2020 OFA Virtual Workshop

USING SPDK TO OPTIMIZE YOUR STORAGE STACK

Seth Howell and Alexey Marchuk

Intel and Nvidia

AGENDA

- **Design Goals and Primitives**
- **Initiator Side Port Failover**
- **RDMA performance optimizations for NVMe-oF protocol:**
  - Non-signaled RDMA_WRITE operation
  - Work requests batching
- **Data Integrity:**
  - Overview of Data Integrity, T10DIF format
  - SPDK SW implementation (DIF insert or strip)
  - T10DIF HW offload using NVIDIA Mellanox NIC

NVME-OF DESIGN

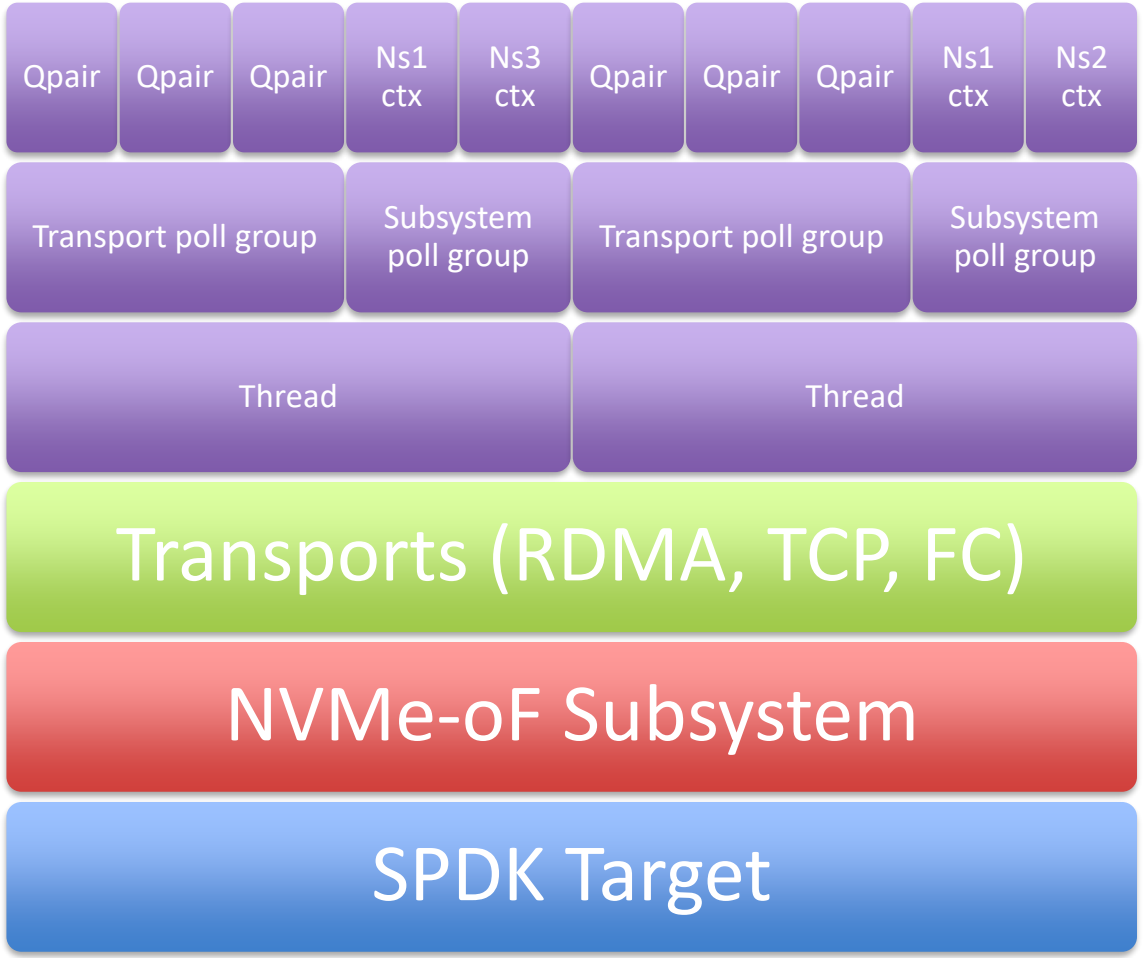
SPDK NVME-OF DESIGN GOALS

- **Userspace Implementation of NVMe-oF Stack**
  - Ideal for RDMA
- **Lockless in the I/O path**
- **Scalable (Efficiently handle many connections to many devices)**
- **Simple and Efficient Transport Plugin Interface**
  - Pluggable at two levels Transport and Provider
- **BSD Licensed**

SPDK NVME-OF PRIMITIVES

Form Fits Function

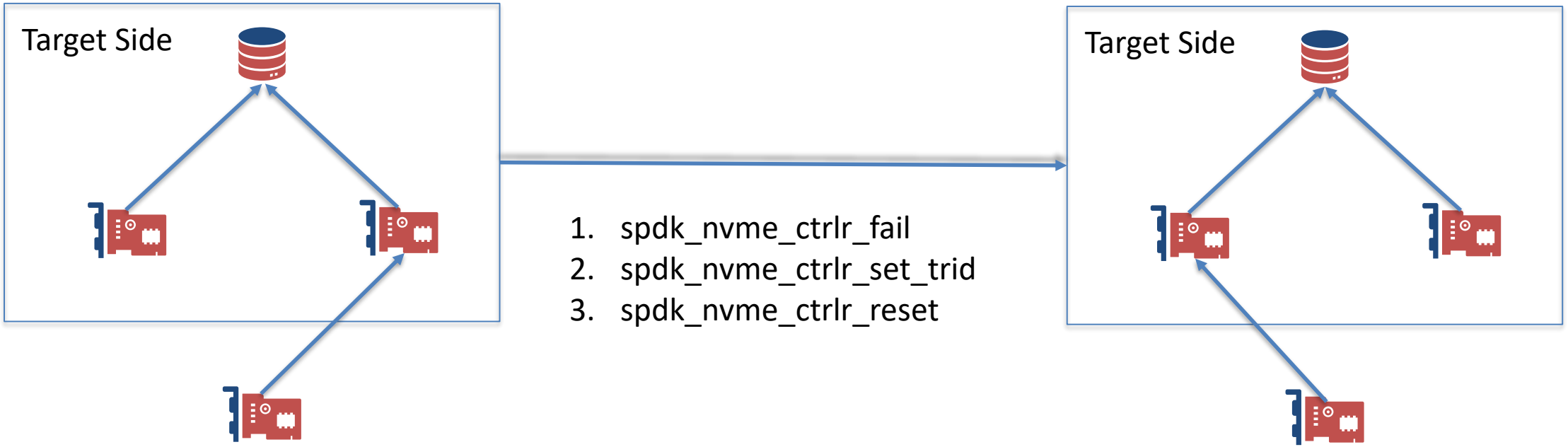
- **Globally available**
  - Subsystem – Some collection of NVMe controllers and namespaces. Unit of access control for NVMe connections
  - Transport – Function table and constants used to define an NVMe transport (e.g. RDMA, FC)
- **Per-Thread**
  - Subsystem Poll Group – per thread context containing controller information
  - Transport Poll Group – per thread context containing information needed to communicate over the transport. Unit for polling for more work on qpairs.*
  - qpair – Unique object only accessible from a single thread. On the target this object must belong to a poll group



RDMA PERFORMANCE OPTIMIZATIONS FOR NVME-OF PROTOCOL

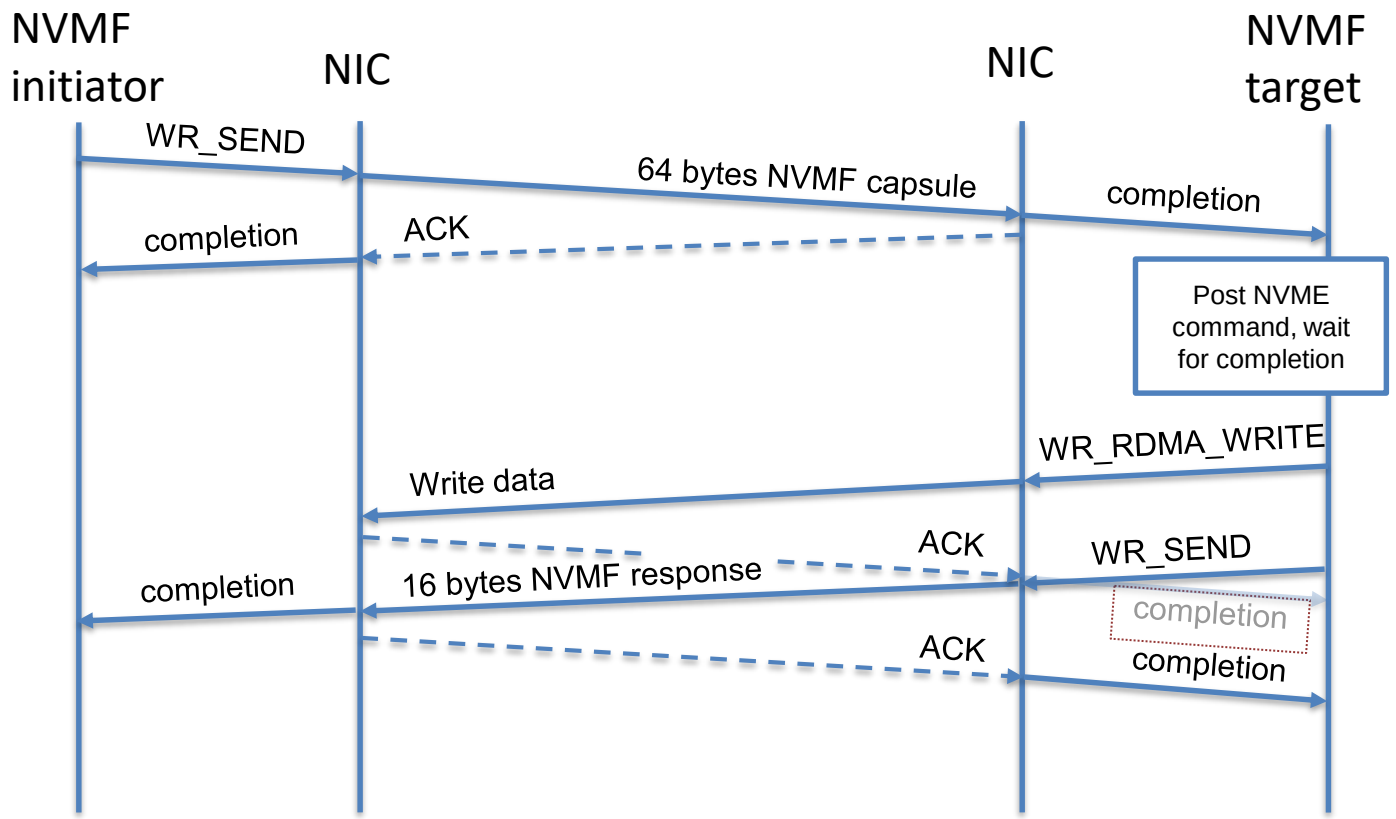
INITIATOR-SIDE MULTIPATH SUPPORT

- spdk_nvme_ctrlr_set_trid API allows users to change the TRID (IP address and port) of an NVMe controller with minimal application overhead.
- All qpairs remain present in the initiator, but are reconnected to the new Transport ID.
- Useful for load balancing or failover in case of a target side network event.



NON-SIGNALED RDMA_WRITE OPERATION

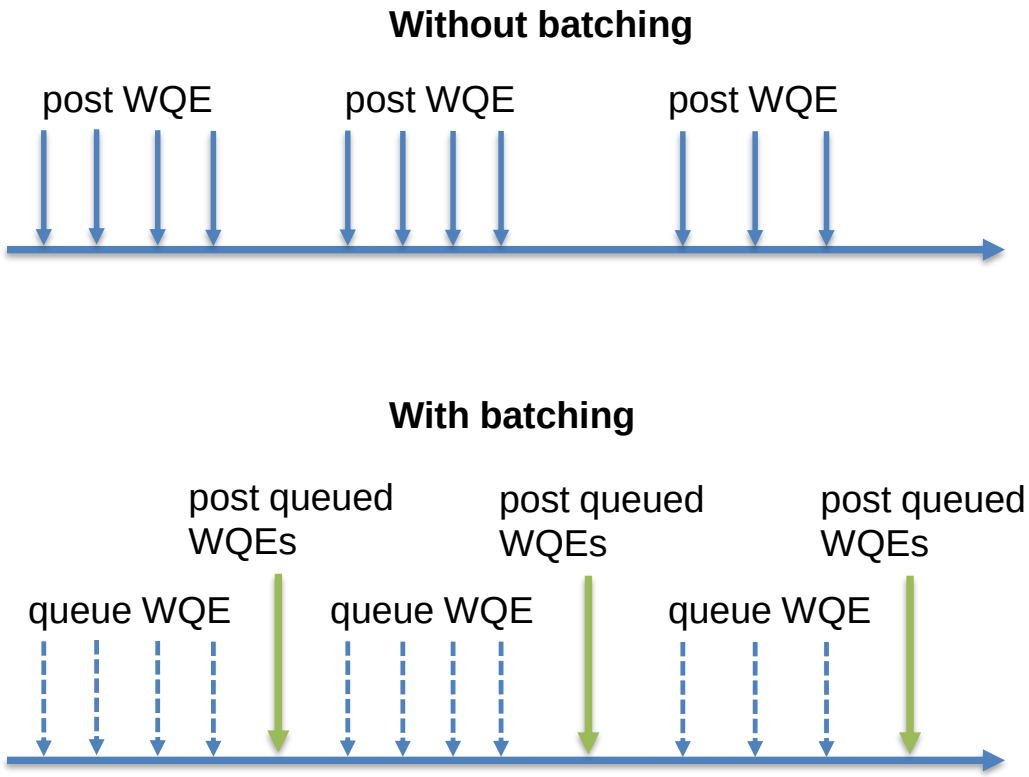
- **In IO Read flow, RDMA_WRITE is followed by RDMA_SEND**
  - Completion for RDMA_WRITE can be skipped
- **Reduces the amount of PCI transactions**
- **Saves on processing of extra Work Completion, increases IOPS in read flow: up to 15% on ARM**
- **Merged to SPDK in v19.07**



WORK REQUESTS BATCHING

In SPDK NVMEoF target and initiator

- **Every IO operation requires posting of 1 or more SEND and 1 RECV work requests**
- **The default approach for WQE (work request element) transferring requires separate MMIO for each WQE – update of "door bell" register**
- **WQE batching reduces CPU utilization and PCI bandwidth by using single "door bell" update for multiple WQEs**
- **WQE batching improves heavy load cases:**
  - NVMEoF target: when receive multiple IO requests in one ibv_poll_cq
  - NVMEoF initiator: when the user submits multiple IO requests before polling cq
- **Benefit on ARM:**
  - Randread: up to 5% more IOPS
  - Randwrite: up to 12% more IOPS
  - NVMEoF target, 64 queue depth, 4k payload

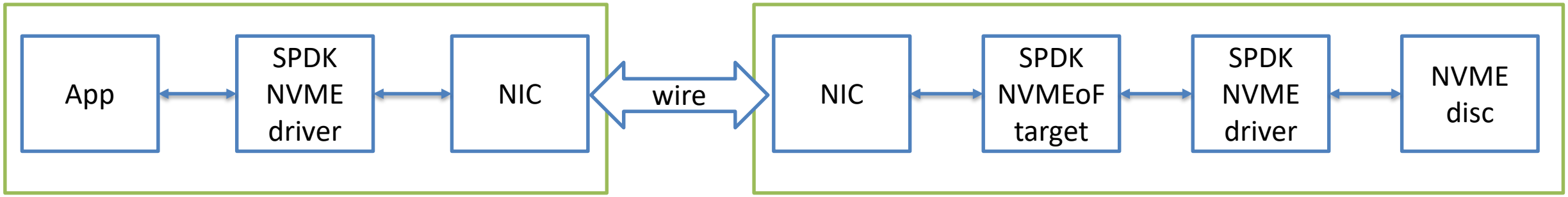


DATA INTEGRITY

DATA INTEGRITY ERRORS

Why should we bother?

- **Any data corruption is fatal to business logic**
- **Backups may contain bad data**
- **Data might be corrupted in any element of a data flow chain**

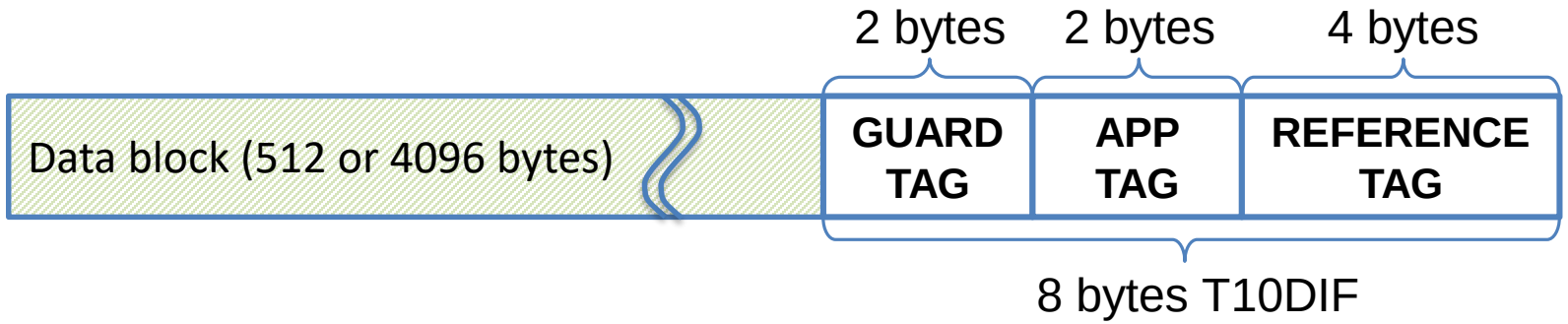


- **OS/driver/FW bugs - individual checks in each element can miss data integrity errors**
- **End-to-end data protection – proactive error detection:**
  - Write operation – the application is notified about the data corruption and can repeat the operation
  - Read operation – it is better to receive an error and not read the data rather than reading corrupt data
  - End-to-end protection is verified by NVME controller on read/write operations
- **Protection information is stored in metadata that needs to come together with data block**

T10DIF IN NVME

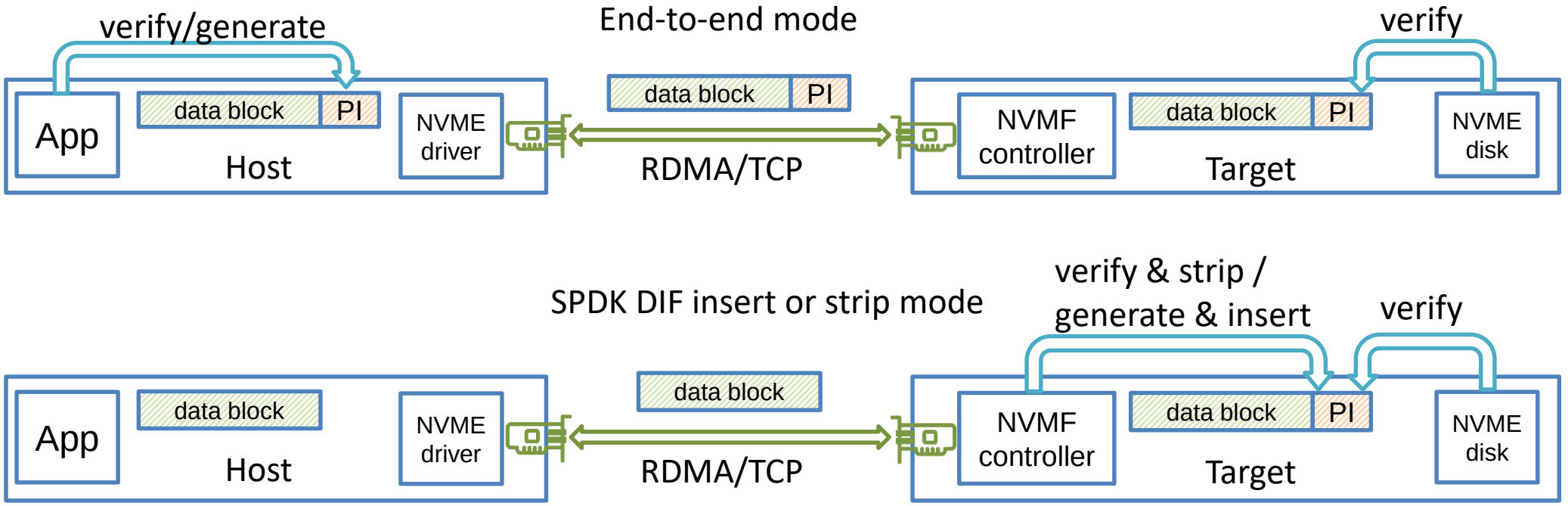
Standard overview

- **T10DIF - 8 bytes field with data protection for each data block (typically 512 or 4096 bytes)**
- **Guard tag**
  - 2 bytes CRC, covers data block
- **Application tag**
  - 2 bytes, user defined content, opaque
- **Reference tag**
  - Associates data block with an address, protects against misdirect block transfers. Usage depends on the PI type
- **Protection information (PI) type**
  - Type 0: no end-to-end data protection
  - Type 1: Ref tag must match 4 least significant bytes of LBA, incremented for subsequent logical blocks.
  - Type 2: Ref tag can be initialized to any value. Ref tag extends App tag.
  - Type 3: Ref tag for each subsequent logical block remains the same. Only Guard tag is verified
- **DIF: PI is contiguous with data block. Extended LBA format**
- **DIX: PI is stored in a separate buffer. Not supported by NVMEoF specification**



PROTECTION INFORMATION MODES

E2E and SPDK insert or strip



- **Insert or strip mode**
  - Metadata/PI capabilities are hidden from the host
  - Read operation: verify & strip PI received from NVME controller
  - Write operation: generate & insert PI after each data block before writing to the NVME controller

RDMA SW T10DIF IMPLEMENTATION

Insert or strip mode

- **Initiator writes/reads data without metadata, PI properties of namespace are hidden**
- **Verify or generate CRC16 (optimized by Intel ISA-L)**
- **Limitations:**
  - In-capsule data is not supported – no space for metadata
  - Only PI type 1 is supported since remote side doesn't fill App/Ref tag. Ref tag is set to 4 LS bytes of start LBA
- **Advantage: out of box solution suitable for applications which don't support PI**
- **Disadvantage: low performance due to SW calculation, high CPU load**
- **Merged to SPDK in v19.10**

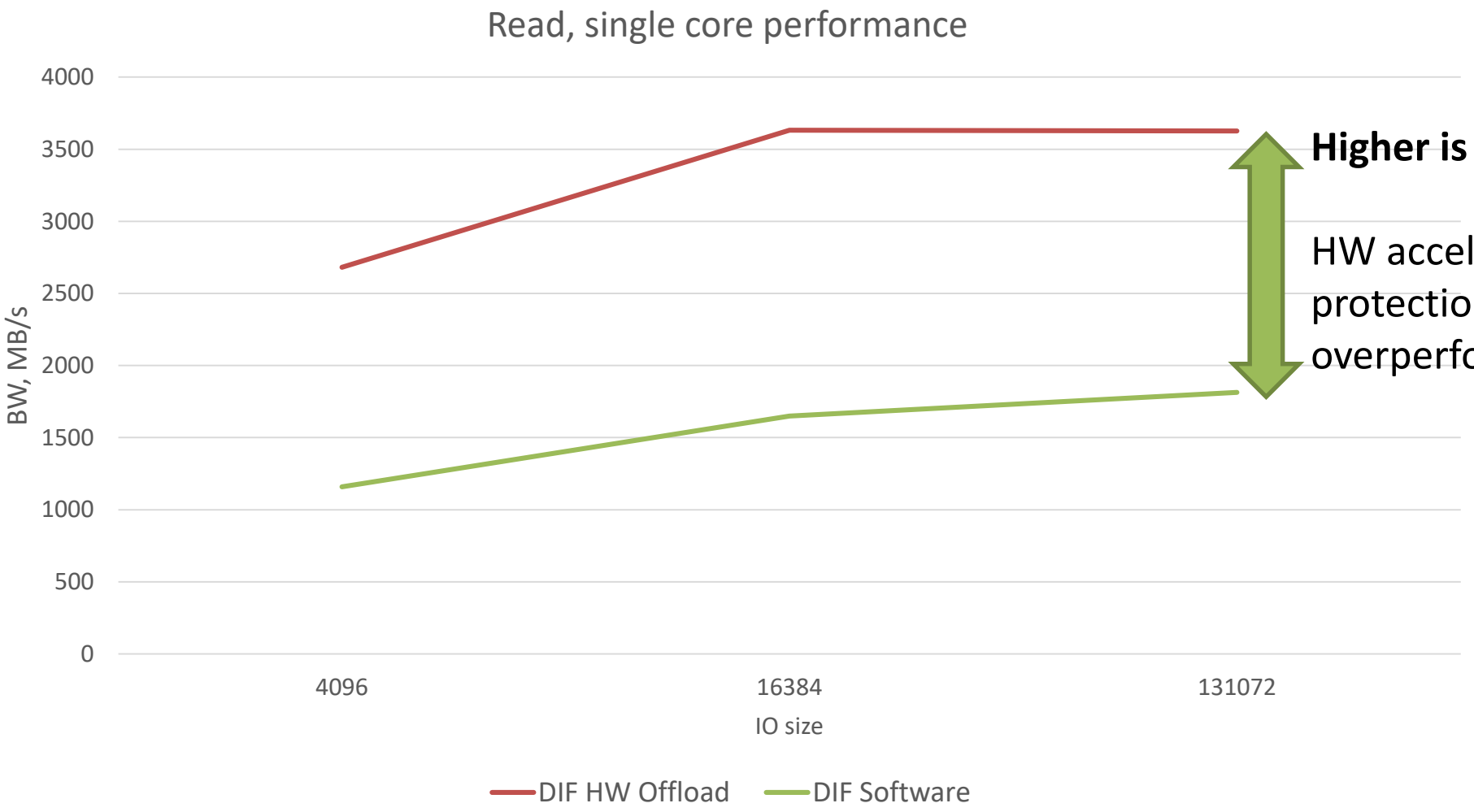
HW SIGNATURE OFFLOAD

Signature MR

- **User space API for T10DIF offload:**
  - Signature is property of memory layout. Signature MR (SIGMR) describes an IO transaction.
  - Two domains are defined in SIGMR – memory and wire.
  - Each domain can be configured with PI properties.
  - Combination of wire and memory domains allows us to use different scenarios – PI can be added/removed/verified in one domain and removed in another one. Can be used to support DIX in NVMEoF
  - Register SIGMR:
    - Build SGL with pointers to data and metadata buffers and local memory key
    - Register SIGMR using send request with special opcode
    - Use address and local/remote key from the registered SIGMR in data path
  - Invalidate SIGMR using send request with special opcode once IO operation is completed
- **Roadmap:**
  - Upstreaming of signature API to rdma-core (2020) – implemented in Mellanox Direct Verbs provider, an extension of UMR (User Memory Region) API
  - Enabling of T10DIF HW offload in SPDK

T10DIF PERFORMANCE RESULTS

SW and HW offload



Higher is better

HW acceleration for DIF data protection overperforms SW by 200%

Queue depth: 128
Block size: 4096+8
Disk: Samsung PM1725b
Platform: x86

2020 OFA Virtual Workshop

THANK YOU

Seth Howell and Alexey Marchuk

Intel and Nvidia

BACKUP

NVME/NVMEOF END-TO-END DATA PROTECTION

DIF/DIX

- **DIF: PI is contiguous with data block**
- **DIX: PI is stored in a separate buffer. Not supported by NVMEoF**
- **Namespace PI properties:**
  - PI type (0,1,2,3)
  - PI placement: first 8 bytes of metadata or last 8 bytes of metadata – if the namespace is formatted with more than 8 bytes of metadata
  - PI transfer: metadata is transferred at the end of data block (extended LBA format) or in a separate buffer
- **NVME read/write command properties:**
  - Protection Information Action (PRACT) – determines whether NVME driver should strip PI (read) or insert PI (write) or verify it and pass as is
  - Protection Information Check (PRCHK) – defines fields that needs to be checked – bitwise combination of Guard, App and Ref tags identifiers.
  - Initial Logical Block Reference Tag (ILBRT) – specifies the initial value to be used in E2E data protection
  - Logical Block Application Tag Mask (LBATM) – App tag mask, specifies which bits of App tag will be checked
  - Logical Block Application Tag (LBAT) – App tag value, to be checked with App tag contained in metadata

NVME PI TYPES

In details

- **Protection information (PI) type**
  - Type 0: no end-to-end data protection
  - Type 1: Ref tag must match Initial Logical Block Reference Tag (ILBRT) or Expected Initial Logical Block Reference Tag (EILBRT) field in NVME command. Ref tag must be incremented for each subsequent logical block in this command. ILBRT or EILBRT must be initialized to 4 least significant bytes of LBA.
  - Type 2: The same as type 1 except of ILBRT or EILBRT may be initialized to any value. Ref tag extends App tag.
  - Type 3: Ref tag for each subsequent logical block remains the same.

- **Tags impact on verification:**
  - For PI types 1, 2 if App tag has a value of 0xFFFF then all protection information checks are disabled
  - For PI type 3 if App tag has a value of 0xFFFF and Reference tag has a value of 0xFFFFFFFF then all protection information checks are disabled
  - the command may be aborted with status Invalid Field in Command if PRCHK bit which enables checking of App tag is set and PI type is 3