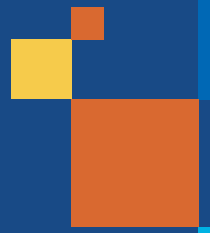
Storage Performance Development Kit (SPDK)
Persistent Memory Development Kit (PMDK)
Intel® VTune™ Profiler

Virtual Forum

Optimizing user space NVMe-oF TCP transport solution with both software and hardware methodologies

Ziye Yang,

Cloud software engineer, Intel



intel®

Notices and Disclaimers

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to www.intel.com/benchmarks.

Performance results are based on testing as of date disclosed in the system configuration and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at www.intel.com.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804.

Intel, the Intel logo, Intel Atom®, Xeon and Xeon logos, are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© 2020 Intel Corporation.

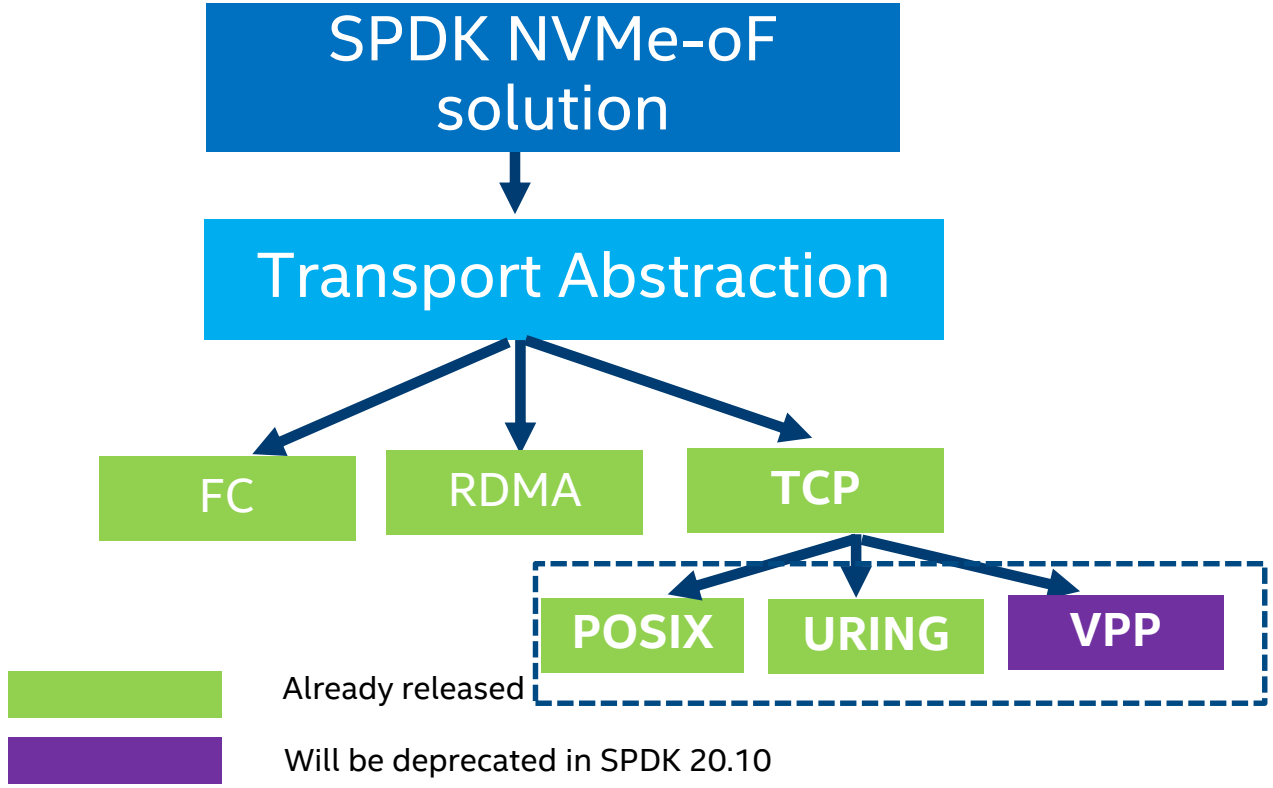
Agenda

- Introduction to SPDK user space NVMe-oF TCP solution
- Software optimization based on Kernel TCP/IP and io_uring
- SPDK NVMe/TCP acceleration using Intel® Ethernet 800 Series with Application Device Queues (ADQ)
- SPDK NVMe/TCP performance evaluation with ADQ
- Conclusion

Introduction to SPDK user space NVMe-oF TCP solution

SPDK NVMe-oF TCP introduction

It is part of SPDK user space NVMe-oF solution based on SPDK application framework, user space NVMe device drivers etc.



- POSIX (Stable, no dependency on kernel)
- Uring (Request Linux kernel > 5.4.3), currently it is experimental
- VPP : Deprecated.

SPDK NVMe-oF target design highlights

NVMe* over Fabrics Target Features	Performance Benefit
Utilizes user space NVM Express* (NVMe) Polled Mode Driver	Reduced overhead per NVMe I/O
Group polling on each SPDK thread (binding on CPU core) for multiple transports	Efficient Parallelism among different CPU cores
Each connection pinned to dedicated SPDK thread	No synchronization overhead among connections.
Asynchronous NVMe CMD handling in whole life cycle	No locks in NVMe CMD data handling path

Performance optimization for TCP Transport

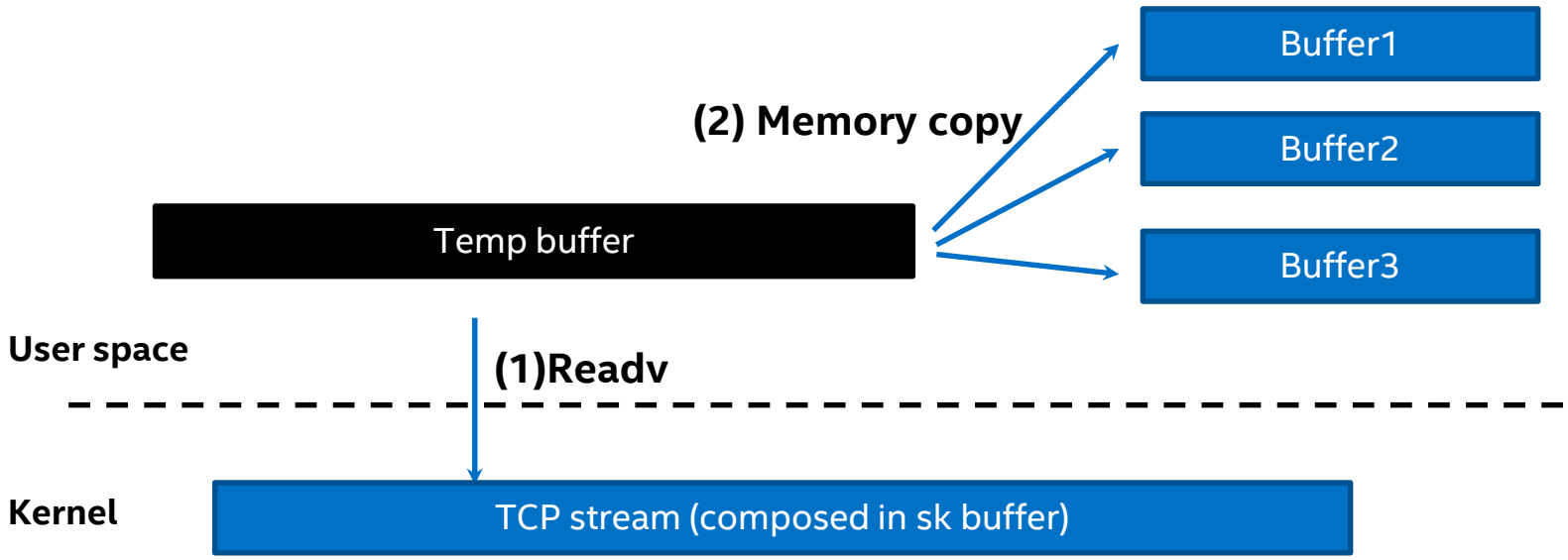
- Great efforts spent to optimize the TCP transport solution from both software and hardware aspects.
- Software:
  - Advanced optimizations for using Kernel TCP/IP stack and io_uring based async programming approach.
- Hardware:
  - Acceleration using Intel® Ethernet 800 Series with Application Device Queues (ADQ)

Software optimization based on Kernel TCP/IP stack and io_uring

Optimization Techniques for using kernel TCP/IP stack

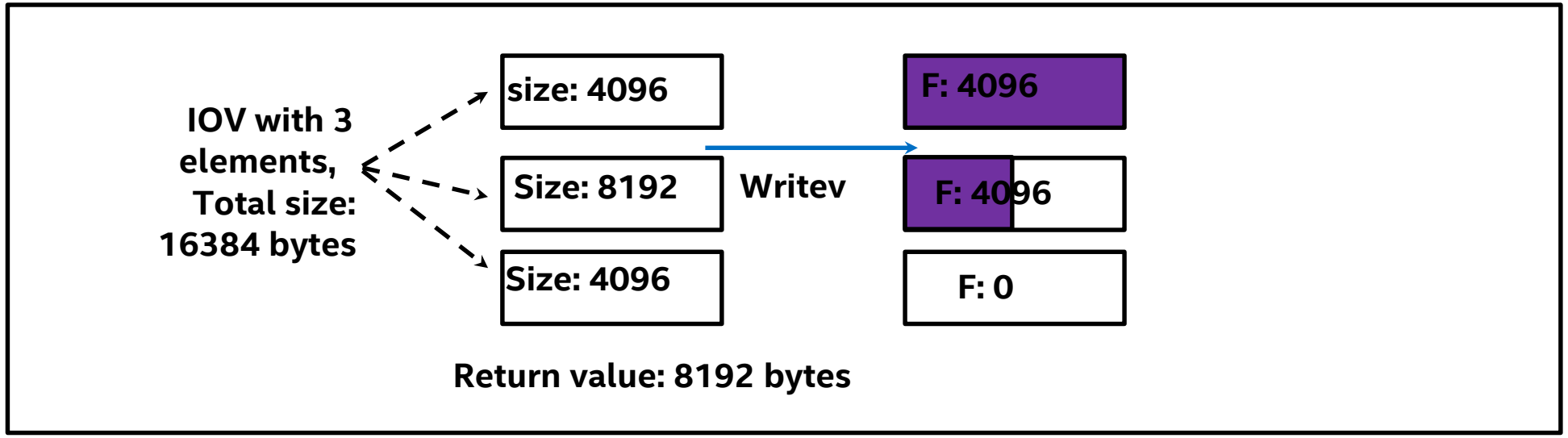
- **Nonblock mode**: O_NONBLOCK setting on FD
- **Group based strategy** on Pollin, Write(v) for many TCP connections.
  - Benefit: Reduce the system call overhead
  - For example, (1) Group Pollin reduce number of readv calls; (2) Group based writev operations via uring sock on 16 TCP connections can reduce 15 system calls in one round.
- **Dedicated CPU core handling**: Each connection on file descriptor should be handle by dedicated thread or CPU core.
- **Buffered read on each socket**: Reduce system call overhead
- **Batched write on each socket**: To reduce system call and improve throughput

Buffered read support in SPDK



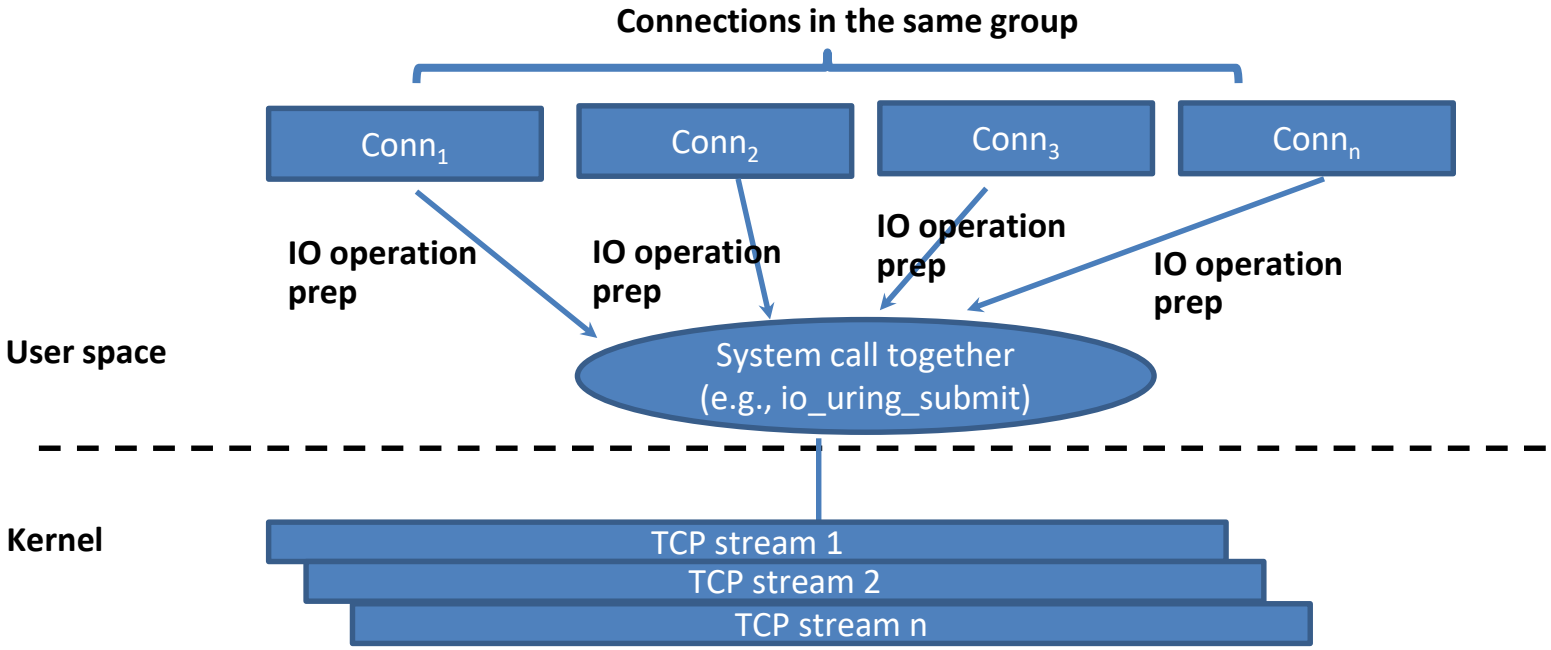
- In this case, Buffer1, 2, 3 can be determined by the application's own logic. This solution tries to reduce the system calls, but it introduces memory copy overhead. We recommend to use IOAT driver and CBDMA to avoid CPU copy overhead.
- SPDK has util library (located in lib/util in spdk source folder) which supports the read buffer with pipe usage manner.

Batched write support in SPDK



- SPDK posix/uring libraries can merge the write I/Os from the app into big vectors to reduce system calls.
- With Merged write, still need handle partial write if NONBLOCK I/O is used.

Group based asynchronous I/O operation with io_uring



Implemented in SPDK uring sock library (located in module/sock/uring in spdk folder)

SPDK NVMe/TCP Acceleration using Intel® Ethernet 800 Series with ADQ

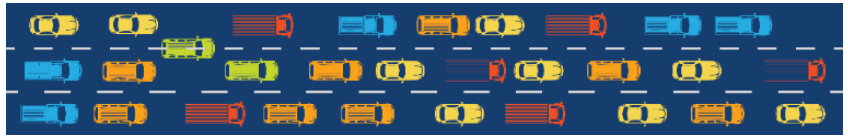
Intel® Ethernet 800 Series with Application Device Queues (ADQ)

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

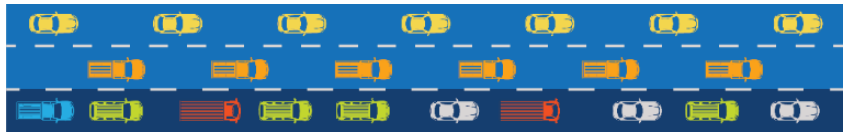
ADQ works by:

- Filtering application traffic to a dedicated set of queues
- Connecting application threads of execution to specific queues within the ADQ queue set
- Controlling bandwidth 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



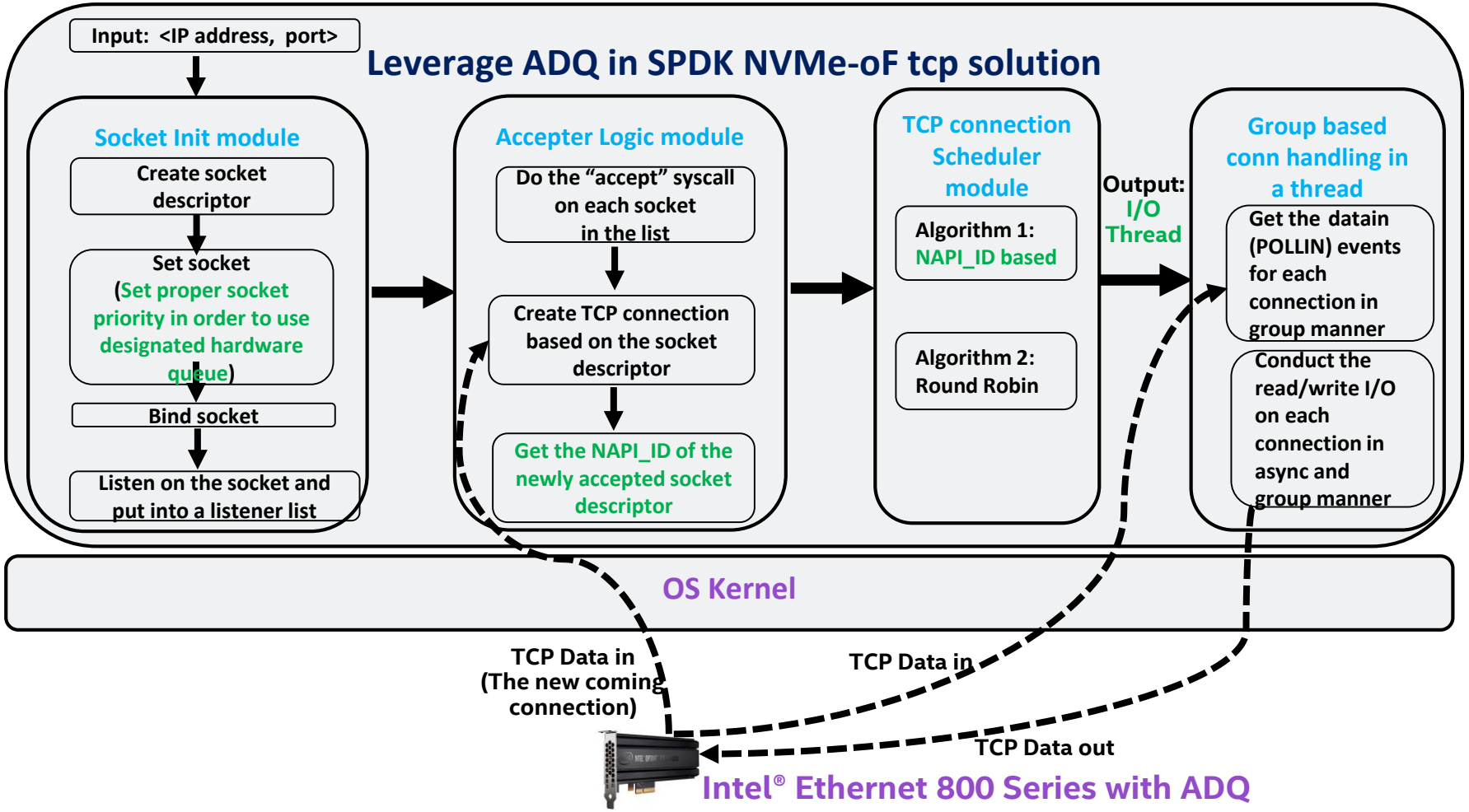
ADQ benefits:

ADQ enabling in SPDK

- SPDK sock library
  - Implements a function to get NAPI_ID for the socket file descriptor.
  - Implements a function to get optimal socket polling group.
- SPDK NVMe/NVMe-oF layer
  - Provides NAPI_ID base scheduler to group NVMe TCP connections with a same NAPI_ID into the same polling group.

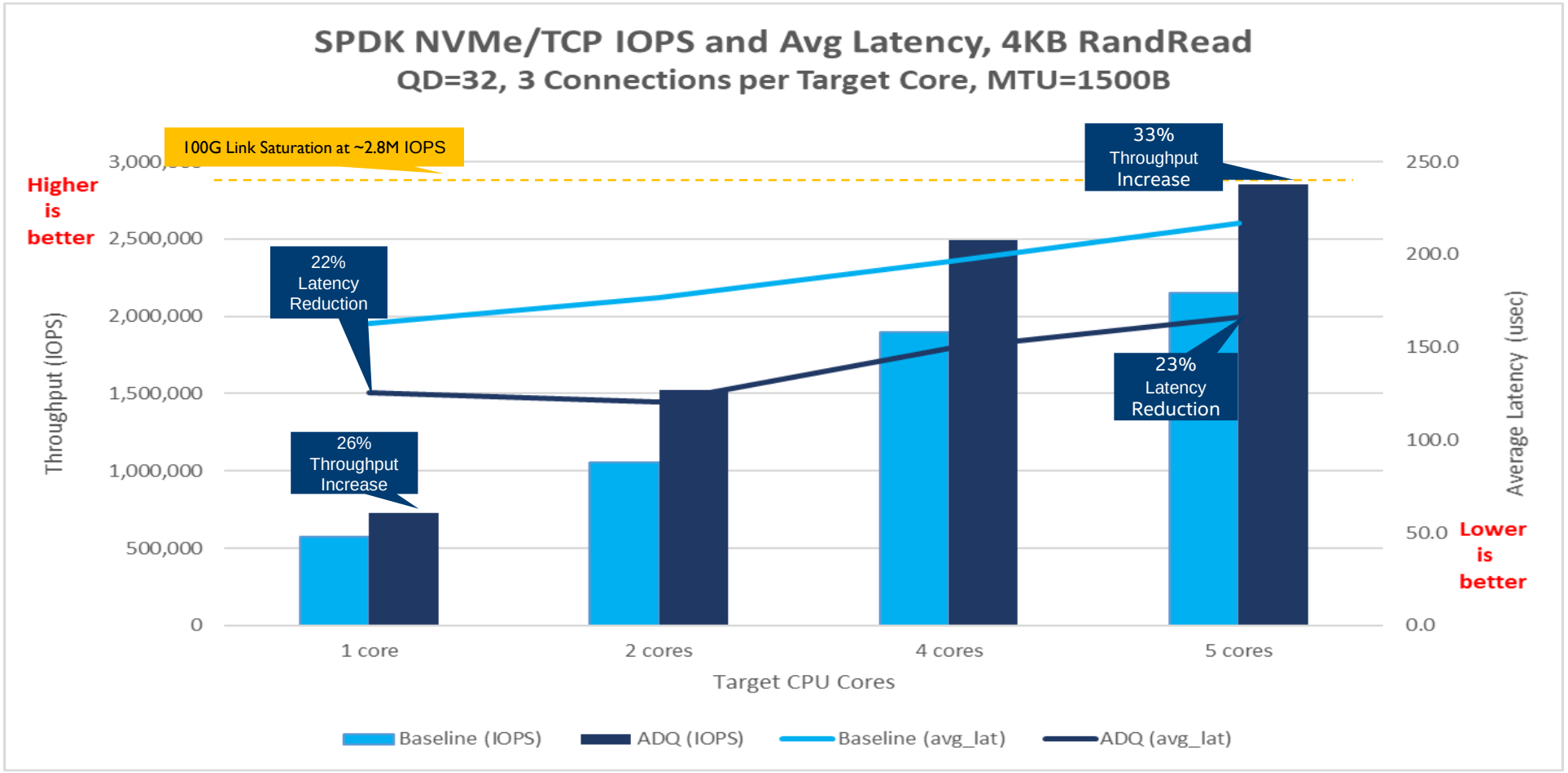
For details on ADQ SW enabling in kernel and device driver, please refer to
- SDC'20 presentation on Tuning and Optimizing Ethernet-based NVMe over Fabric transport Protocols
- SDC'19 presentation on Selecting an NVMe-oF™ Ethernet Transport RDMA or TCP?
- Netdev 0x14 presentation on ADQ for system-level network I/O performance improvements

ADQ integration in SPDK NVMe-oF TCP



Performance Evaluation on Intel® Ethernet 800 Series with ADQ

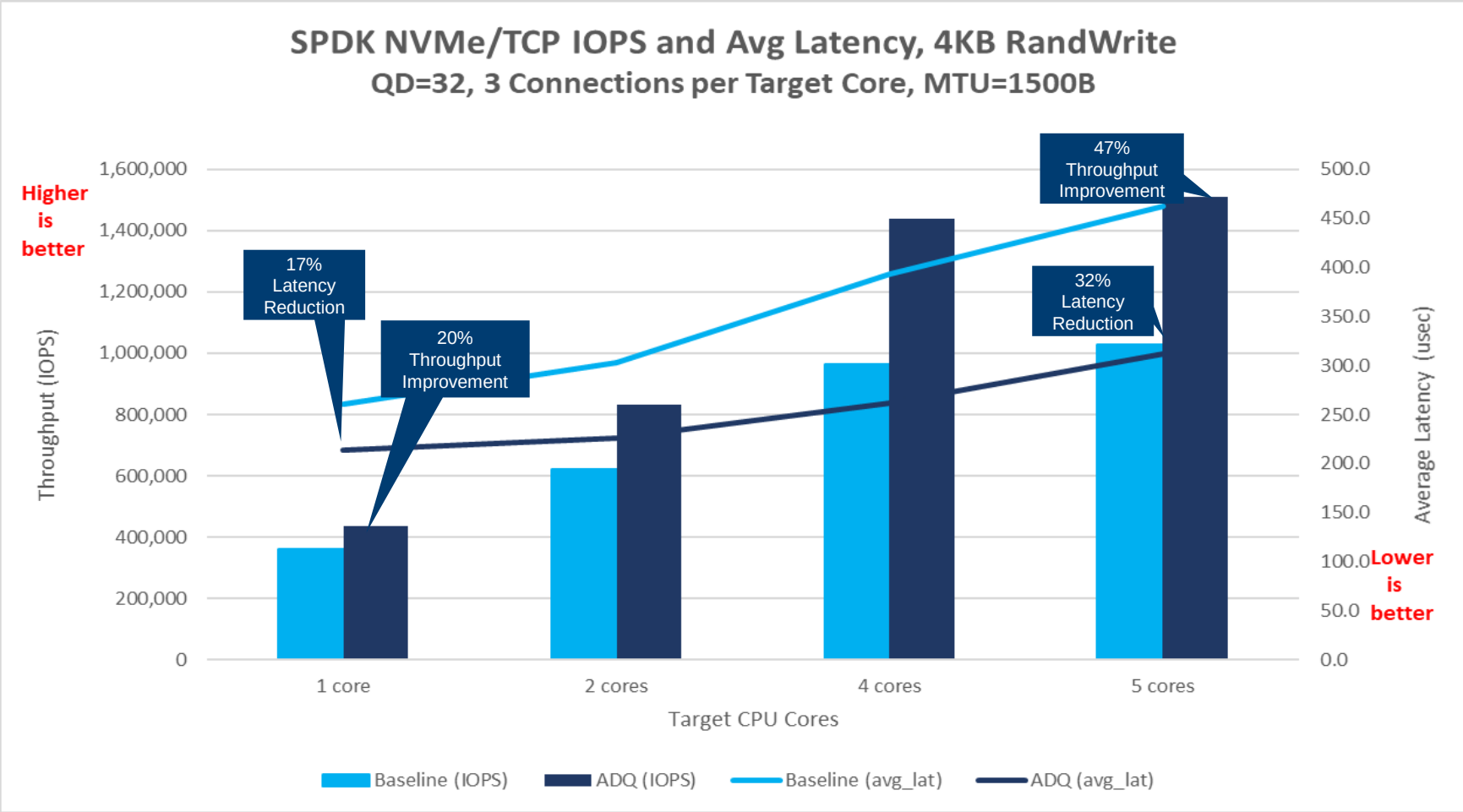
ADQ Improves IOPS with Reduced Latency - Read



- ADQ shows substantial performance improvement to SPDK NVMe/TCP implementation.
- With ADQ, 5 cores can saturate 100Gb link with 4KB IO size.

* For more complete information about performance and benchmark results, visit www.intel.com/benchmarks. Test configurations in slide 24 and 25.

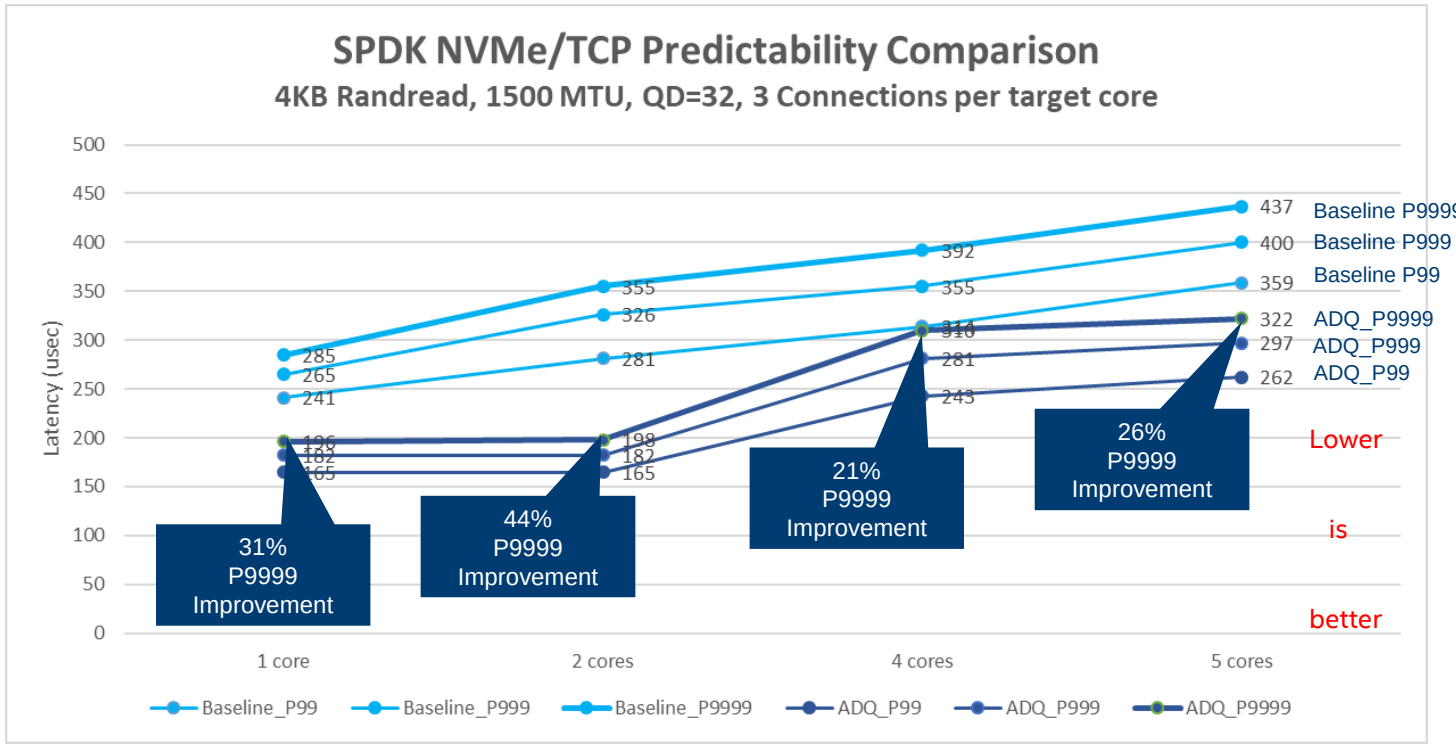
ADQ Improves IOPS with Reduced Latency - Write



- ADQ shows substantial performance improvement to SPDK NVMe/TCP implementation.

* For more complete information about performance and benchmark results, visit www.intel.com/benchmarks. Test configurations in slide 24 and 25.

ADQ Improves Predictability



- ADQ reduces tail of latency substantially across different configurations.

* For more complete information about performance and benchmark results, visit www.intel.com/benchmarks. Test configurations in slide 24 and 25.

Conclusion

Conclusion

- SPDK NVMe-oF solution is well adopted by the industry. In this presentation, we talked about:
  - Software and hardware methods to optimize SPDK NVMe/TCP transport.
  - SPDK NVMe/TCP performance acceleration using Intel® Ethernet 800 Series with ADQ.
  - ADQ support in SPDK 20.07 release.
  - ADQ significantly improves SPDK NVMe/TCP performance with higher IOPS and reduced latency.
    - Up to 33% IOPS improvement and 26% P9999 latency reduction at wire rate.
- Call for activity in SPDK community
  - Welcome to bug submission, idea discussion and patch submission.

Q & A

Appendix: SPDK NVMe/TCP with ADQ Testing Configuration

	SUT	Client
Test by	Intel	Intel
Test date	8/27/2020	8/27/2020
Platform	Dell R740XD	Dell R740XD
# Nodes	1	1
# Sockets	2	2
CPU	Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz	Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz
Cores/socket, Threads/socket	28 cores per socket, 56 threads per socket	28 cores per socket, 56 threads per socket
ucode	0x500001c	0x500001c
HT	Disabled *	Disabled
Turbo	Enabled	Enabled
BIOS version	2.1.8	2.1.8
System DDR Mem Config: slots / cap / run-speed	8 slots / 16GB / 2666 MT/s + 4 slots / 8GB / 2666 MT/s	8 slots / 16GB / 2666 MT/s + 4 slots / 8GB / 2666 MT/s
System DCPMM Config: slots / cap / run-speed	N/A	N/A
Total Memory/Node (DDR+DCPMM)	160GB DDR4-2666 DIMM	160GB DDR4-2666 DIMM
Storage - boot	100GB SATA SSD	100GB SATA SSD
Storage - application drives	6x Intel® Optane ™ SSD DC P4800X, PCIe 3.0, x4	N/A
Network Adapter	Intel E810-C	Intel E810-C
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 8.1 (Ootpa)	Red Hat Enterprise Linux 8.1 (Ootpa)
Kernel	5.8.0	5.8.0
Workload & version	Fio 3.15	Fio 3.15
Compiler	GCC 8.3.1 20191121 (Red Hat 8.3.1-5)	GCC 8.3.1 20191121 (Red Hat 8.3.1-5)
Network Adapter Driver	Ice-1.2.0-rc4 FW ver: 0x8000433e	Ice-1.2.0-rc4 Fw ver: 0x8000433e

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

Appendix: OS and Network Adapter Configuration

Red Hat* Enterprise Linux 8.1 Configuration
stopped: irqbalance, cpupower, firewalld, SELINUX disabled
tuned-adm profile throughput-performance
Kernel 5.8.0
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
sysctl -w net.core.busy_poll=0

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
<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 $((porti)) 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
<path-to-ice>/scripts/set_irq_affinity –X local <interface>
ethtool -L <interface> rx $queues tx $queues


intel