

SNIA DEVELOPER CONFERENCE



BY Developers FOR Developers

September 16-18, 2024
Santa Clara, CA

Leveraging SPDK Acceleration Framework for Optimal IPU/DPU Storage Workflows

Presented by Deb Chatterjee

Co-presenters: Jaroslaw Kogut, Konrad Szyber, Jacek Kalwas

Notices and Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing on certain dates using certain configurations and may not reflect all publicly available updates. Reach out to Intel for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

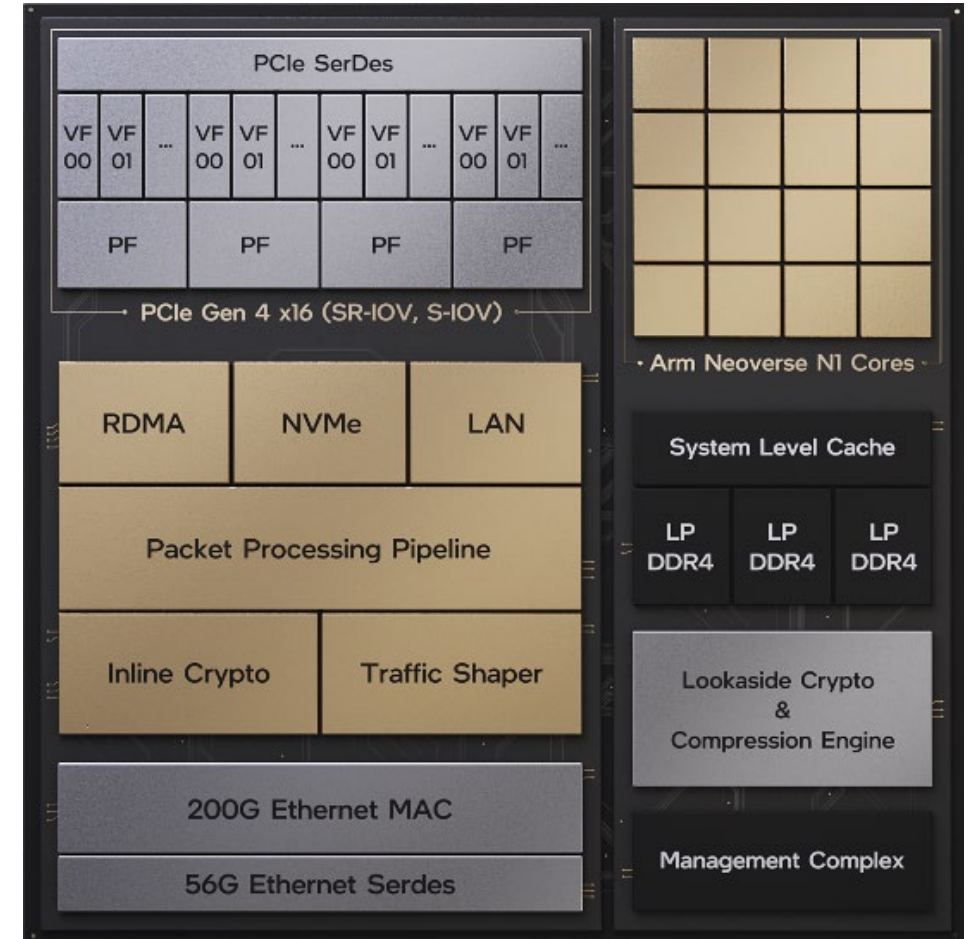
© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.



Introduction to the Intel[®] IPU and NVMe Initiator with SPDK

Intel Infrastructure Processing Unit (Intel® IPU)

- A new class of device in data center
 - Accelerates network packet processing, NVMe, RDMA
 - Includes accelerators for compression and encryption
 - Intel® IPU ES2000 commercially available
- Advantages and characteristic
 - Isolation of customer workloads from infrastructure tasks
 - Accelerates both data plane and control plane (through the 16 ARM N1 cores within the IPU)
 - Highly Programmable HW and SW
 - Excellent support for storage use cases

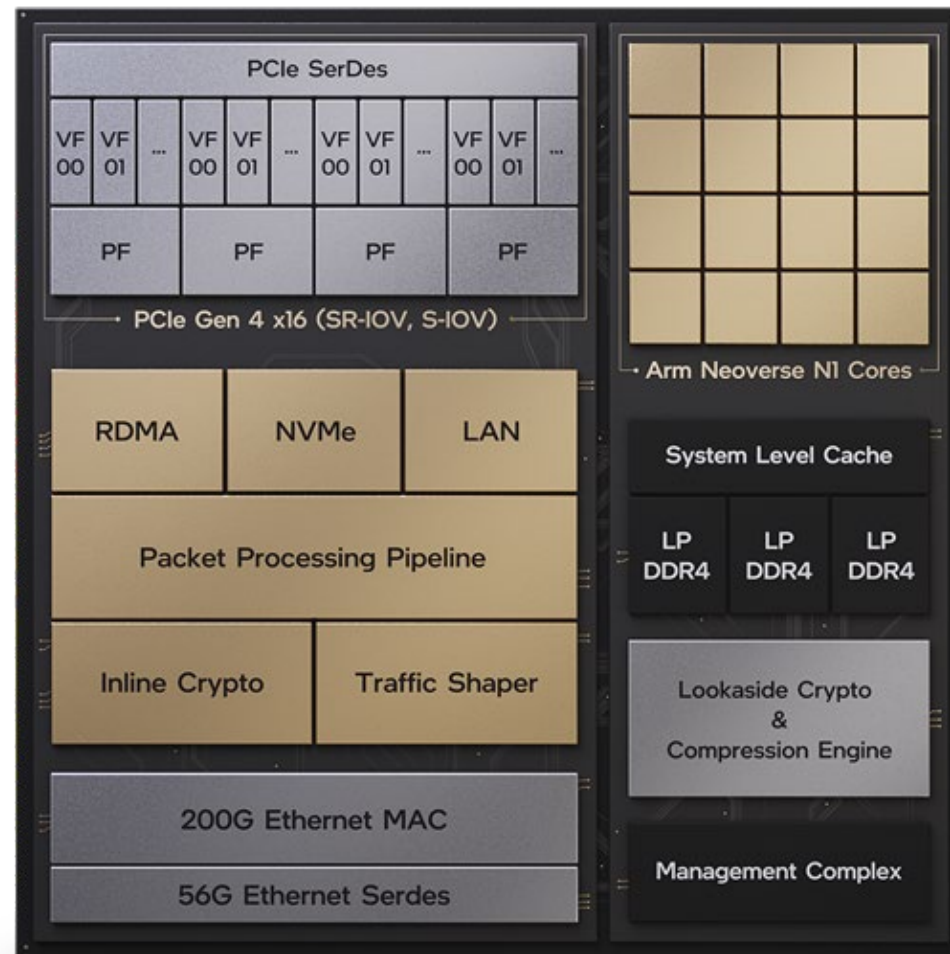


Intel IPU E2100 SoC

Performance Optimized 200GbE ASIC IPU

Shipping

Features	
Ethernet Controller: Up to 1x200GbE, 2x100GbE – 200G max BW <ul style="list-style-type: none"> Up to 200M packets per second LAN PCIe 4.0 1x16, 2x8, 4x4 ; Support for up to 4 Hosts 	
High Performance Compute <ul style="list-style-type: none"> Flexible on-board compute - up to 16 cores with shared L3 cache Up to 3 Channels LP-DDR4 	
Next Generation IOV <ul style="list-style-type: none"> 12K Flexible Host Queues, S-IOV, SR-IOV & Transmit Rate Limiters; Adaptive Virtual Function (AVF), Virtio-net, Virtio-blk, vmxnet3, custom virtual devices 	
Storage Features <ul style="list-style-type: none"> NVMe: NVMe Initiator Offload Customized Storage Protocols w/ AES-XTS & CRC offloads on the Compute Complex 	RDMA <ul style="list-style-type: none"> 200Gbps throughput; ~2us latency RTT; 150Mmsgs/s ; 1M Queue Pairs ROCEv2 transport Falcon - reliable low-latency hardware transport
Advanced Packet Processing <ul style="list-style-type: none"> P4 Programmable Pipeline w/ Inline IPsec, Hardware Connection Tracking & Stateful ACLs Up to 1M LPM Routes, Up to 16M Exact Match Entries, 1M Meters/Policers/Shapers, TCAM & range tables Programmable Parsing, Multi-stage Match-Action, Mirroring, Multicast, Modification & Recirculation 	
Security/ Crypto <ul style="list-style-type: none"> Inline IPsec ESP , AES-GCM 128/256 engine @ 170M pps per direction Lookaside 200G Bulk Crypto per direction, including TLS offload Internal/External RoT, Secure Boot, Secure Debug TRNG via management complex 	
Additional features <ul style="list-style-type: none"> IEEE 1588; NC-SI 	PRQ: Q4'23 – Contact us for controller specific sales interest



Intel IPU E2100 Adapter – available today!

- Intel IPU E2100 SoC (Mt Evans)
- 200GbE Ethernet Pipeline; 170MPPS Bi-Directional
- Inline crypto for network security at line rate
- Dedicated compute and local memory for control plane processing
- Full vSwitch offload in hardware
- NVMe Hardware Offloads
- Dedicated lookaside compression and crypto algorithms engines



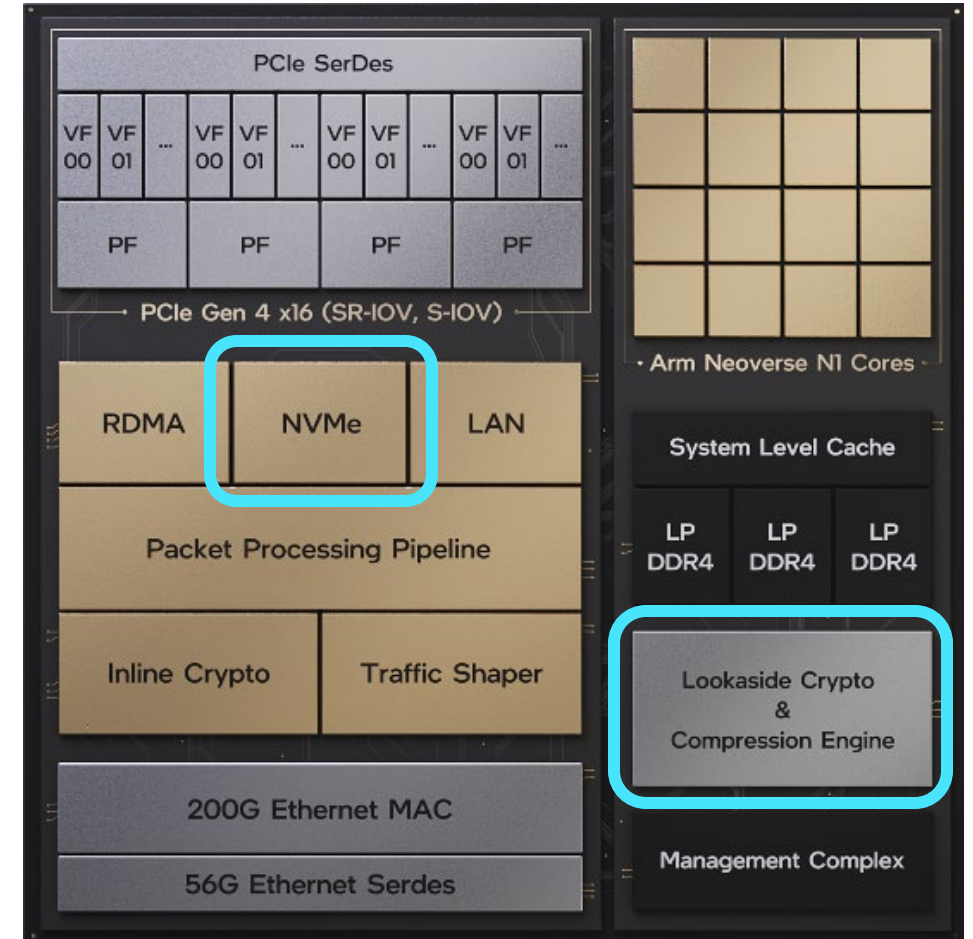
[Product information](#)

FEATURE	Intel IPU E2100 Adapter 2x100 GbE
Schedule	SRA / Production: Shipping
Ethernet Connection	1x 200GbE or 2x100GbE 4x25 NRZ/ 4x 50GbE PAM4 (via breakout cable)
Host Interface	PCIe 4.0 x16
Form Factor	PCIe CEM Spec Rev5 (with egress out east end) Full-Height, single-width, ¾ length
Connectors	2x QSFP56 connectors 1x RJ45 Connector to Datacenter Management Network 1x NC-SI Connector 1x USB Debug
EMI	FCC Class A
Operating Temperature	0-45 °C
Power	Auxiliary power required. Power consumption is workload dependent. Supporting MEV TDP at 75W.
Thermal Management	Passive heat sink with required airflow (TBD: ≈ 1000 LFM)
Compute	Arm Neoverse N1 cores– up to 16 cores
Memory	3 channels LPDDR4x, 16GB LPDDR4x per channel, total 48GB
Local Storage/Interface	64GB, PCIe Gen2 x1 (for ACC and IMC)
Manageability	Maintenance port (RJ-45) to IMC NC-SI 1.2, NC-SI
IPU OS support	Host: Rocky Linux, Red Hat IMC: Rocky Linux ACC: Rocky Linux

Shipping

Intel® IPU HW acceleration engines for storage workload

- NVMe Processing Engine
 - Enables NVMe Initiator offload
 - Exposes PCI Express (PCIe) NVMe devices to the host processor
- Lookaside Crypto & Compression Engine (LCE)
 - An array of advanced crypto algorithms and compression algorithms supported in HW
 - For storage, when data is at rest, LCE provides device-level encryption, storage-level encryption, and data protection on disk.
 - LCE supports chained operations, such as compression followed by encryption



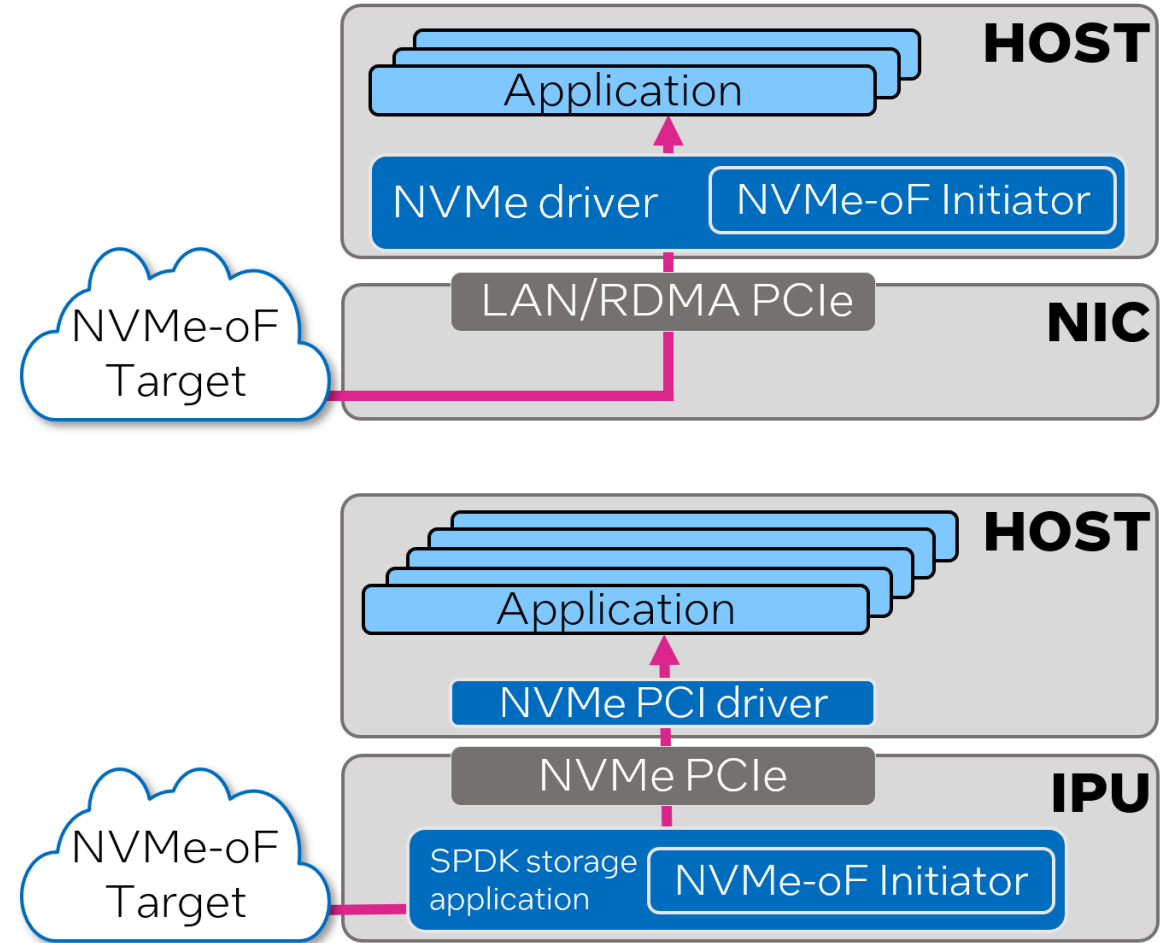
NVME Initiator - From standard NIC to IPU

- Standard NIC

- Host CPU cycles are consumed by NVMe-oF software stack
- NVMe-oF software stack is in Linux kernel or in user space (SPDK) on the host

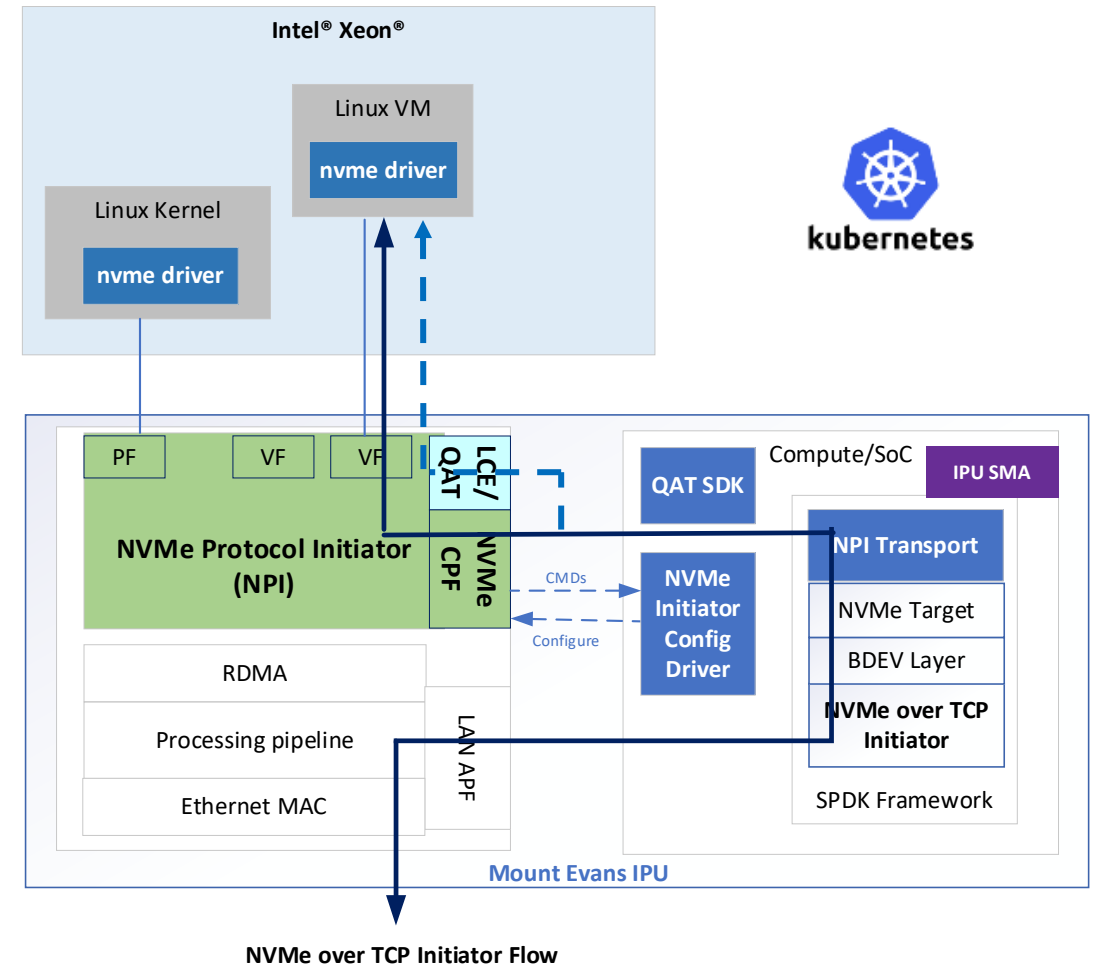
- IPU

- Frees host CPU for customer workload
- Host treats IPU as NVMe Controller connected over PCIe
- NVMe-oF is performed on IPU



IPU SW/HW components and their roles

- SPDK's modular design enables a clean integration with the IPU's NVMe initiator
- NPI (NVMe Protocol Initiator) Transport to interface with nvmf layer
- NPI Transport interacts with IPU SDK and API for device configurations and CMDs processing.
- QAT SDK interacts with NPI Transport and is responsible for data movement between host and IPU memory




SPDK Acceleration Framework (Accel)

Storage Performance Development Kit


- User-space Tools, Libraries, Drivers, & Applications
- Open Source & BSD Licensed
- Optimized for leading edge storage solutions
- SPDK is an ideal framework used for IPU/DPU based storage solutions.

Architecture

 **Target Protocols**

Network: NVMe-oF (RDMA, TCP, FC), iSCSI

Virtualization: vhost (scsi, blk), vfiio-user

 **Services**


Partitioning: Logical Volumes, GPT

Host FTL: ZNS

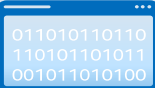
Accelerator Framework: DMA, Crypto, Compression, CRC32C

Caching: OCF

Pooling: RAID-0

 **Block Providers**

NVMe	Ceph RBD	Third Party
virtio	io_uring	malloc
	Linux AIO	null

 **Drivers**

NVMe: PCIe, RDMA, TCP

virtio: scsi, blk

Accelerators: DSA, Intel® QAT

Platform: VMD

Integrations

Orchestration
Cinder
Kubernetes

Database
RocksDB

SDS
DAOS, Ceph, Linstor

Virtualization
Qemu

Analyzers
VTune™

Tools

Benchmarking
fio
SPDK perf

Management
nvme-cli
spdk-cli

Diagnostic
spdk_top

Accel: Introduction

- Acceleration framework that already exists: https://spdk.io/doc/accel_fw.html
- Offloading SPDK compute heavy operations to HW accelerators
 - encrypt/decrypt, compress/decompress, CRC calculation and more ...
- HW accelerator examples
 - Intel® I/O Acceleration Technology (IOAT) engine
 - Intel® Data Streaming Accelerator (DSA) engine
- software plug-in module
 - Fallback path
 - E.g. ISA/L is used for optimized CRC32C calculation

Accel: Motivation for change

- **Current approach**
 - Only single operation is requested/performed on IO buffer
 - Many stages of IO buffer processing → many operations on accelerator
- **New requirements and possibilities**
 - New capabilities in HW accelerators: chaining operations on IO buffers
 - Operations scheduled in different software layers
 - HW offload possible: IPU Lookaside Crypto & Compression Engine (LCE)
- **Requested changes and outcome**
 - Introduce “chained operation” in Accel FW API
 - Adopt SPDK components to new HW capabilities through new API
 - Validate approach with the IPU LCE capabilities
- **Desired benefits**
 - Improved performance of system
 - Reduced pressure on memory bandwidth

Accel: new API and usage

■ Driver API

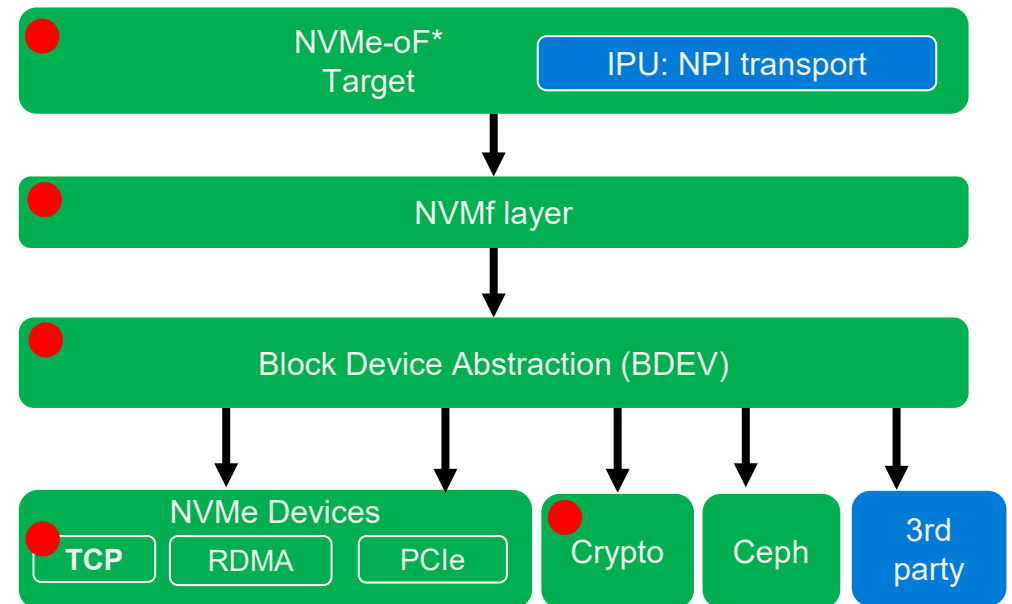
- Used to execute operations via hardware offload
- `spdk_accel_driver_opts`
 - `execute_sequence()`
 - `get_operation_info()`
- `spdk_accel_sequence_continue()`

■ User API

- Allows users to schedule operation and build operation chains
- `spdk_accel_append_encrypt()`,
`spdk_accel_append_copy()`
- `spdk_accel_sequence_reverse()`,
`spdk_accel_sequence_finish()`

● Engaged components

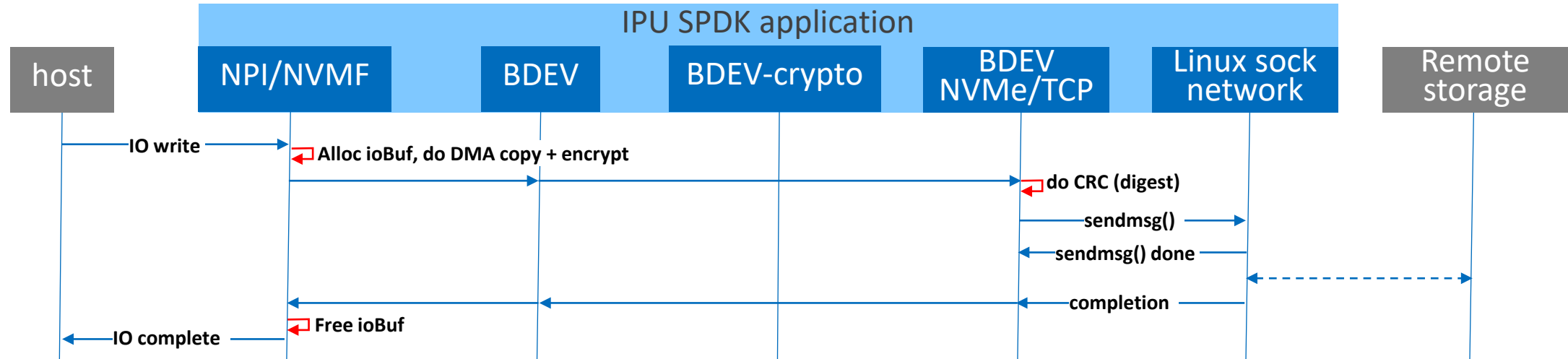
- bdev layer, NVMe-oF layer
- bdev NVMe, bdev crypto,
- NVMe/TCP transport



Accel: Optimized IO buffers

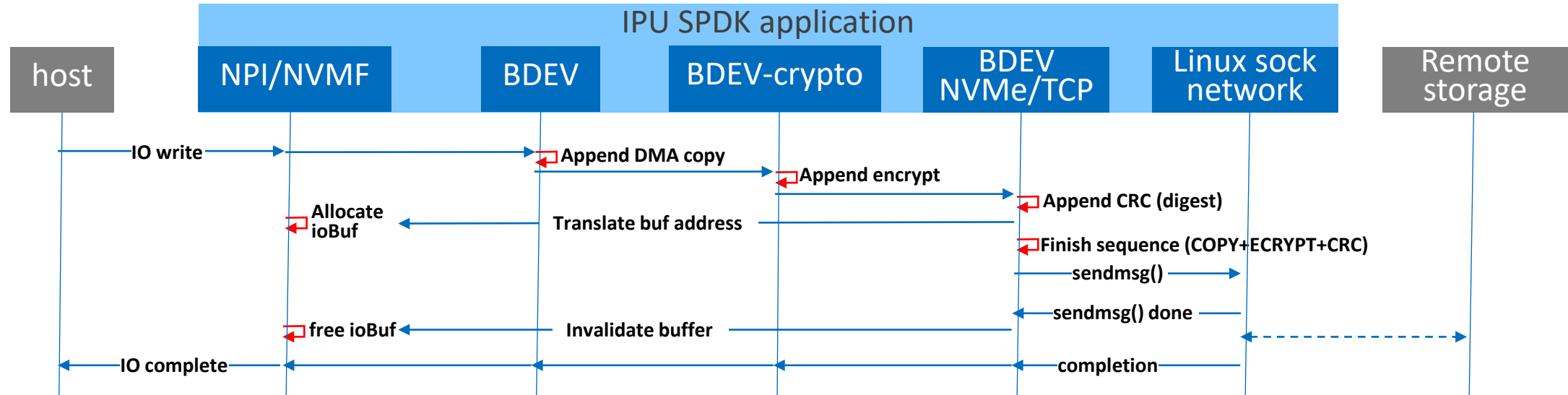
- Key innovations
 - Deferred buffer allocation
 - Important for IO reads, less vital for write operation
 - Introduced fake buffer mechanism
 - Faster buffer free
 - Important for IO writes, freed after submitting to network
 - Benefit: less pressure on memory size and usage
- Implemented in open-source SPDK core instead in vendor-specific internal code
 - Vendor-specific code is responsible only for programming vendor hardware

Accel: previous approach



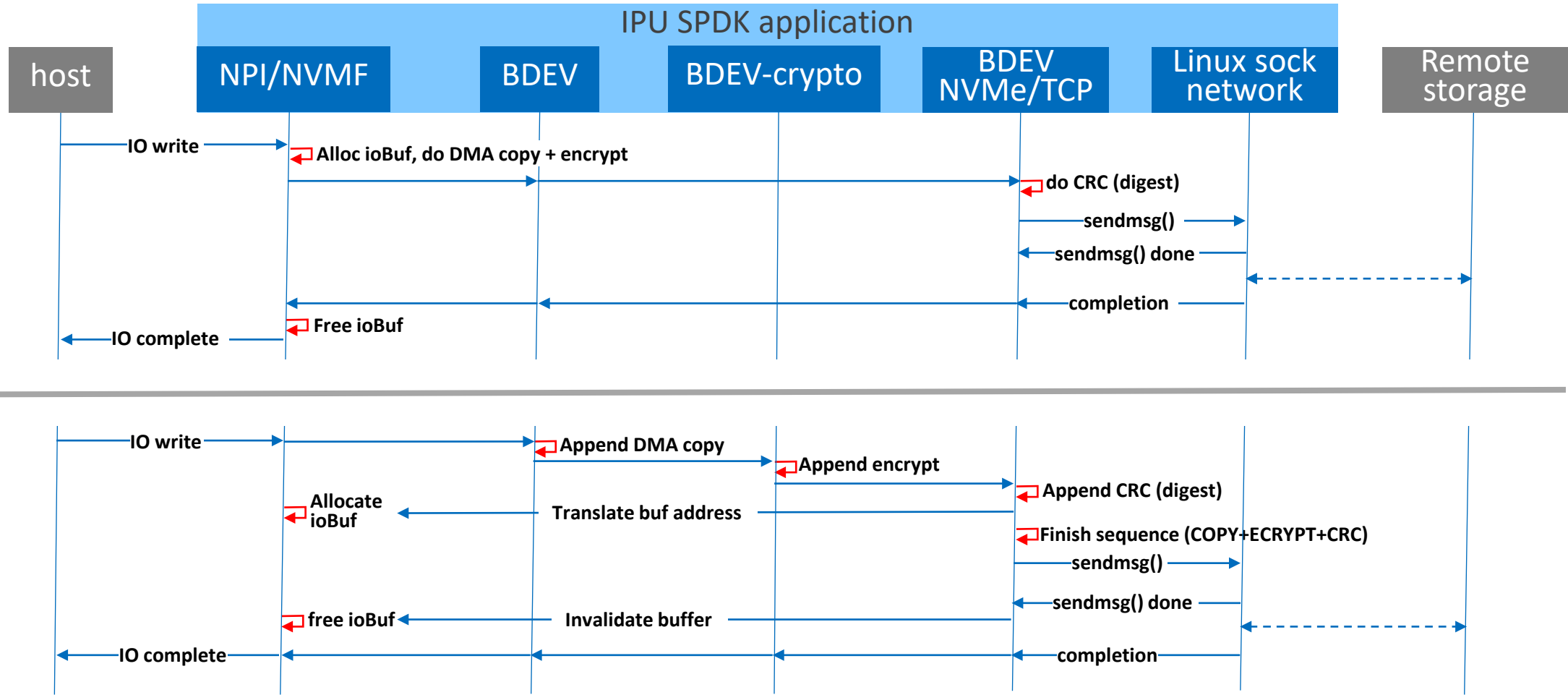
- Do dma + crypto and do CRC
 - two separate operations/two buffer translations
- „Do dma + crypto“
 - performed outside SPDK framework to reduce memory bandwidth cost
- Ineffective buffer management

Accel: new approach



- All buffer transformation requests are a single operation/transformation
- Managing this transformation perfectly fits into SPDK framework without additional performance penalty
- Delayed buffer allocation and early freeing of buffer

Accel: previous and new approaches, visually



Accel: current status

- **Already done in SPDK v24.05 release**
 - ACCEL API changes delivered
 - SPDK components adopted
 - Demonstrated with IPU that the new approach makes a sense

- **Planned for future releases**
 - More performance tuning in SPDK layers



Key Takeaways

Section Subtitle

Key Takeaways

- Upon completion, participant will be able to understand how the SPDK acceleration framework is structured and how is it used
- Upon completion, participant will be able to understand how the HW capabilities of IPU-like devices can be integrated to SPDK acceleration framework
- Upon completion, participant will be able to learn how the NVME initiator offload has been done for the Intel IPU, and what are the benefits of using the SPDK framework and the IPU HW capabilities together



Please take a moment to rate this session.

Your feedback is important to us.

Later slides are backup to remove

Section Subtitle

Section Title

Section Subtitle



Section Title

Section Subtitle

Light Slide Title

- Bullets 1
 - Bullets 2
 - Bullets 3
 - Bullets 4
 - Bullets 5

Dark Slide Title

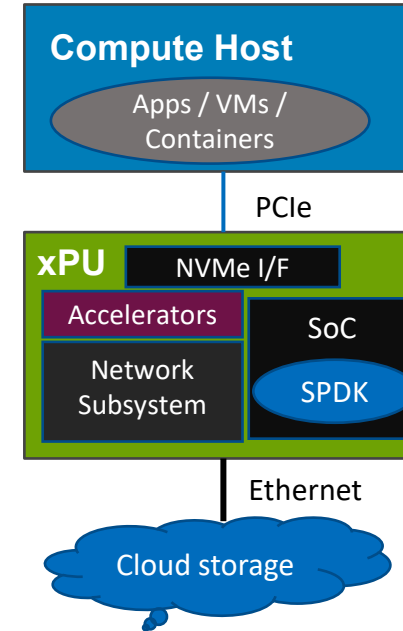
- Bullets 1
 - Bullets 2
 - Bullets 3
 - Bullets 4
 - Bullets 5

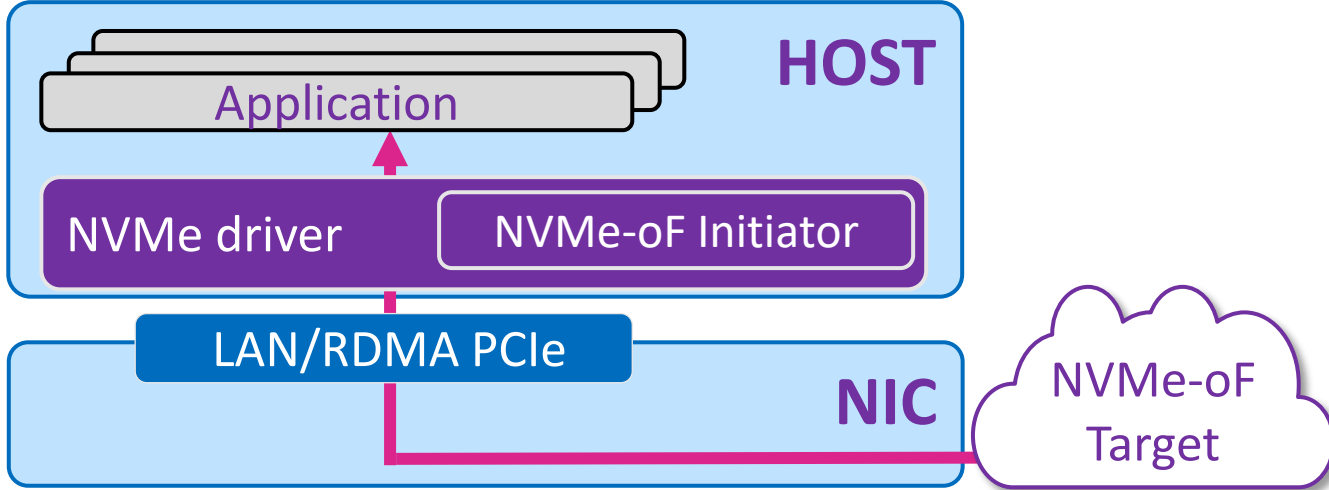


Please take a moment to rate this session.

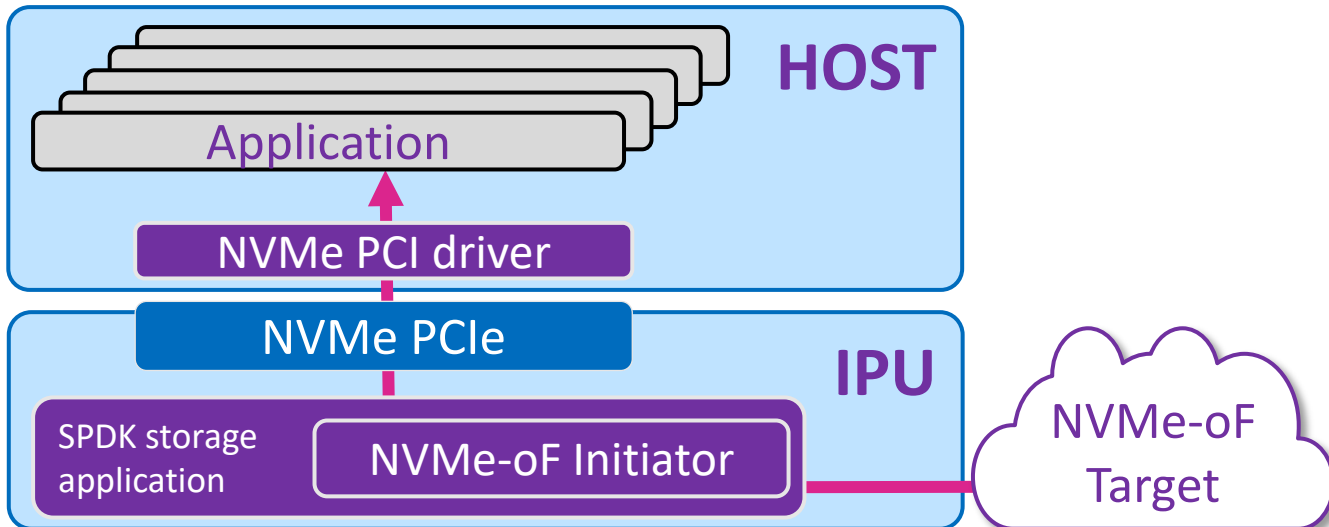
Your feedback is important to us.

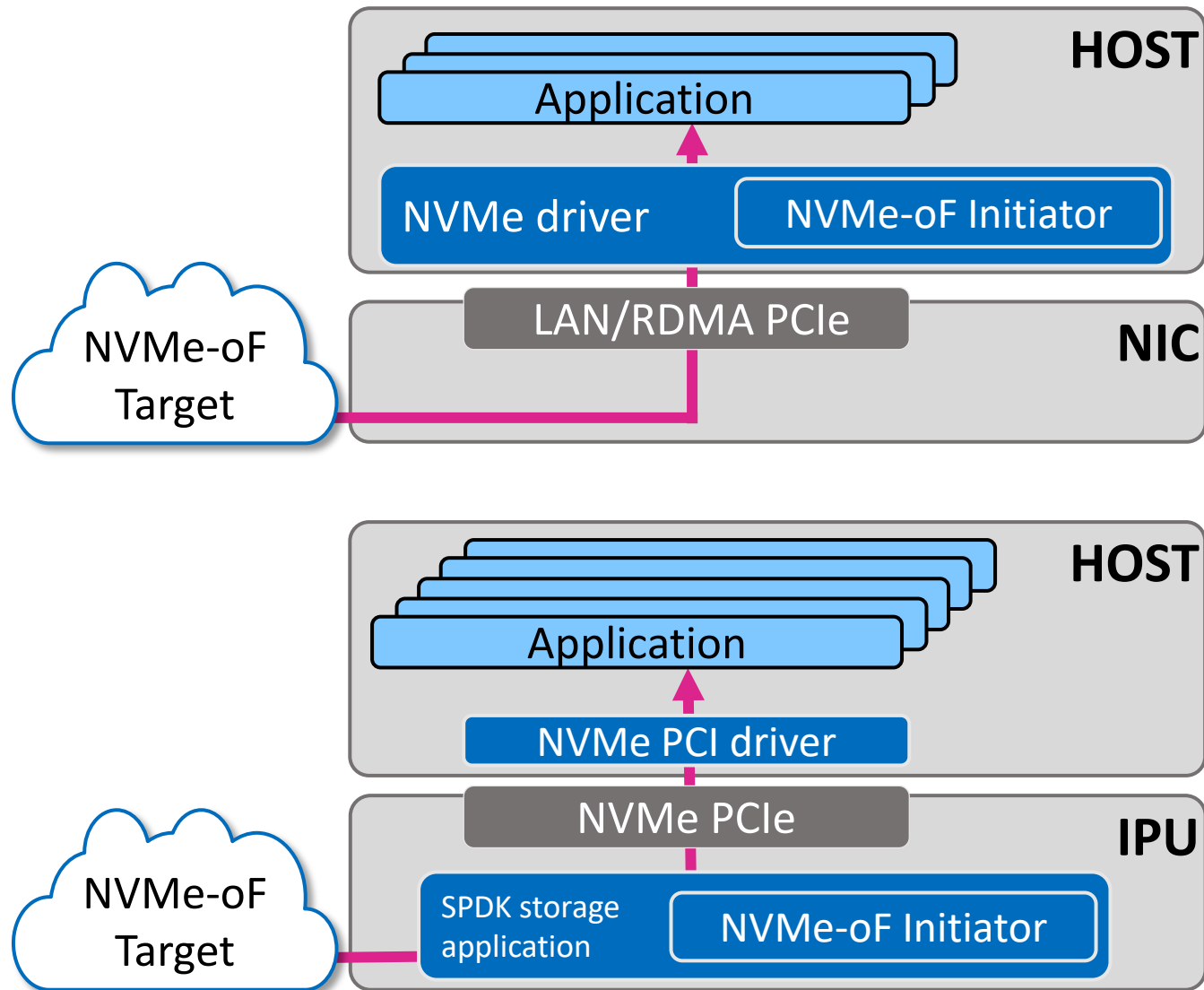
xPU: NVMe-oF Initiator Usage (for zoom figure)



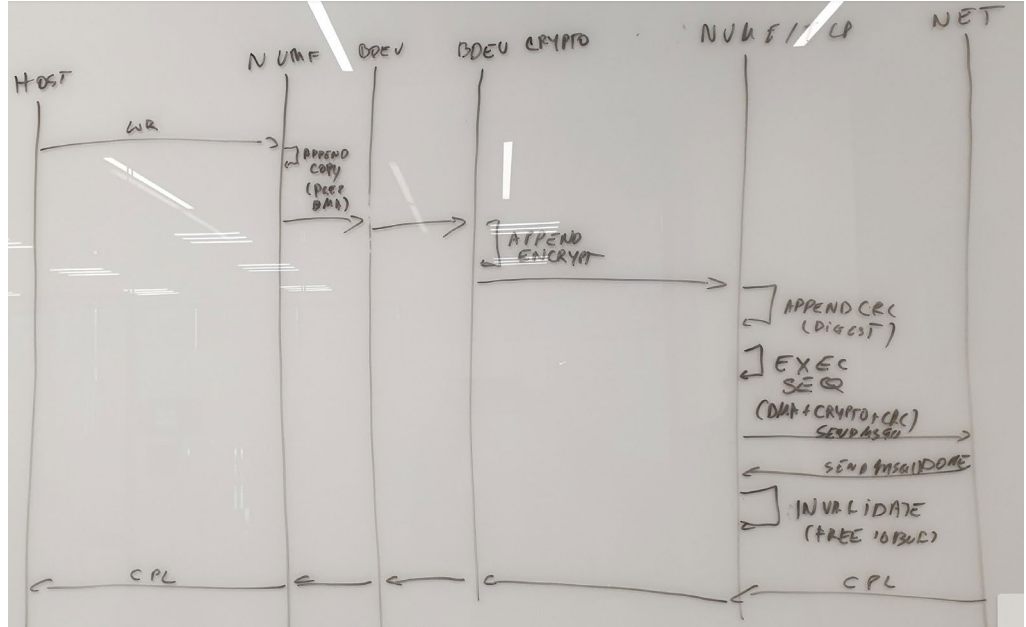
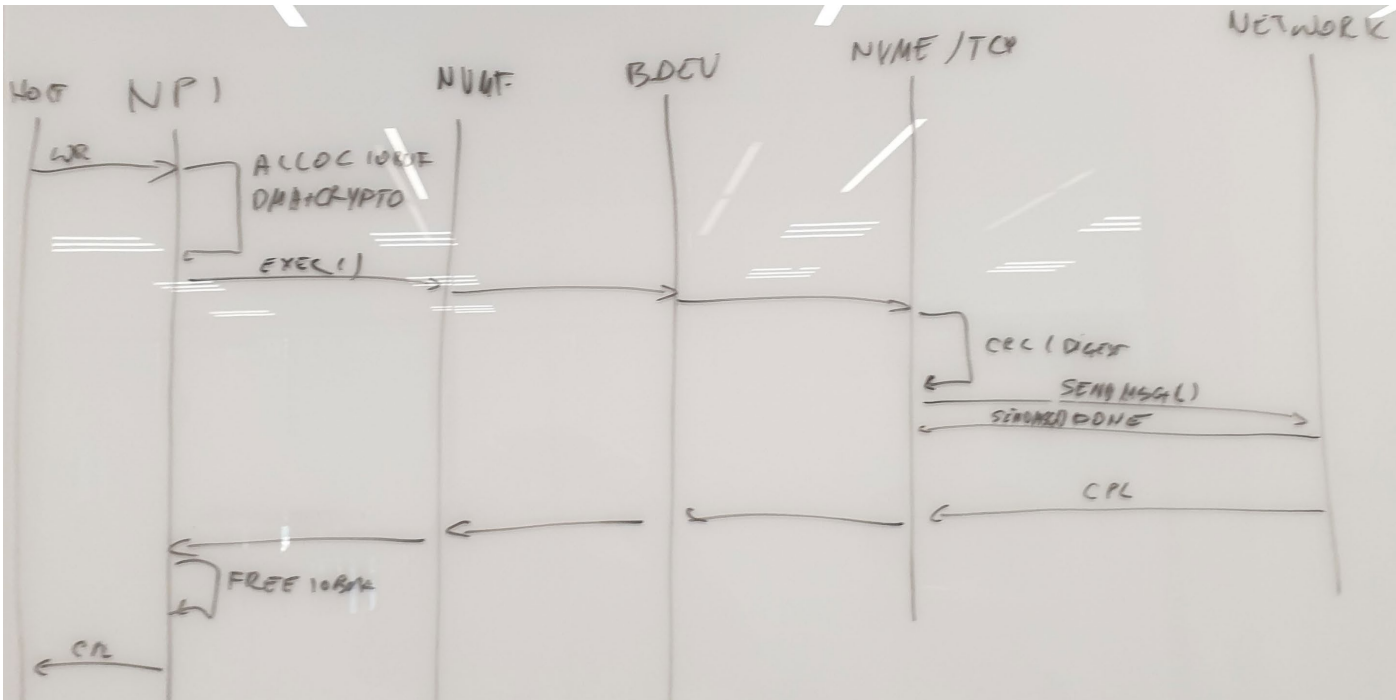


■ X





ACCEL FW: previous and new approaches



Previous approach

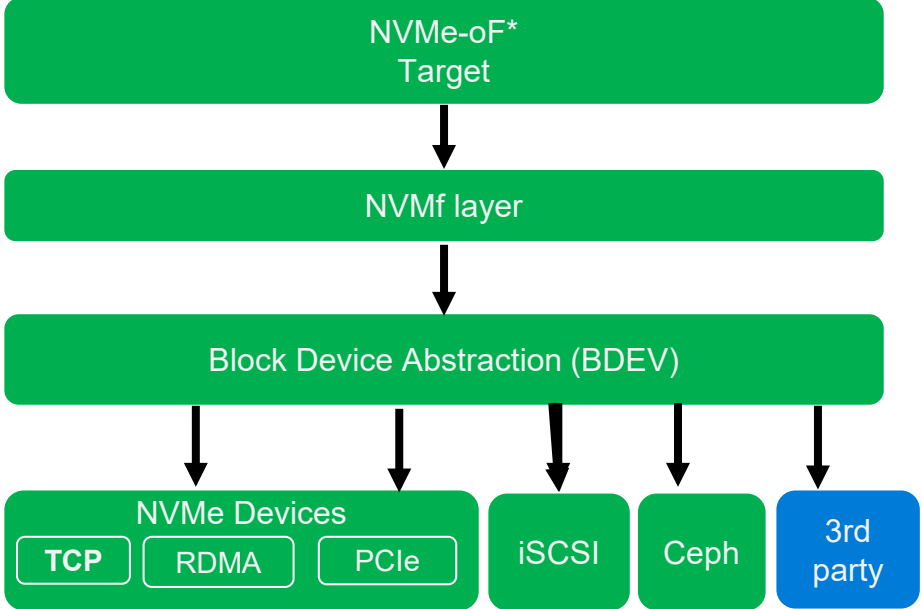
new approach

TBD: alternative slide to previous one.
Decide which of them work better for us.

SPDK NVMe-oF Software Stack

- Storage Performance Development Kit (SPDK) provides a set of tools and libs for writing high performance, scalable, user-mode storage applications.
- SPDK provides a flexible architecture for supporting a choice of protocols and enables customizations.
- SPDK is an ideal framework used for IPU/DPU based storage solutions.

SPDK Components for NVMe-oF solutions.
For details, please refer to <https://spdk.io/>



Accel: challenges and innovations

- Type of challenges
 - Collecting operation chains across all software layers
 - Submit all chains as single operation to execute them together in one shot
 - Address differences for IO read/write: each of them performs steps in different order/moment in the SW stack
- Optimized IO buffers usage model
 - Deferred buffer allocation
 - Important for IO reads, less vital for write operation
 - Introduced fake buffer mechanism
 - Faster buffer free:
 - Important for IO writes, freed after submitting to network
 - Benefit: less pressure on memory size usage
- The more innovations in open-source SPDK core instead in vendor-specific internal code
 - Vendor specific code is responsible only for programming vendor's hardware