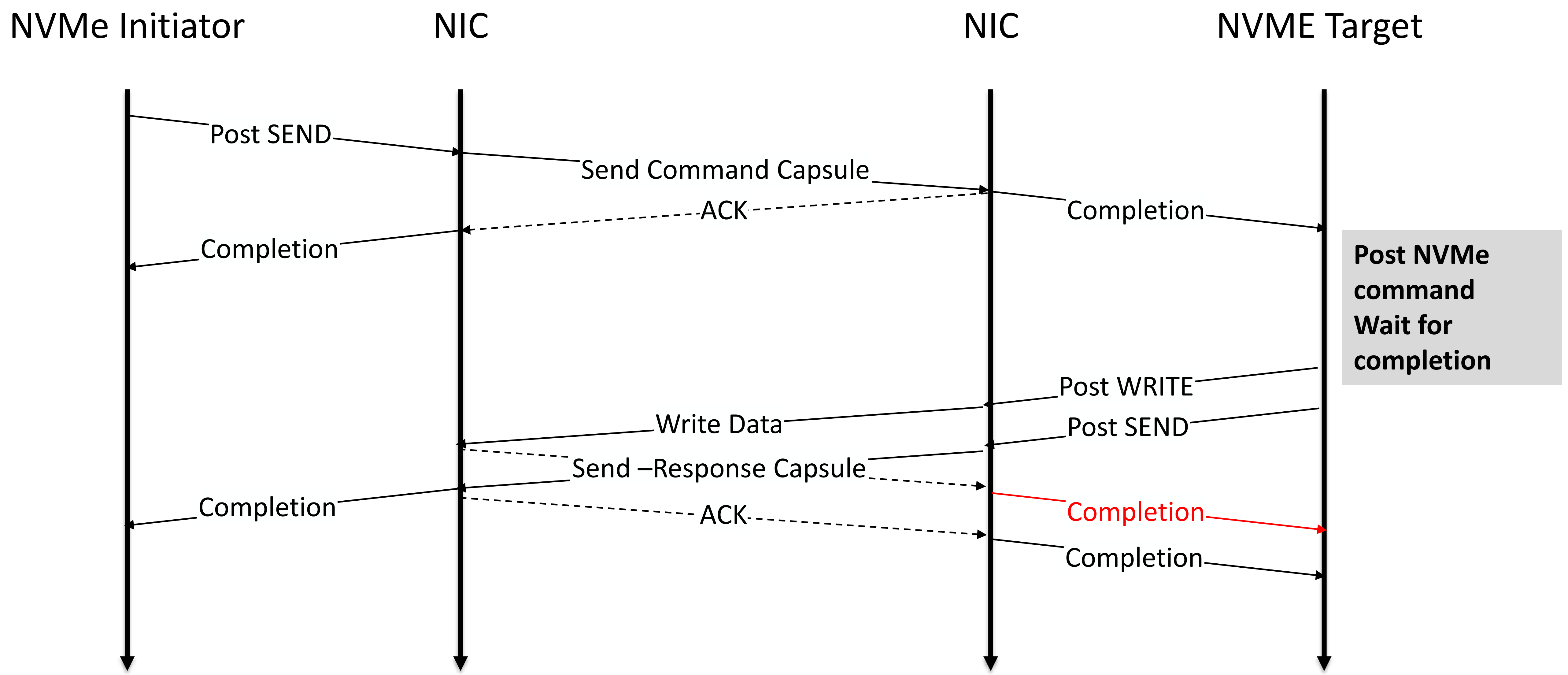
20 YEAR ANNIVERSARY

Thank You

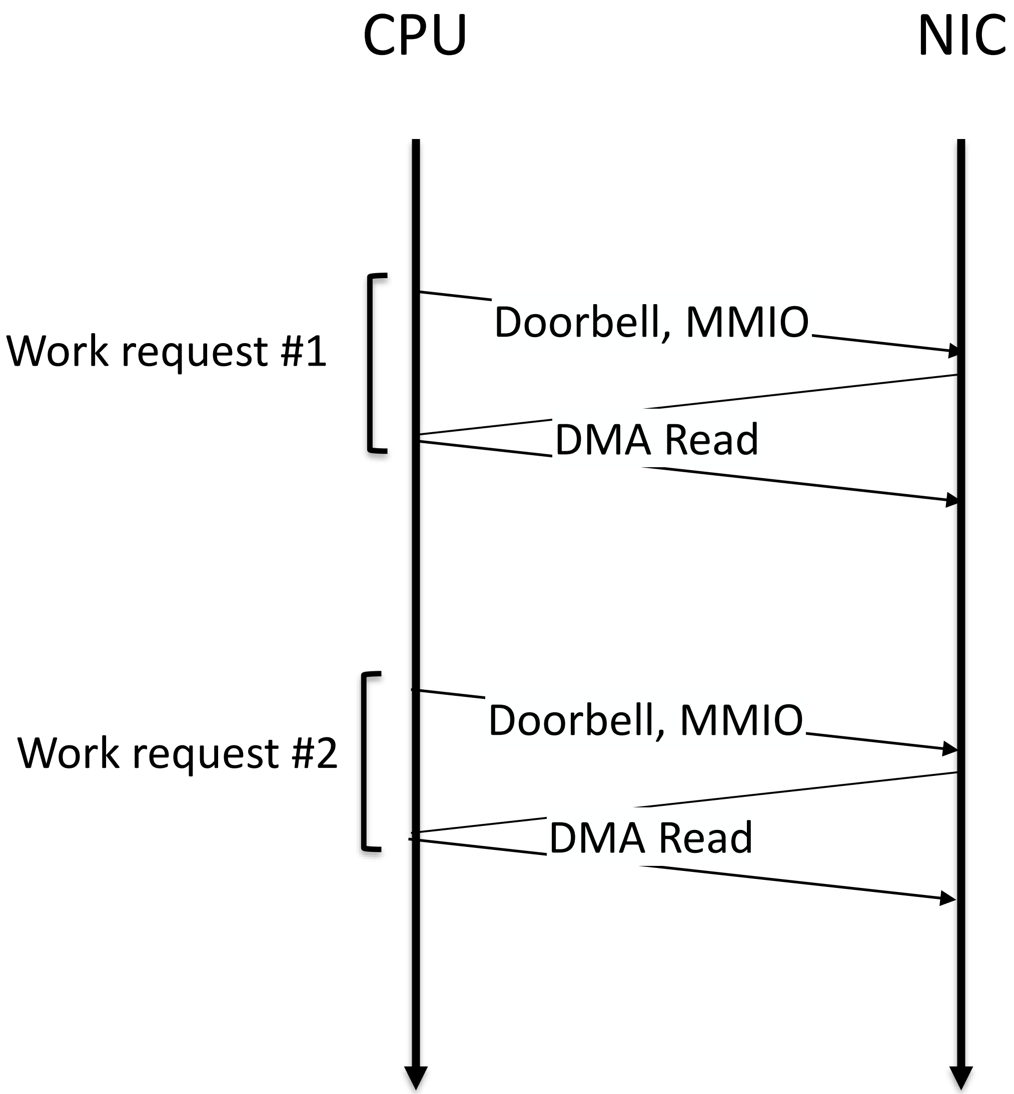


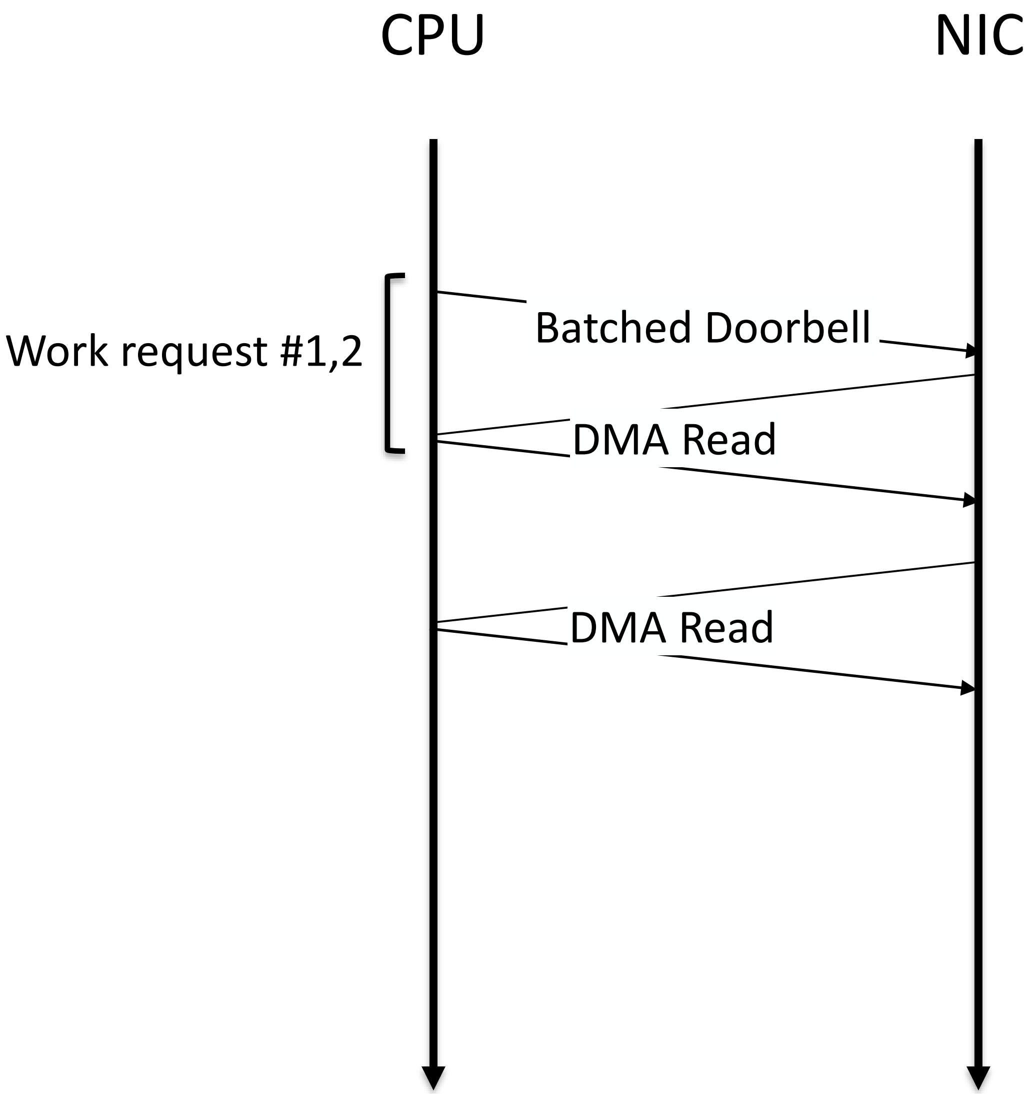
Backup

NVME-OF RDMA. IO Read. Selective signaling



NVME-OF RDMA. Request batching


CPU
NIC
Work request #1
Doorbell, MMIO
DMA Read
Work request #2
Doorbell, MMIO
DMA Read


CPU
NIC
Work request #1,2
Batched Doorbell
DMA Read
DMA Read