STORAGE DEVELOPER CONFERENCE
SDC21
BY Developers FOR Developers

Virtual Conference
September 28-29, 2021

Intel Mount Evans IPU Based NVMe/TCP Initiator Offload

Presented by
Ziye Yang, Staff Software Engineer, **Intel**
Yadong Li, Principal Engineer, **Intel**
Sudheer Mogilappagari, Software Engineer, **Intel**

Agenda

- Brief Introduction of IPU
- Intel Mount Evans IPU Introduction
- SPDK NVMe/TCP Initiator Design
- SPDK NVMe/TCP Initiator Performance Evaluation with Intel® Ethernet 800 Series with Application Device Queues (ADQ)
- Summary

Brief Introduction of IPU

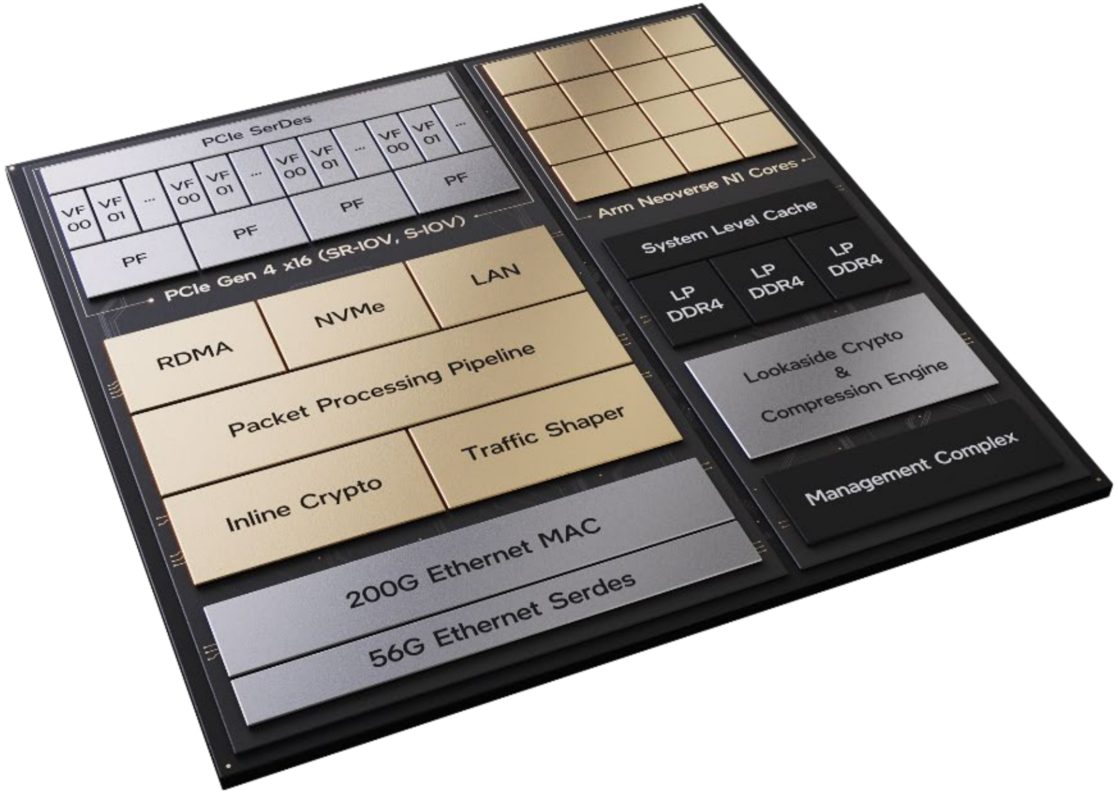
Brief Introduction of IPU

- Infrastructure Processing Unit (IPU) is an evolution of SmartNIC, provides full infrastructure offload.
  - Highly intelligent infrastructure acceleration. Minimize latency and jitter and maximize revenue from CPU.
  - Separation of Infrastructure & Tenant, system-level security, control and isolation.
  - HW and SW programmable, built to customer needs.
- In this topic, we will briefly introduce Intel's ASIC based IPU – Mount Evans, and then describe NVMe over TCP Initiator implementation as an example for IPU based storage offload.

Intel Mount Evans IPU Introduction

Mount Evans

Intel's 200G IPU



Hyperscale Ready

- Co-designed with a top cloud provider
- Integrated learnings from multiple gen. of FPGA sNIC/IPU
- High performance under real world load
- Security and isolation from the ground up

Technology Innovation

- Highly Programmable Packet Processing Engine
- NVMe storage interface scaled up from Intel Optane Tech
- Next Generation Reliable Transport
- Advanced crypto and compression accel.

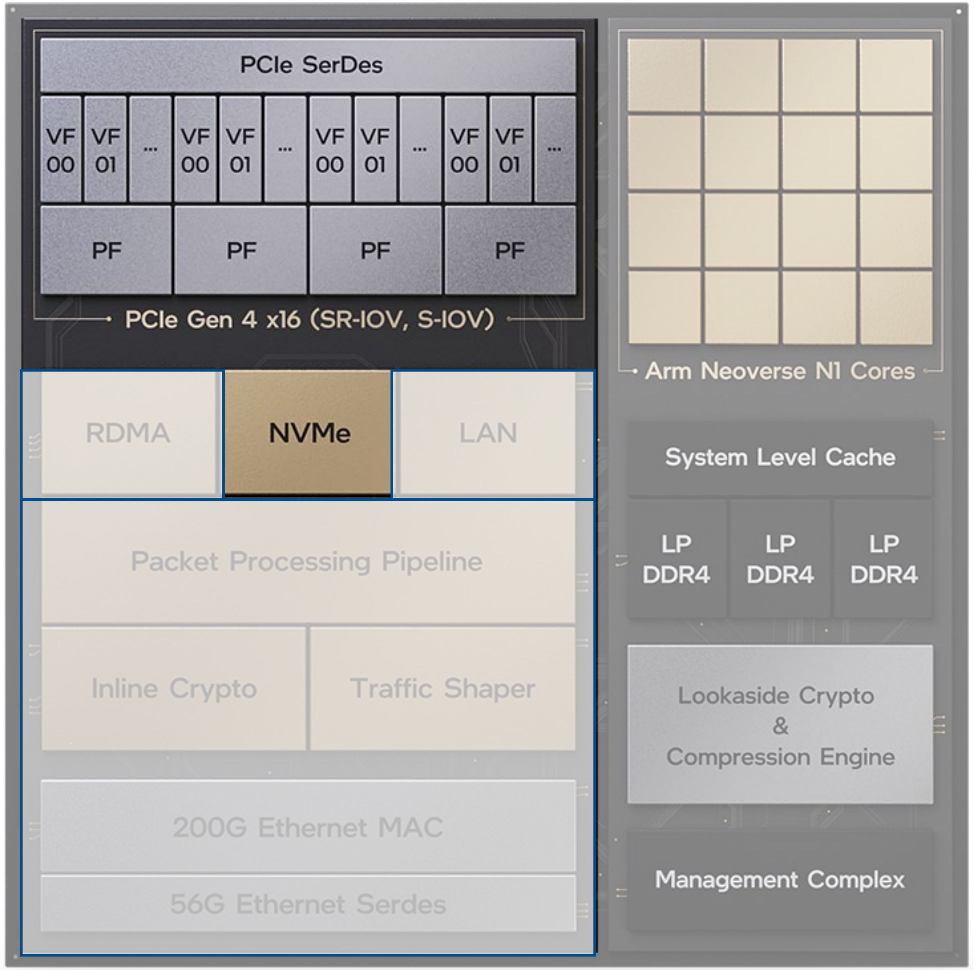
Software

- SW/HW/Accel co-design
- P4 Studio based on Barefoot
- Leverage and extend DPDK and SPDK
- Enable broad adoption of IPUs

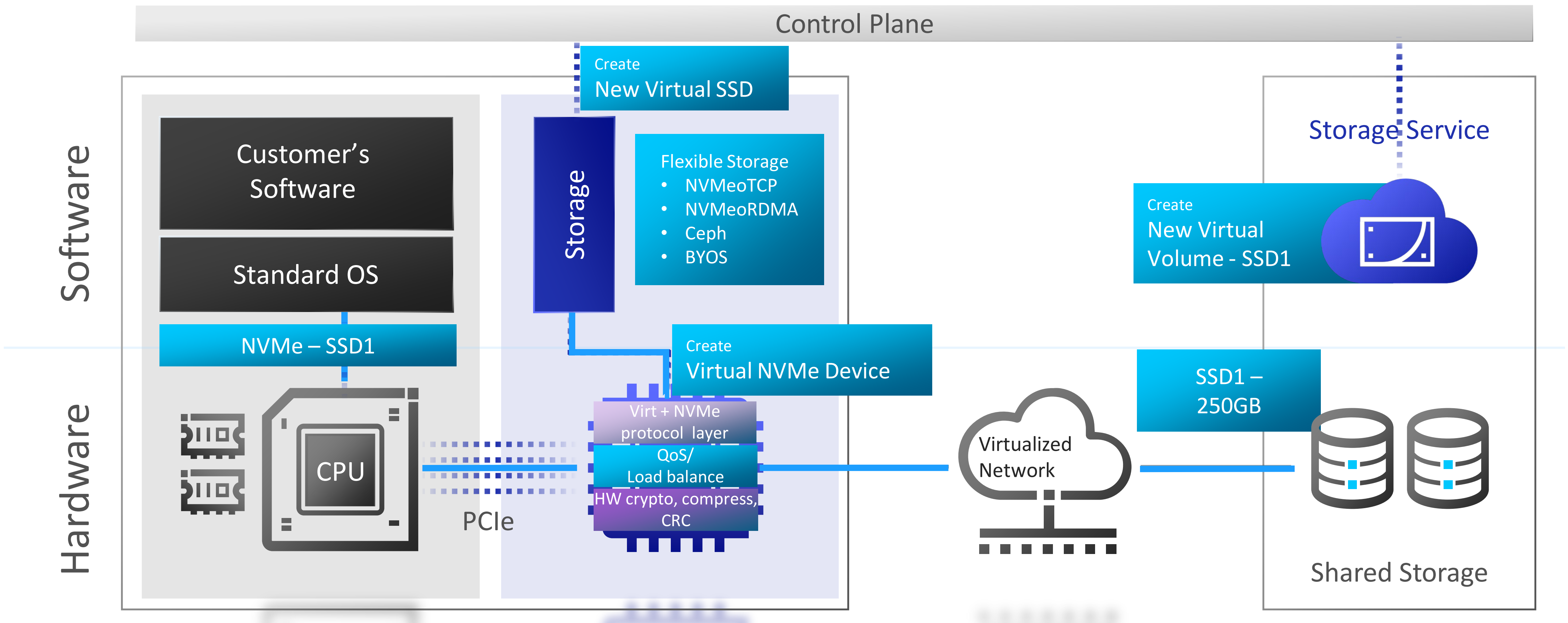
Mount Evans

NVMe Initiator

- Multi-Host support (4 PFs for Hosts)
- SR-IOV support
- Data-at-rest crypto (AES-XTS)
- IOPS and BW limiting
- LCE for lookaside crypto, compression and CRC offloads, support chained ops
- Support local attached storage (4 PCIe root ports)



Scale out Storage Architecture

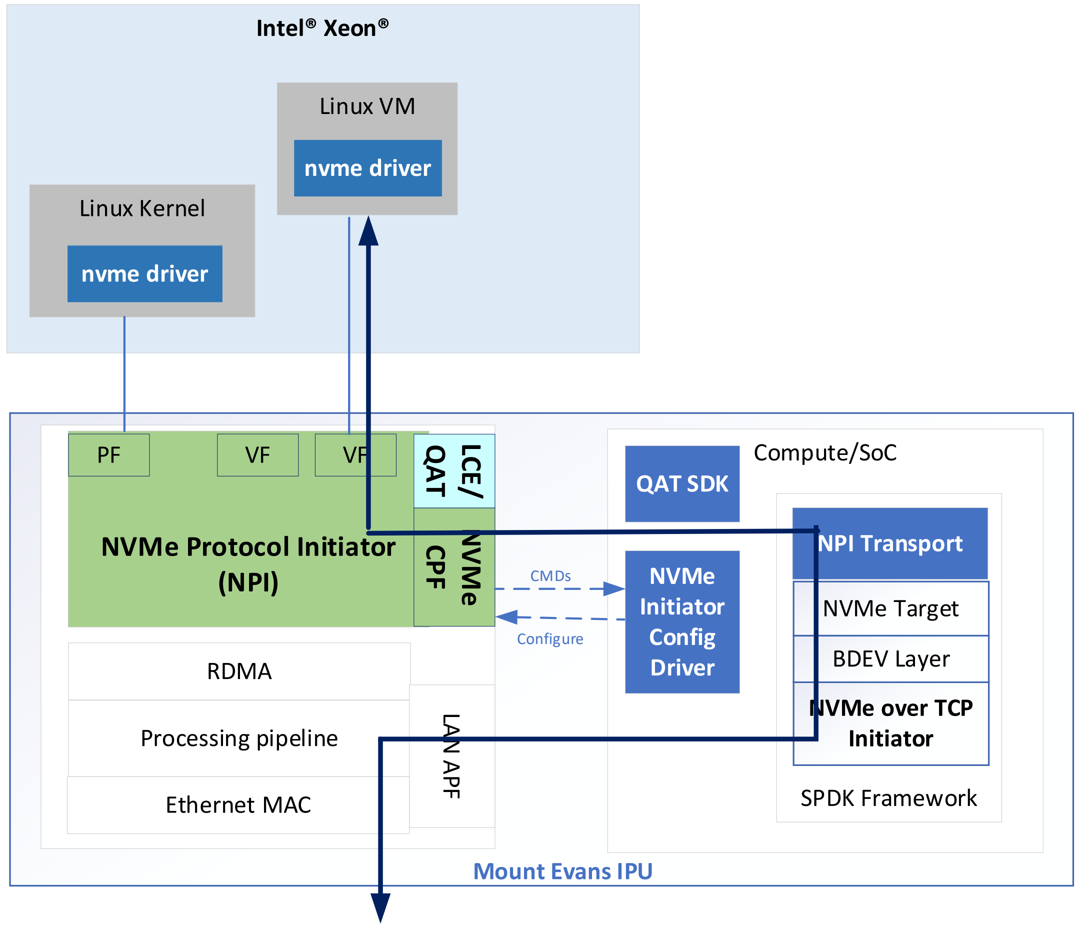


 |

STORAGE DEVELOPER CONFERENCE SDC 21

Mount Evans Based NVMe over TCP Initiator

- NVMe Protocol Initiator (NPI) HW auto fetches NVMe CMDs into SoC memory.
- SoC SW uses LCE/QAT DMA engine to fetch PRP lists and move data payloads.
- LCE/QAT used for CRC, crypto and compression offloads. Support chained ops.
- Fully integrated with SPDK NVMe-oF SW stack
  - NPI Transport for integration with nvmf layer.
  - NPI Transport uses NVMe Initiator Config Driver for device configurations and CMDs processing.



NVMe over TCP Initiator Flow

SPDK NVMe over TCP Initiator Introduction

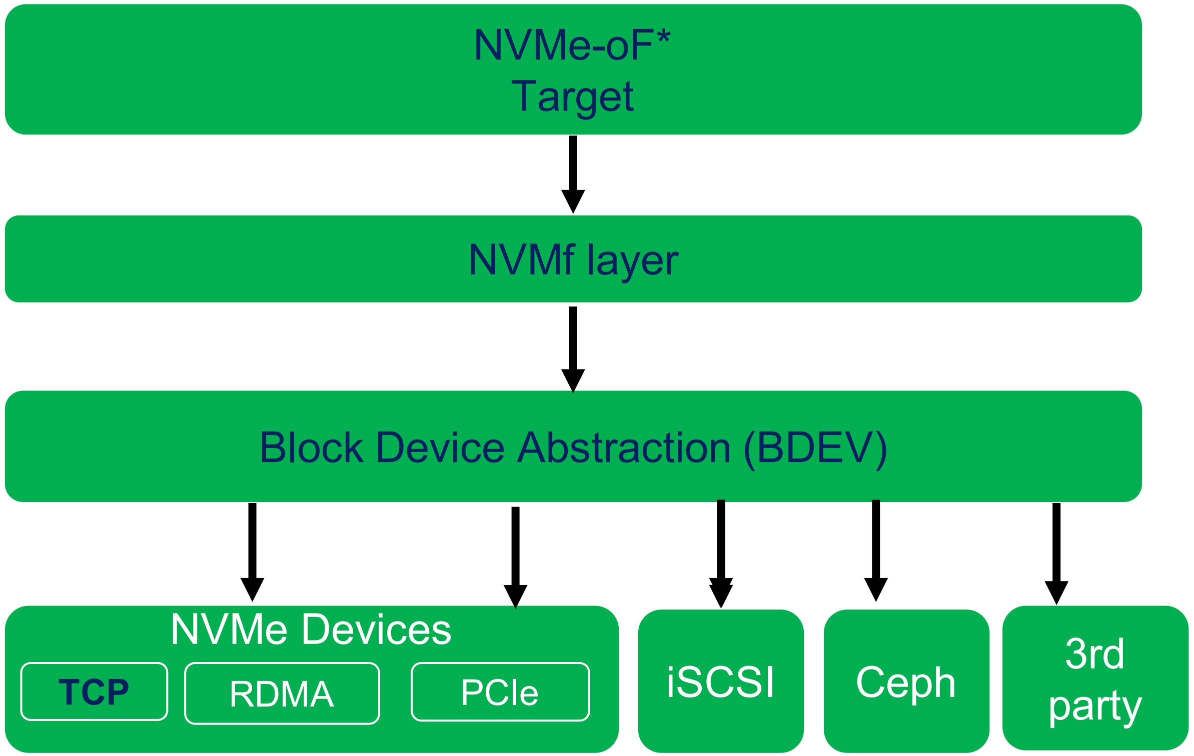
SPDK NVMe over TCP Initiator Introduction

Following topics will be introduced in this section:

- SPDK quick overview with NVMe-oF related components.
- Detailed information of user space NVMe over TCP initiator.
- How Application Device Queues (ADQ) Technology is used in SPDK NVMe over TCP initiator.

SPDK Architecture (NVMe-oF view)

https://SPDK.IO
https://SPDK.IO/CN
Main Web Presence

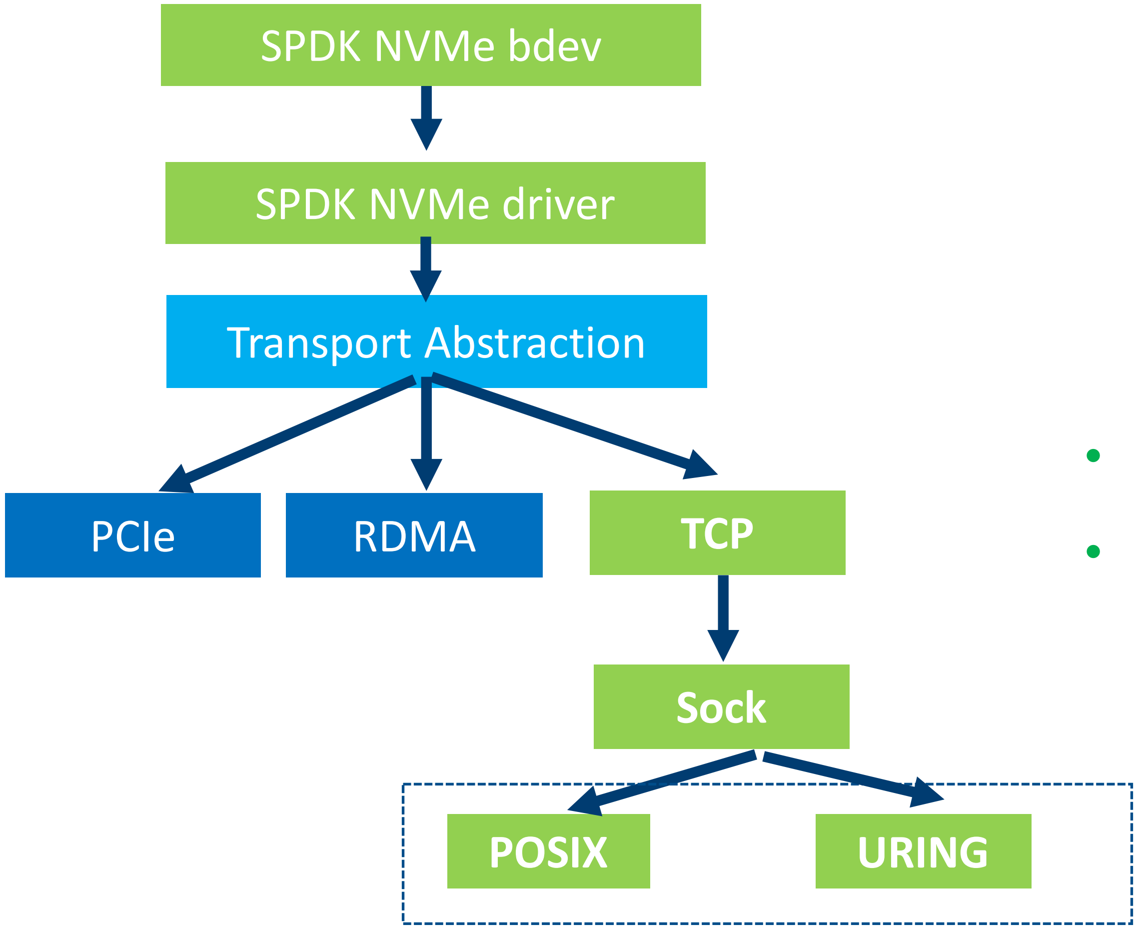


IPU storage initiator SW for NVMe-oF TCP can be integrated with SPDK at

- Nvmf layer
- bdev layer
- NVMe tcp driver

SPDK NVMe over TCP Initiator

It is part of SPDK user space NVMe-oF solution for the initiator side based on user space NVMe device drivers, socks etc.



- POSIX (Stable, no dependency on kernel)
- Uring (Requires Linux kernel > 5.4.3, currently it is experimental)

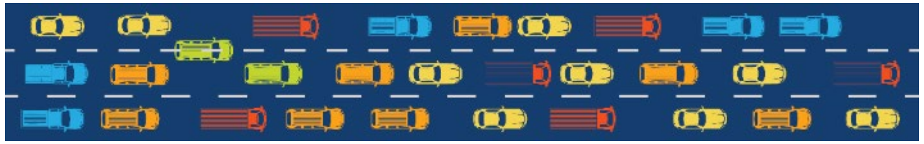
Intel® Ethernet 800 Series with Application Device Queues (ADQ)

ADQ is an open technology designed to improve application specific queuing and steering.

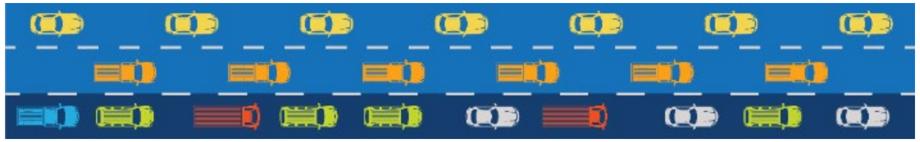
ADQ works by:

- Filtering application traffic to a dedicated set of queues
- Application threads of execution are connected to specific queues within the ADQ queue set
- Bandwidth control of application egress (Tx) network traffic

Without ADQ
Application traffic intermixed with other traffic types



With ADQ
Application traffic to a dedicated set of queues



ADQ benefits:

INCREASES APPLICATION PREDICTABILITY

REDUCES APPLICATION LATENCY

IMPROVES APPLICATION THROUGHPUT

ADQ Enabling in SPDK Socket Layer

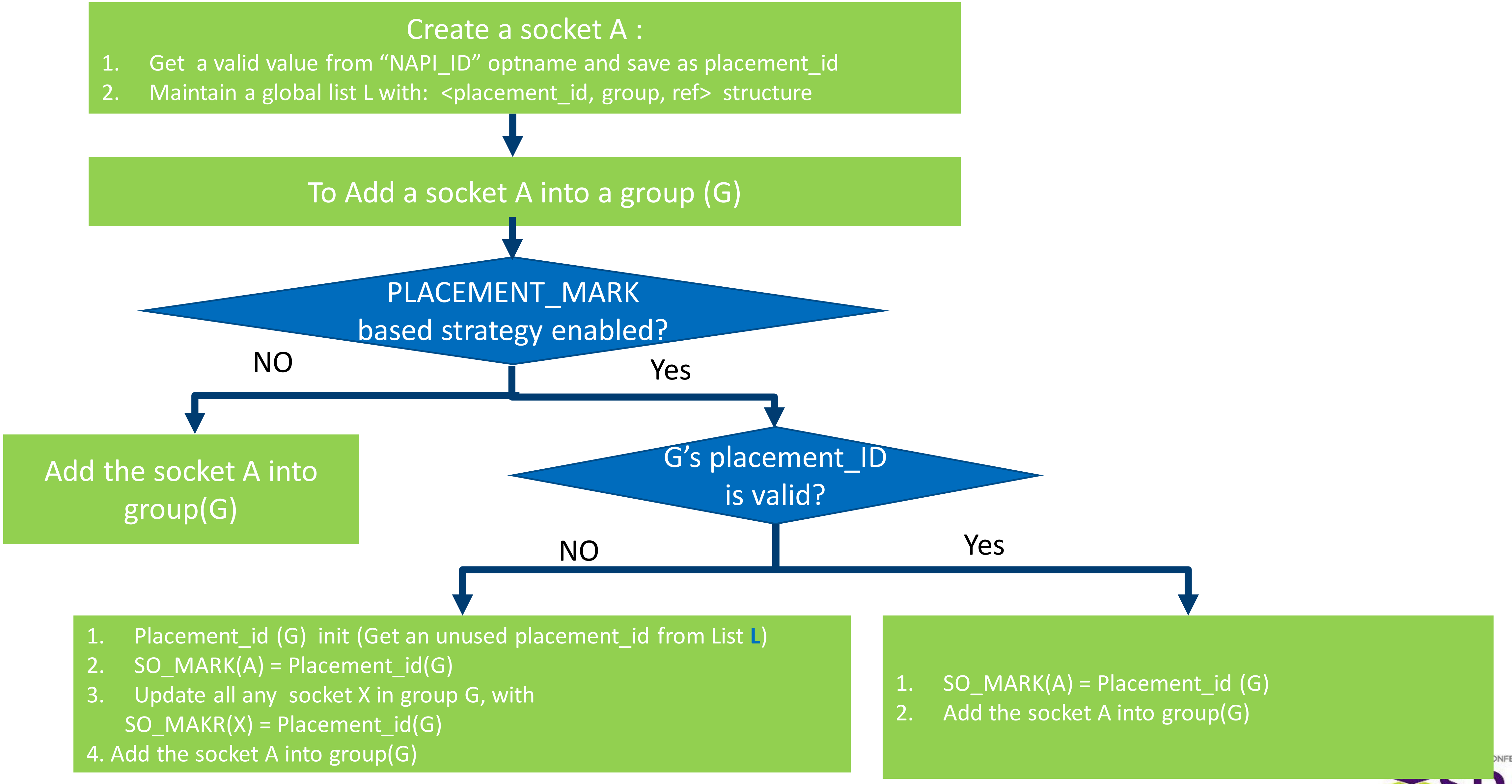
SPDK sock library support

- Common support for both server & client side.
  - Implement a function to get NAPI_ID for the socket file descriptor(FD).
    - Through *getsockopt* function on optname "**SO_INCOMING_NAPI_ID**"
- Support for server side
  - In order to address the efficiency usage issue on ADQ inside a SPDK application which is started with **multiple CPU cores**.
    - Implement a function to get optimal socket polling group for each socket FD.
    - Then the upper layer can create a scheduler to distribute socket FD with the same NAPI_IDs into the same socket polling group to better leverage the ADQ features.

ADQ Enabling in SPDK Socket Layer to Support Initiator Usage

- Challenges:
  - Socket is already created on a dedicated SPDK_thread on the initiator side, so cannot use the same methodology on the server side, i.e., Seek an optimal socket polling group and schedule the socket into it.
    - We need leverage **spdk_thread_send_msg** among spdk_threads to do socket I/O and there is **performance penalty**.
- Solutions:
  - Leverage "*NAPI_ID & SO_MARK*" optnames on the socket. The purpose is to enforce all the sockets in each polling group with the same value for SO_MARK optname.
  - Then the NIC driver (with ADQ enabled) in Linux kernel queues packets of all sockets with same value of SO_MARK optname on same hardware queue.

Socket and Group Management: Add a socket into a group

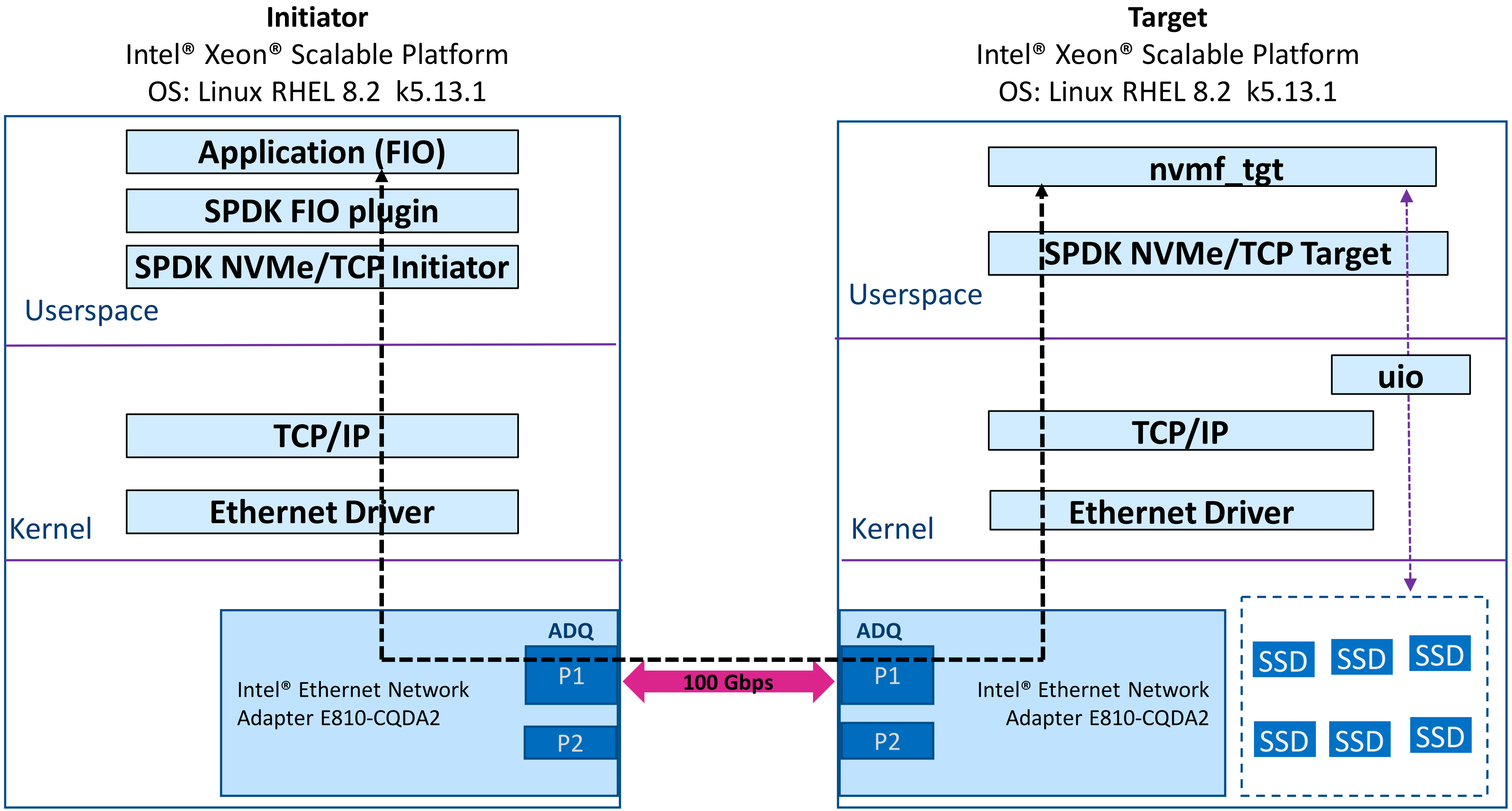


SPDK NVMe/TCP Initiator Performance Evaluation on Intel® Ethernet 800 Series with Application Device Queues (ADQ)

* For SPDK NVMe/TCP Target performance, please refer to SDC'20 presentation – Optimizing User Space NVMe-oF TCP Transport Solution

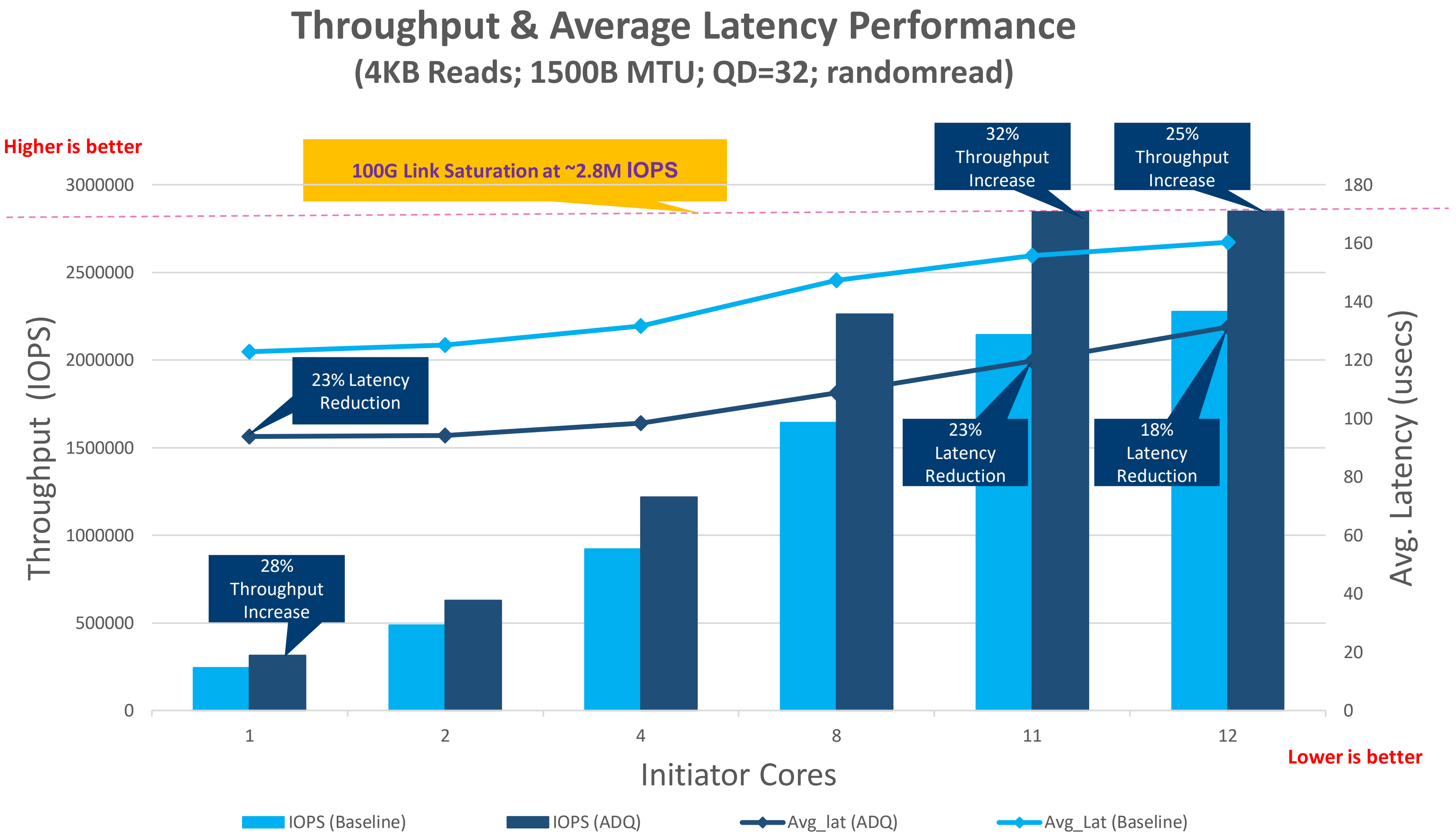
SPDK NVMe/TCP with ADQ Test Topology

Intel® Ethernet Network Adapter E810-CQDA2 with ADQ



SPDK NVMe/TCP Initiator Read Performance

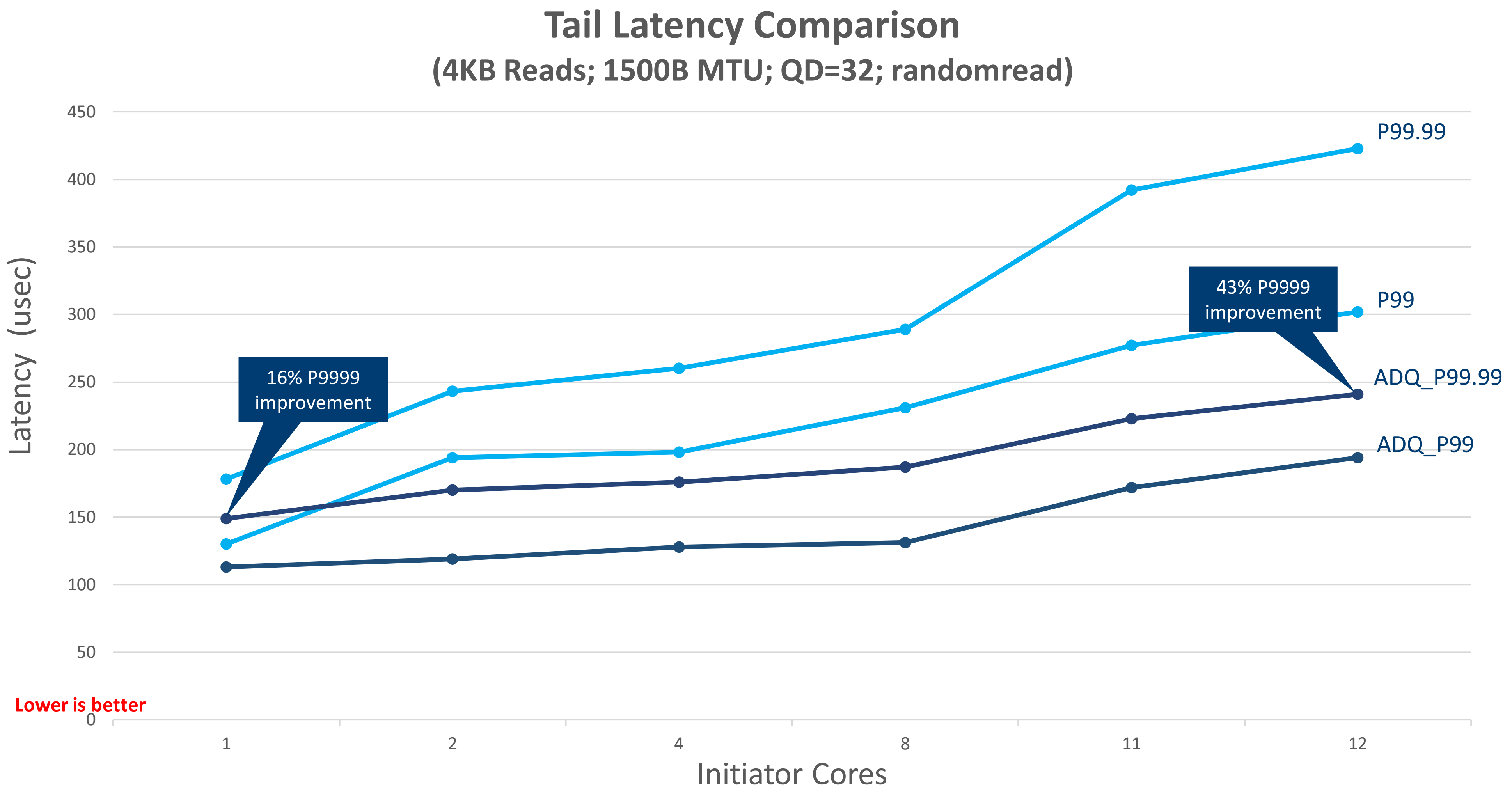
Intel® Ethernet Network Adapter E810-CQDA2 with ADQ



SEE APPENDIX FOR TESTING CONFIGURATION. Results may vary. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks. See configuration disclosure for details. For more information regarding performance and optimization choices in Intel software products., please visit https://software.intel.com/en-us/articles/optimization-notice.

SPDK NVMe/TCP Initiator Tail Latency

Intel® Ethernet Network Adapter E810-CQDA2 with ADQ



SEE APPENDIX FOR TESTING CONFIGURATION. Results may vary. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks. See configuration disclosure for details. For more information regarding performance and optimization choices in Intel software products., please visit https://software.intel.com/en-us/articles/optimization-notice.

Summary

Summary

- IPU provides efficient infrastructure offload and improves data center efficiency. Designed to help minimize latency and jitter and maximize revenue from CPU.
- In this session, we shared
  - Intel Mount Evans IPU based NVMe/TCP initiator offload.
  - Integration of SPDK NVMe/TCP initiator into Mount Evans SW stack.
  - Tests with Intel® Ethernet 800 Series Network Adapter showed significant performance improvement with ADQ enabled for NVMe/TCP initiator.
    - Integration of Intel ADQ feature provides a low latency and high performance SPDK NVMe/TCP initiator solution.
- Finally, we welcome any SPDK optimizations and enhancements to make it best for IPU storage solutions.

Legal Information

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex. Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup 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.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps. All product plans and roadmaps are subject to change without notice.

Statements in this document that refer to future plans or expectations are forward-looking statements. These statements are based on current expectations and involve many risks and uncertainties that could cause actual results to differ materially from those expressed or implied in such statements. For more information on the factors that could cause actual results to differ materially, see our most recent earnings release and SEC filings at www.intc.com.

Q & A

Appendix: Configuration

SPDK NVMe/TCP with ADQ Testing Configuration

	Initiator	Target
Test by	Intel	Intel
Test date	9/9/2021	9/09/2021
Platform	Sawtooth Pass (S2600STQ)	Sawtooth Pass (S2600STB)
# Nodes	1	1
# Sockets	2	2
CPU	Intel(R) Xeon(R) Platinum 8280L CPU @ 2.70GHz	Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz
Cores/socket, Threads/socket	28 cores per socket, 56 threads per socket	28 cores per socket, 56 threads per socket
ucode	0x5003102	0x2006b06
HT	Disabled	Disabled*
Turbo	Enabled	Enabled
BIOS version	SE5C620.86B.02.01.0008.031920191559	SE5C620.86B.02.01.0008.031920191559
System DDR Mem Config: slots / cap / run-speed	16 slots / 16GB / 2666 MT/s	12 slots / 16GB / 2666 MT/s
System DCPMM Config: slots / cap / run-speed	N/A	N/A
Total Memory/Node (DDR+DCPMM)	128GB DDR4-2666 DIMM	96GB DDR4-2666 DIMM
Storage - boot	100GB SATA SSD	100GB SATA SSD
Storage - application drives	N/A	6x Intel Corporation Optane SSD 900P Series, PCIe 3.0, x4
NIC	Intel® Ethernet Network Adapter E810-CQDA2	Intel® Ethernet Network Adapter E810-CQDA2
PCH	Intel Corporation C620 Series Chipset Family	Intel Corporation C620 Series Chipset Family
Other HW (Accelerator)	N/A	N/A
OS	Red Hat Enterprise Linux 8.2 (Ootpa)	Red Hat Enterprise Linux 8.2 (Ootpa)
Kernel	5.13.1	5.13.1
Workload & version	Fio 3.25; SPDK 21.07	SPDK 21.01
Compiler	GCC 8.3.1 20191121 (Red Hat 8.3.1-5)	GCC 8.3.1 20191121 (Red Hat 8.3.1-5)
NIC Driver	ice-1.6.8; FW: 0x80008125	ice-1.6.8; FW: 0x80008256

*HT was turned off for benchmark purposes to not schedule FIO on threads of same physical core

SPDK NVMe/TCP with ADQ: OS and Network Adapter Configuration

Red Hat* Enterprise Linux 8.2 Configuration

```
stopped: irqbalance, cpupower, firewalld, SELINUX disabled
tuned-adm profile throughput-performance
Kernel 5.13.1
sysctl -w net.core.somaxconn=4096
sysctl -w net.core.netdev_max_backlog=8192
sysctl -w net.ipv4.tcp_max_syn_backlog=16384
sysctl -w net.core.rmem_max=16777216
sysctl -w net.core.wmem_max=16777216
sysctl -w net.ipv4.tcp_mem="764688 1019584 16777216"
sysctl -w net.ipv4.tcp_rmem="8192 87380 16777216"
sysctl -w net.ipv4.tcp_wmem="8192 65536 16777216"
sysctl -w net.ipv4.route.flush=1
sysctl -w vm.overcommit_memory=1
x86_energy_perf_policy performance
systemctl stop irqbalance
```

Network Adapter Configuration

ADQ “On”

```
ethtool –coalesce <interface> adaptive-rx off rx-usecs 0
 ethtool –coalesce <interface> adaptive-tx off tx-usecs 500
 ethtool --set-priv-flags <interface> channel-inline-flow-director on
 ethtool --set-priv-flags <interface> channel-pkt-clean-bp-stop on
 ethtool --set-priv-flags <interface> channel-pkt-clean-bp-stop-cfg on
 ethtool --offload <interface> hw-tc-offload on
 sysctl -w net.core.busy_read=1
 sysctl -w net.core.busy_poll=1
 <path-to-ice>/scripts/set_irq_affinity –X local <interface>
 <path-to-ice>/scripts/set_xps_rxqs <interface>
 tc qdisc add dev <interface> root mqprio num_tc 2 map 0 1 queues 2@0 $adq_qs@2 \
    hw 1 mode channel
 tc qdisc add dev <interface> ingress
 tc filter add dev <interface> protocol ip parent ffff: prio 1 flower dst_ip $addr/32 \
    ip_proto tcp dst_port $((port)) skip_sw hw_tc 1 [on Target system]
 tc filter add dev <interface> protocol ip parent ffff: prio 1 flower dst_ip $addr/32 \
    ip_proto tcp src_port $((port)) skip_sw hw_tc 1  [on Initiator system]
```

ADQ “Off” (Baseline)

```
 ethtool –coalesce <interface> adaptive-rx off rx-usecs 50
 ethtool –coalesce <interface> adaptive-tx off tx-usecs 50
 sysctl -w net.core.busy_read=0
 sysctl -w net.core.busy_poll=0
 <path-to-ice>/scripts/set_irq_affinity –X local <interface>
 ethtool -L <interface> rx $queues tx $queues
```


Please take a moment to rate this session.

Your feedback is important to us.

