Overview of SPDK NVMe-oF solution and Intel's NIC update

Ziye Yang, NCLG, DCG, Intel

Fred, Zhang, CG, Intel

Notices and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

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.

Performance results are based on testing as of February 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product or computer system can be absolutely secure. For more complete information visit http://www.intel.com/benchmarks.

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.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo and others are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

Agenda

- Why NVMe-oF & Why SPDK?
- SPDK NVMe-oF development history & status
- SPDK TCP transport introduction
- Intel NIC update
- Conclusion

Agenda

- **Why NVMe-oF & Why SPDK?**
- SPDK NVMe-oF development history & status
- SPDK TCP transport introduction
- Intel NIC update
- Conclusion

Why NVMe over fabrics?

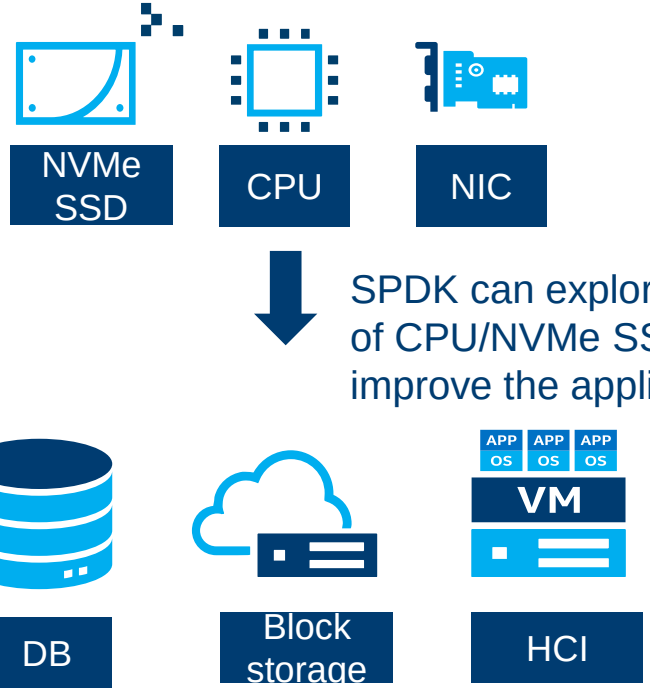
NVMe over Fabrics project Goals: **Simplicity**, **Efficiency**, and **End-to-End** NVMe model

- **Scale NVMe remotely, beyond limitation of local PCIe**
- **Provide remote NVMe performance equivalent to local PCIe**
  - ❑ Take advantage of inherent parallelism of deep multi-Q model (64 commands/queue, up to 64K queues)
  - ❑ Avoid translation to or from other protocols (e.g., SCSI)
  - ❑ NVMe commands and structures are transferred end-to-end
- **Maintain architecture and software consistency between fabric types by standardizing a common**
  - Abstraction
  - encapsulation definition
- **Storage computing disaggregation VS Storage computing aggregation**

Why SPDK solution?

What is SPDK (storage performance development kit: http://spdk.io)

- *User Space, High Performance, Scalable Library*
- *End-to-End storage accelerated Solution*
- *Extensible Framework and Component*



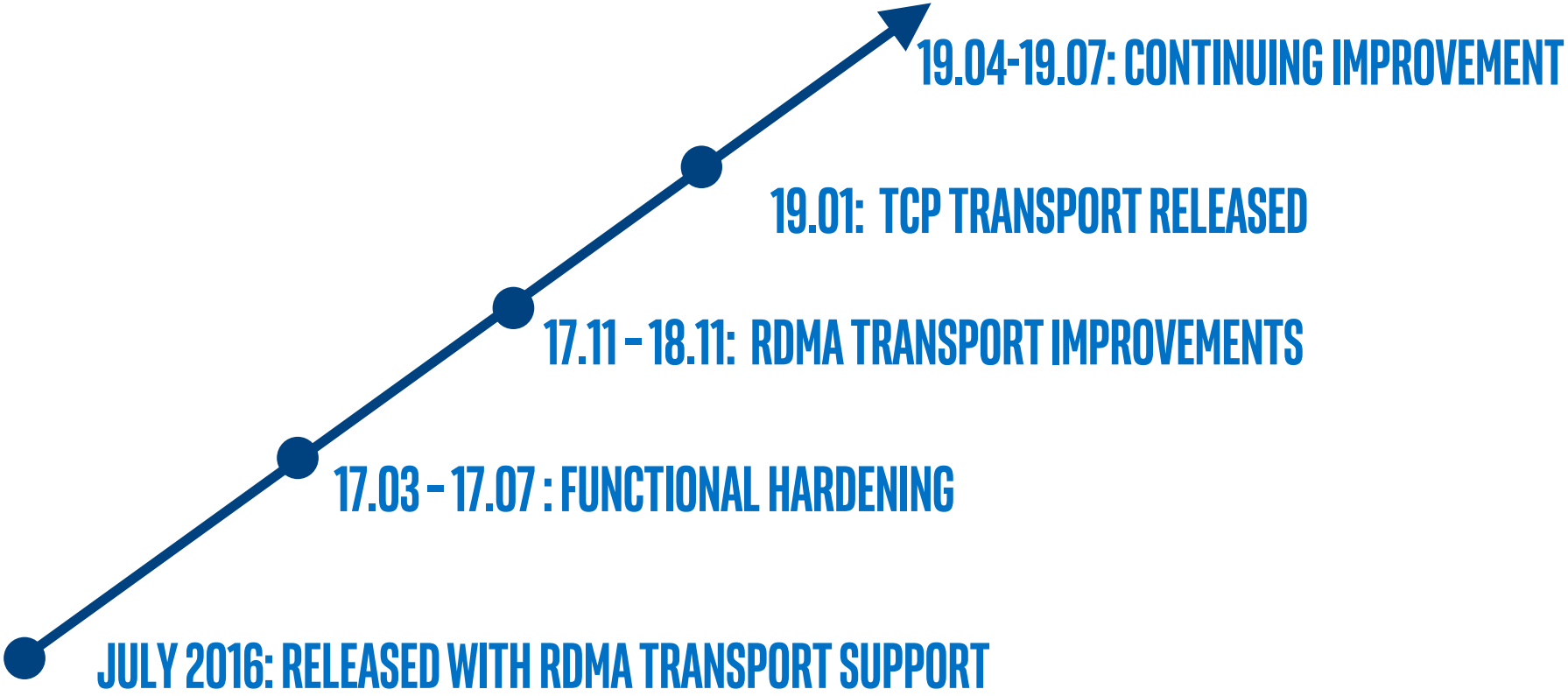
Why SPDK based NVMe-oF user space solution?

Ingredients	Linux Kernel	SPDK solution
Deployment	Need to upgrade the kernel or backporting (Most cloud storage providers' kernel is old)	Easy to deploy (Since it is in user space)
Performance	Worse than SPDK	Better than Linux kernel in IOPS and latency (Especially for per CPU core)
Spec compliance and functionality	Follow NVMe-oF spec	Follow NVMe-oF spec
Service recovery	Need to reload the module or reboot kernel	Restart SPDK NVMe-oF target application
Further development challenge	Need to update the kernel module, difficulty (*****)	Need to develop based on SPDK framework, difficulty (***)

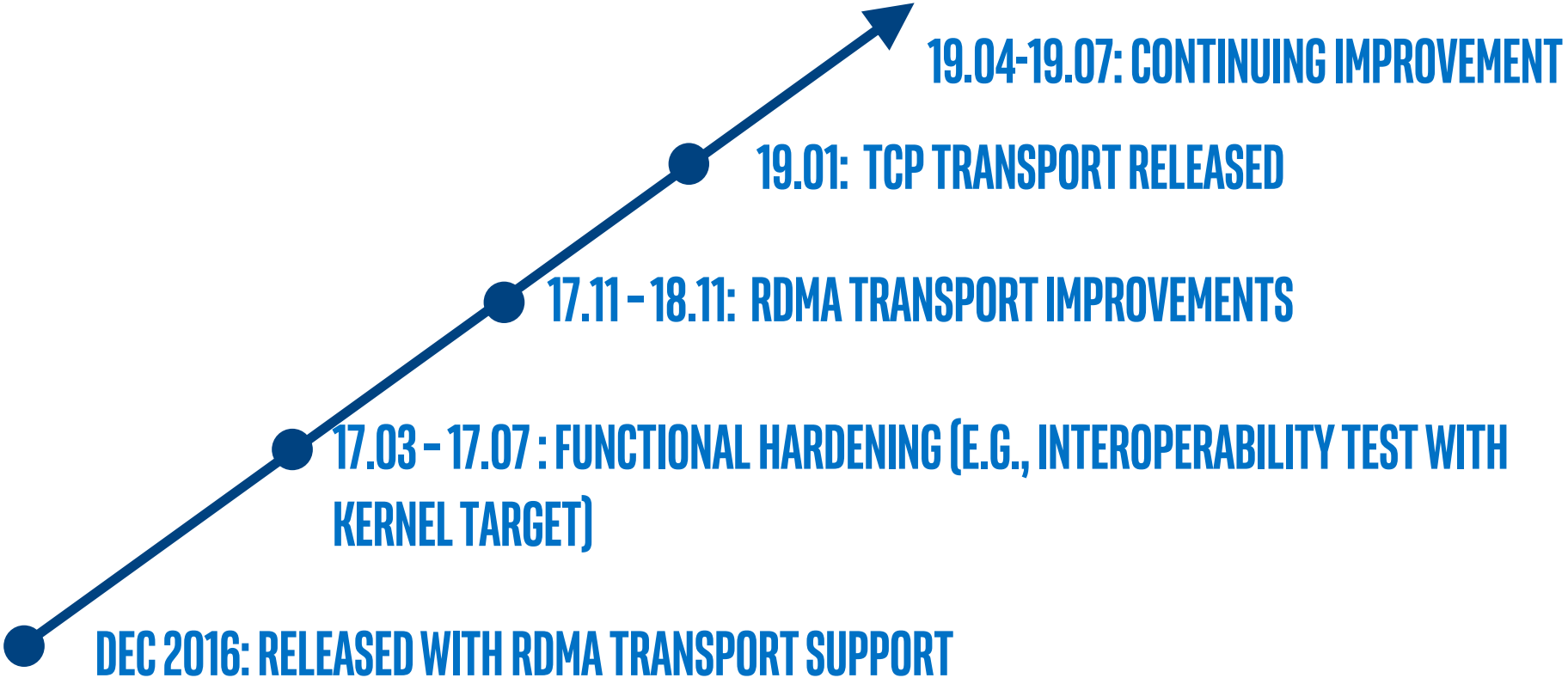
Agenda

- Why NVMe-oF & Why SPDK?
- **SPDK NVMe-oF development history & status**
- SPDK TCP transport introduction
- Intel NIC update
- Conclusion

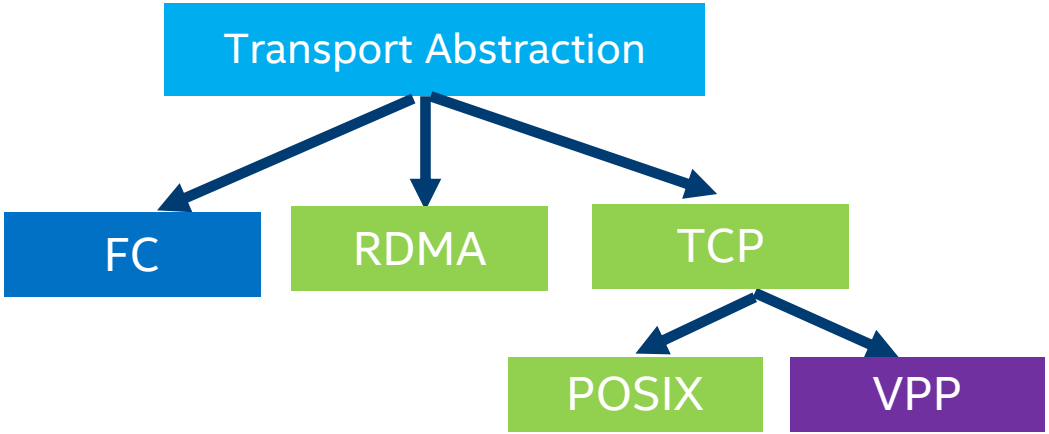
SPDK NVMe-oF Target Timeline



SPDK NVMe-oF Host Timeline



General design and implementation



Just merged, will appear in 19.07

Already released

Still in progress for performance tuning

Agenda

- Why NVMe-oF & Why SPDK?
- SPDK NVMe-oF development history & status
- **SPDK TCP transport introduction**
- Intel NIC update
- Conclusion

Why TCP transport solution?

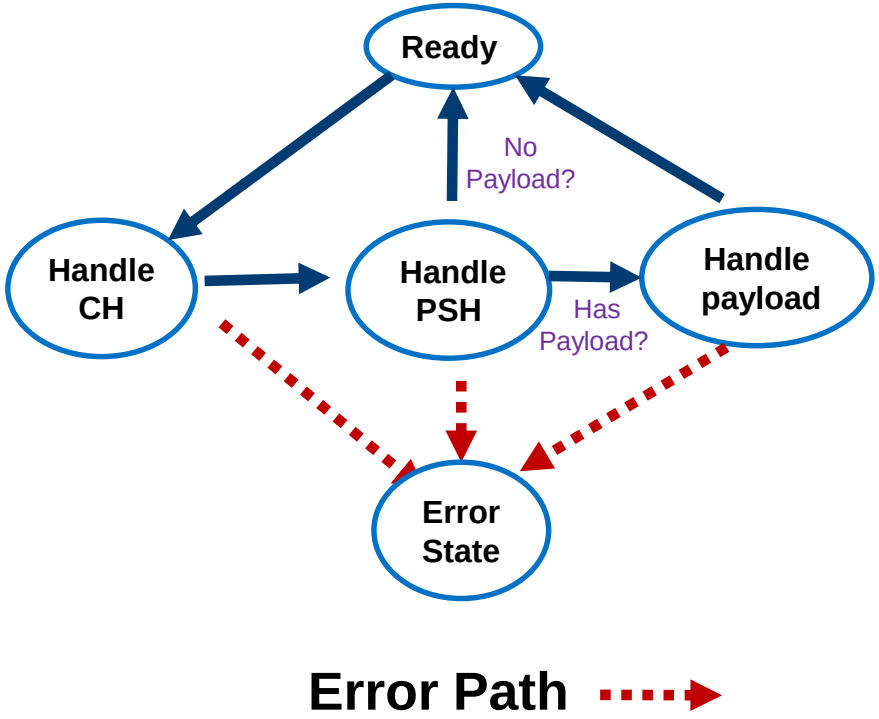
Ingredients	RDMA transport	TCP transport
Performance	High performance	Low performance
Hardware dependency	RDMA capable NICs (supporting RoceV2 or iwarp)	No dependency on NICs (Can reuse the existing NICs in data center)
Physical distance restriction	May only be suitable in a small data center	No distance requirements (suitable for general cloud storage interface)
Programming skills requirements	Must be familiar with RDMA related primitives. (High requirements for developers for debugging)	Relatively easy to debug (Since most programmers are familiar with TCP)
Usage scenarios	Especially for high performance usage	Can be applied into any usage scenarios.

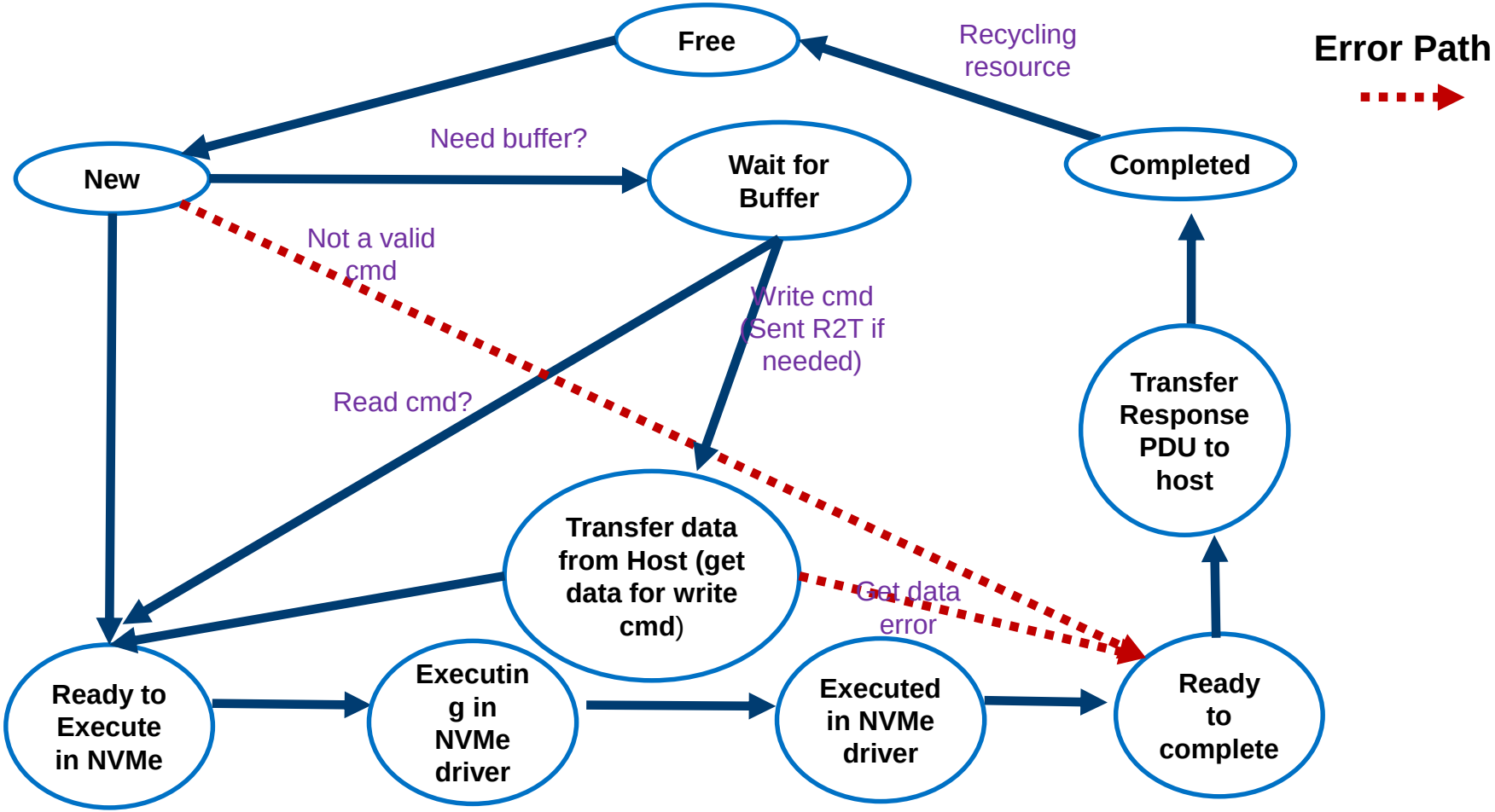
Performance design consideration for TCP transport in target side

Ingredients	Methodology
Design framework	Follow the general SPDK NVMe-oF framework (e.g., polling group)
TCP connection optimization	Use the SPDK encapsulated Socket API (preparing for integrating other stack, e.g., VPP)
NVMe/TCP PDU handling	Use state machine to track
NVMe/TCP request life time cycle	Use state machine to track (Purpose: Easy to debug and good for further performance improvement)

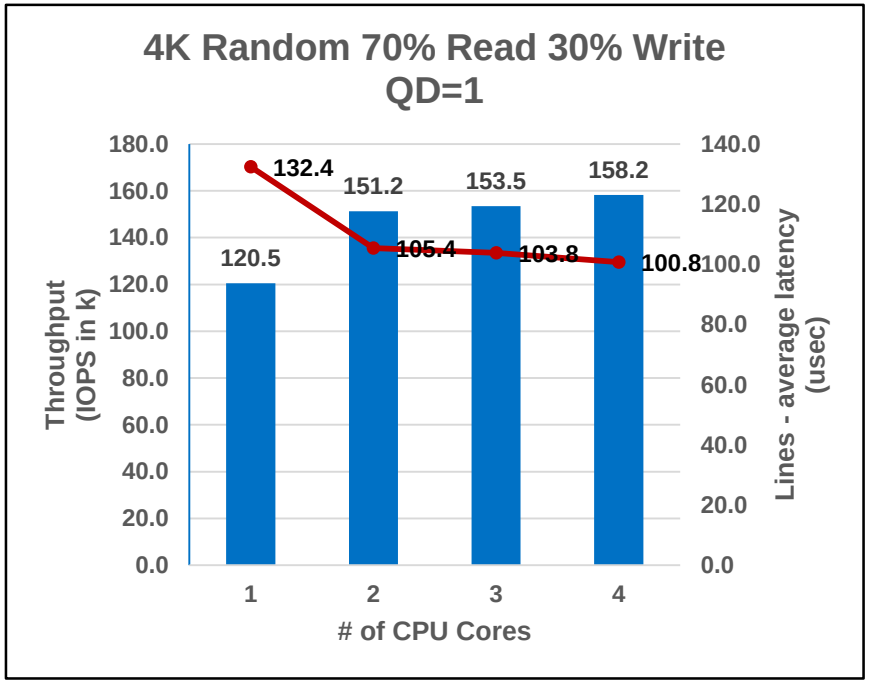
TCP PDU Receiving handling for each connection

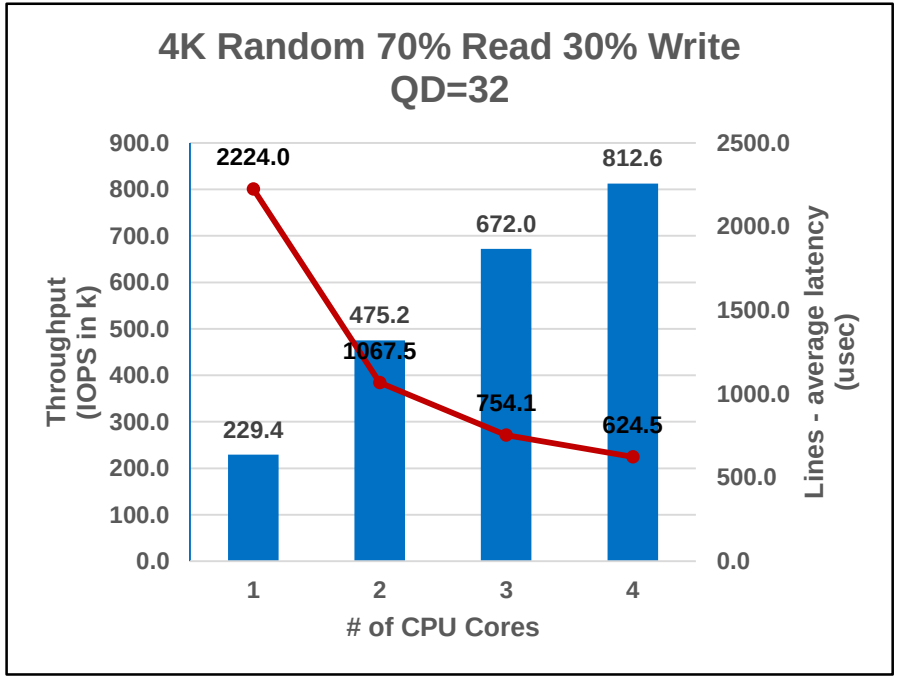
```
enum nvme_tcp_pdu_recv_state {
        /* Ready to wait PDU */
        NVME_TCP_PDU_RECV_STATE_AWAIT_PDU_READY,

        /* Active tqpair waiting for any PDU common header */
        NVME_TCP_PDU_RECV_STATE_AWAIT_PDU_CH,

        /* Active tqpair waiting for any PDU specific header */
        NVME_TCP_PDU_RECV_STATE_AWAIT_PDU_PSH,

        /* Active tqpair waiting for payload */
        NVME_TCP_PDU_RECV_STATE_AWAIT_PDU_PAYLOAD,

        /* Active tqpair does not wait for payload */
        NVME_TCP_PDU_RECV_STATE_ERROR,
};
```




Free
Recycling resource
Error Path
Need buffer?
New
Wait for Buffer
Completed
Not a valid cmd
Write cmd (Sent R2T if needed)
Read cmd?
Transfer Response PDU to host
Transfer data from Host (get data for write cmd)
Get data error
Ready to Execute in NVMe
Executing in NVMe driver
Executed in NVMe driver
Ready to complete

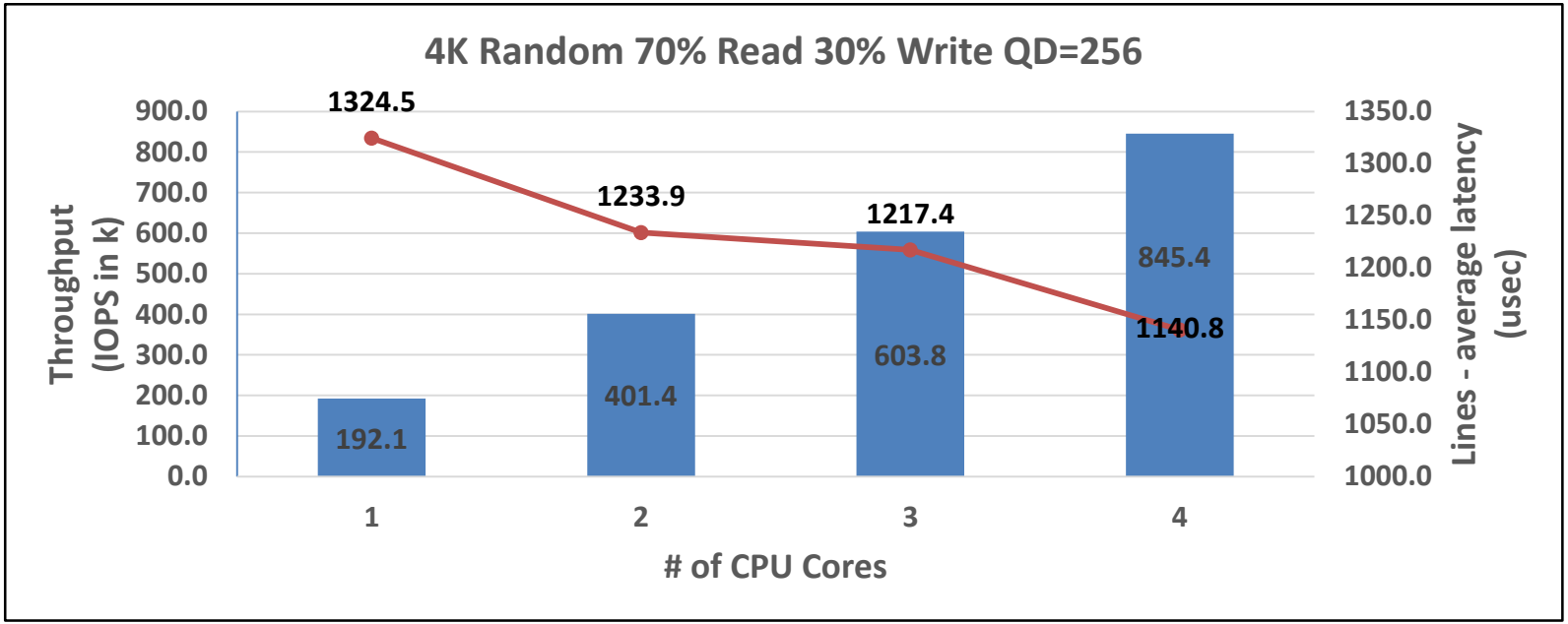
SPDK TARGET SIDE (TCP TRANSPORT): I/O SCALING





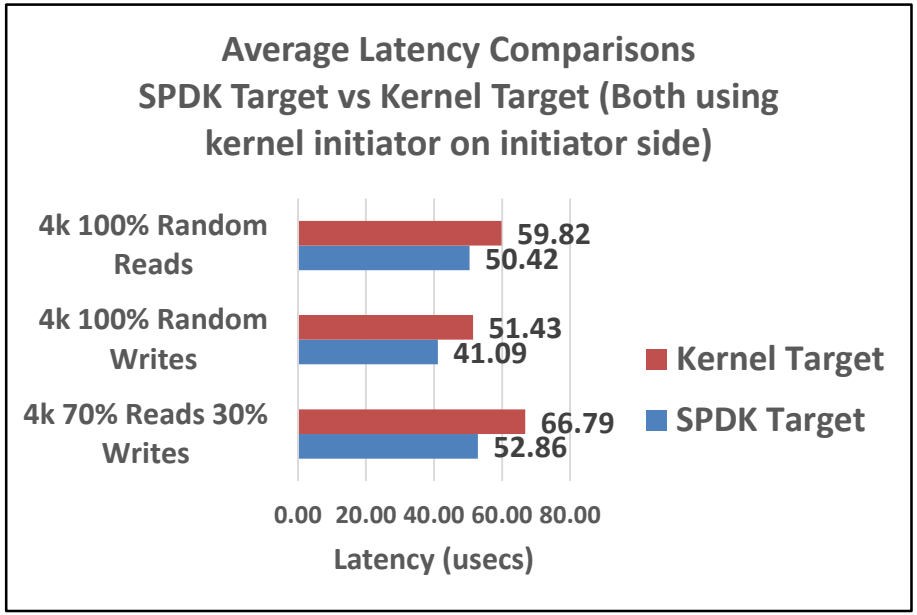
System configuration: (1) Target: server platform: SuperMicro SYS2029U-TN24R4T; 2x Intel® Xeon® Platinum 8180 CPU @ 2.50 GHz, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled, 4x 2GB DDR4 2666 MT/s, 1 DIMM per channel; 2x 100GbE Mellanox ConnectX-5 NICs; Fedora 28, Linux kernel 5.05, SPDK 19.01.1; 6x Intel® P4600TM P4600x 2.0TB; (2) initiator: Server platform: SuperMicro SYS-2028U TN24R4T+; 44x Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz (HT off); 1x 100GbE Mellanox ConnectX-4 NIC; Fedora 28, Linux kernel 5.05, SPDK 19.0.1. (3) : Fio ver: fio-3.3; Fio workload: blocksize=4k, iodepth=1, iodepth_batch=128, iodepth_low=256, ioengine=libaio or SPDK bdev engine, size=10G, ramp_time=0, run_time=300, group_reporting, thread, direct=1, rw=read/write/rw/randread/randwrite/randrw

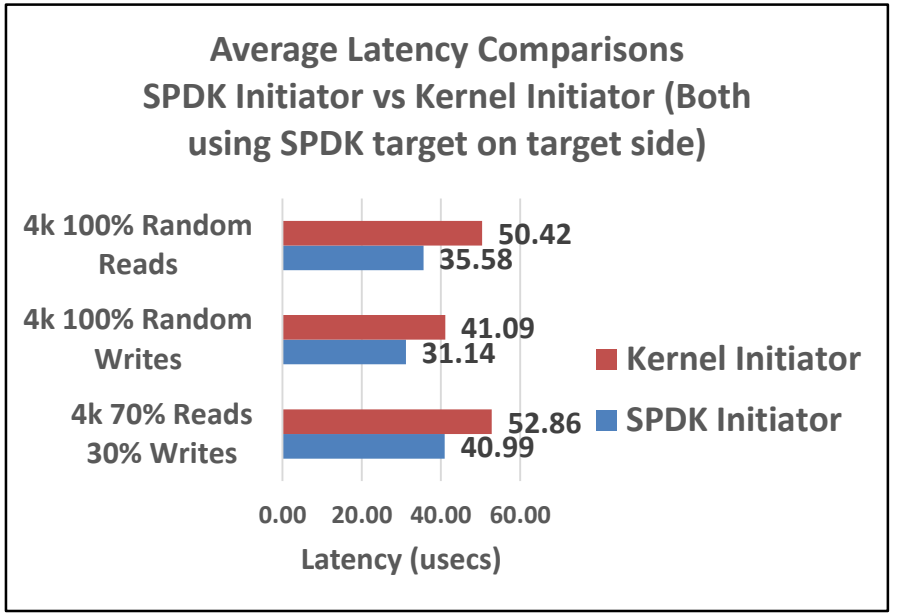
SPDK HOST SIDE (TCP TRANSPORT): I/O SCALING



System configuration: (1) Target: server platform: SuperMicro SYS2029U-TN24R4T; 2x Intel® Xeon® Platinum 8180 CPU @ 2.50 GHz, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled, 4x 2GB DDR4 2666 MT/s, 1 DIMM per channel; 2x 100GbE Mellanox ConnectX-5 NICs; Fedora 28, Linux kernel 5.05, SPDK 19.01.1; 6x Intel® P4600TM P4600x 2.0TB; (2) initiator: Server platform: SuperMicro SYS-2028U TN24R4T+; 44x Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz (HT off); 1x 100GbE Mellanox ConnectX-4 NIC; Fedora 28, Linux kernel 5.05, SPDK 19.0.1. (3) : Fio ver: fio-3.3; Fio workload: blocksize=4k, iodepth=1, iodepth_batch=128, iodepth_low=256, ioengine=libaio or SPDK bdev engine, size=10G, ramp_time=0, run_time=300, group_reporting, thread, direct=1, rw=read/write/rw/randread/randwrite/randrw

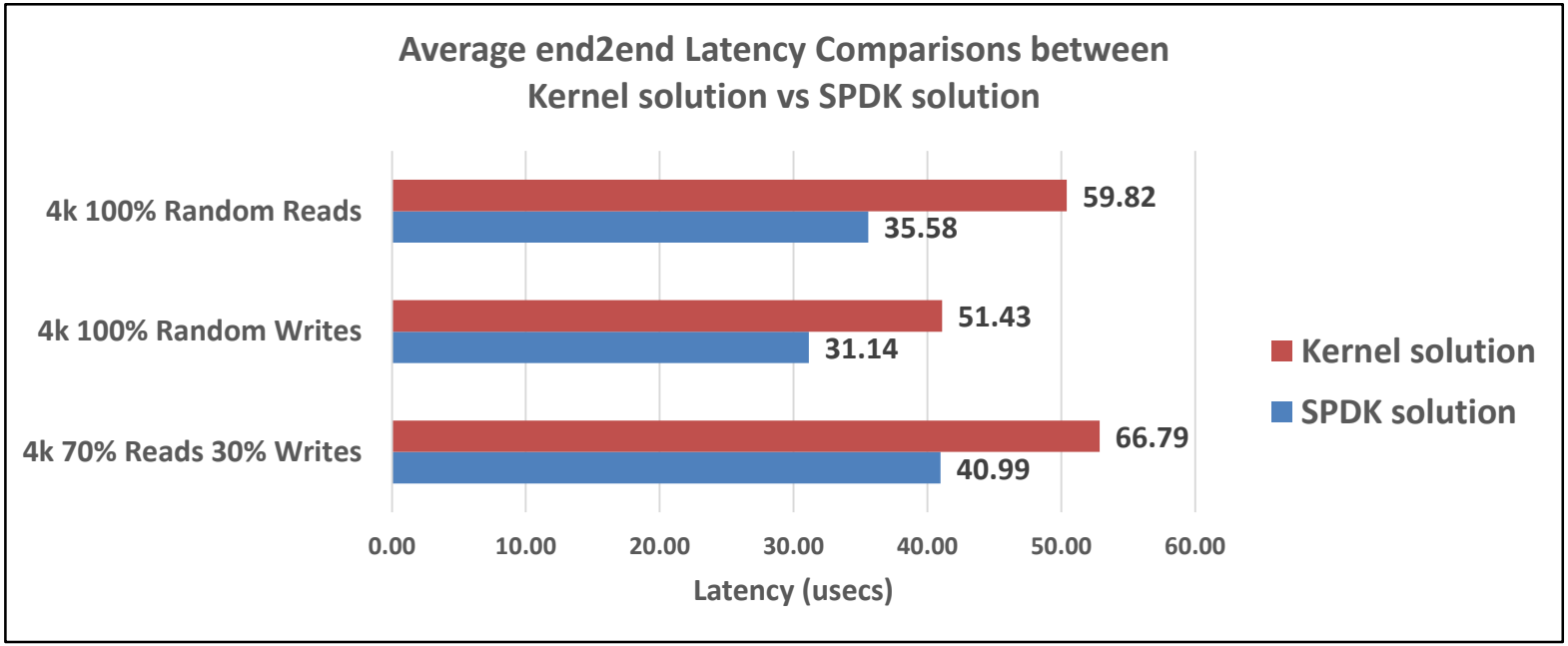
LATENCY COMPARISON BETWEEN SPDK AND KERNEL (NULL BDEV IS USED)





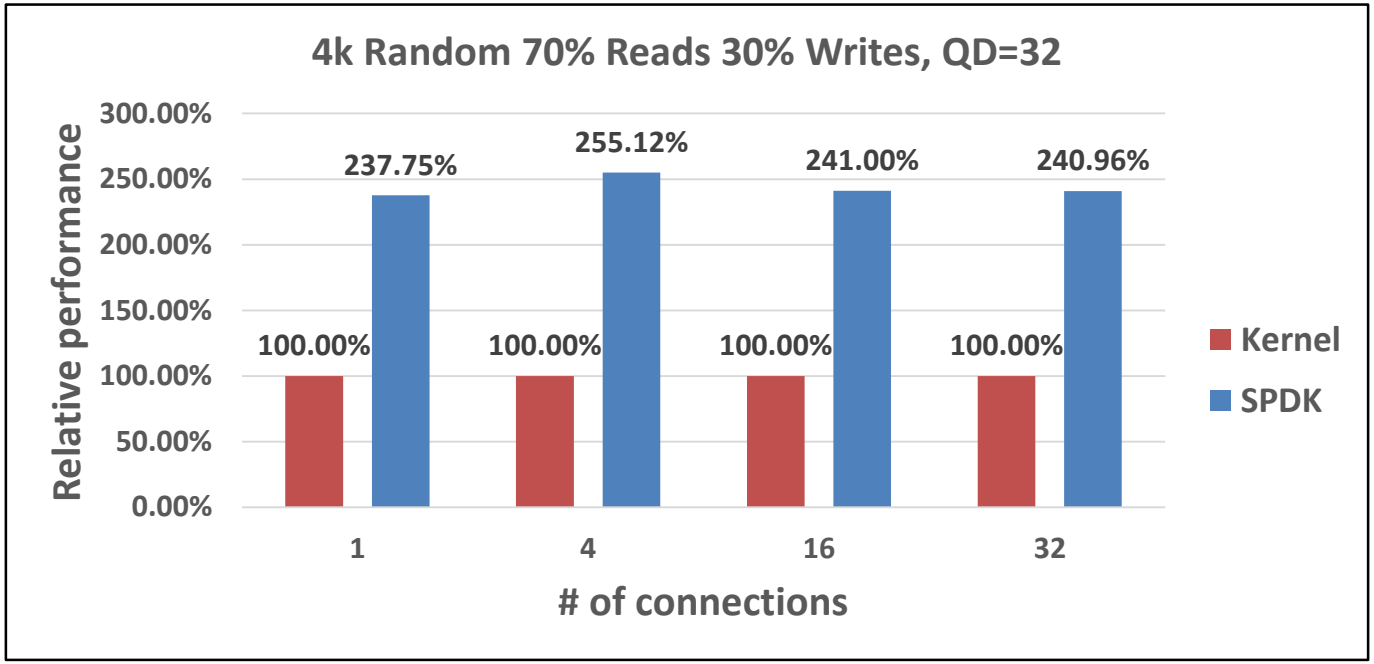
System configuration: (1) Target: server platform: SuperMicro SYS2029U-TN24R4T; 2x Intel® Xeon® Platinum 8180 CPU @ 2.50 GHz, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled, 4x 2GB DDR4 2666 MT/s, 1 DIMM per channel; 2x 100GbE Mellanox ConnectX-5 NICs; Fedora 28, Linux kernel 5.05, SPDK 19.01.1; 6x Intel® P4600TM P4600x 2.0TB; (2) initiator: Server platform: SuperMicro SYS-2028U TN24R4T+; 44x Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz (HT off); 1x 100GbE Mellanox ConnectX-4 NIC; Fedora 28, Linux kernel 5.05, SPDK 19.0.1. (3) : Fio ver: fio-3.3; Fio workload: blocksize=4k, iodepth=1, iodepth_batch=128, iodepth_low=256, ioengine=libaio or SPDK bdev engine, size=10G, ramp_time=0, run_time=300, group_reporting, thread, direct=1, rw=read/write/rw/randread/randwrite/randrw

LATENCY COMPARISON BETWEEN SPDK AND KERNEL (NULL BDEV IS USED)



System configuration: (1) Target: server platform: SuperMicro SYS2029U-TN24R4T; 2x Intel® Xeon® Platinum 8180 CPU @ 2.50 GHz, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled, 4x 2GB DDR4 2666 MT/s, 1 DIMM per channel; 2x 100GbE Mellanox ConnectX-5 NICs; Fedora 28, Linux kernel 5.05, SPDK 19.01.1; 6x Intel® P4600TM P4600x 2.0TB; (2) initiator: Server platform: SuperMicro SYS-2028U TN24R4T+; 44x Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz (HT off); 1x 100GbE Mellanox ConnectX-4 NIC; Fedora 28, Linux kernel 5.05, SPDK 19.0.1. (3) : Fio ver: fio-3.3; Fio workload: blocksize=4k, iodepth=1, iodepth_batch=128, iodepth_low=256, ioengine=libaio or SPDK bdev engine, size=10G, ramp_time=0, run_time=300, group_reporting, thread, direct=1, rw=read/write/rw/randread/randwrite/randrw

IOPS/CORE COMPARISON BETWEEN SPDK AND KERNEL ON TARGET SIDE



System configuration: (1) Target: server platform: SuperMicro SYS2029U-TN24R4T; 2x Intel® Xeon® Platinum 8180 CPU @ 2.50 GHz, Intel® Speed Step enabled, Intel® Turbo Boost Technology enabled, 4x 2GB DDR4 2666 MT/s, 1 DIMM per channel; 2x 100GbE Mellanox ConnectX-5 NICs; Fedora 28, Linux kernel 5.05, SPDK 19.01.1; 6x Intel® P4600TM P4600x 2.0TB; (2) initiator: Server platform: SuperMicro SYS-2028U TN24R4T+; 44x Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz (HT off); 1x 100GbE Mellanox ConnectX-4 NIC; Fedora 28, Linux kernel 5.05, SPDK 19.0.1. (3) : Fio ver: fio-3.3; Fio workload: blocksize=4k, iodepth=1, iodepth_batch=128, iodepth_low=256, ioengine=libaio or SPDK bdev engine, size=10G, ramp_time=0, run_time=300, group_reporting, thread, direct=1, rw=read/write/rw/randread/randwrite/randrw

NVMe/TCP transport: Further development plan

- Continue enhancing the functionality
  - Including the compatible test with Linux kernel solution.
- Performance tuning
- Integration with third party software
  - Deep integration with user space stack: VPP + DPDK, need the stability and performance tuning.
- Leveraging hardware features
  - Use existing hardware features of NICs for performance improvement, e.g., VMA from Mellanox's NIC; ADQ from Intel's 100Gbit NIC.
  - Figuring out offloading methods with hardware, e.g., FPGA, Smart NIC, and etc.

Agenda

- Why NVMe-oF & Why SPDK?
- SPDK NVMe-oF development history & status
- SPDK TCP transport introduction
- **Intel NIC update**
- Conclusion

Intel® Ethernet Architecture Evolution

FOUNDATIONAL NICS

Intel® Ethernet
500 Series
Niantic

Intel® Ethernet
700 Series
Fortville

Intel® Ethernet 800 Series
Columbiaville[1]
Announcing

Capability Richness
More
Less

Fixed Pipeline
SR-IOV and VMDq

Partially Programmable Pipeline
- Table definition modifications with a Dynamic Device Personalization (DDP) profile package

Intel® Ethernet Adaptive Virtual Function (Intel AVF)

Queue and Steering Hardware Assists
- Application Device Queues (ADQ)

Fully Programmable Pipeline
- Table definition with DDP profile packages

Storage
- RDMA (iWARP* & RoCE*v2)

10GbE 40GbE 100GbE

[1]Features & schedule are subject to change. All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

THE GOAL

MOVE DATA FASTER

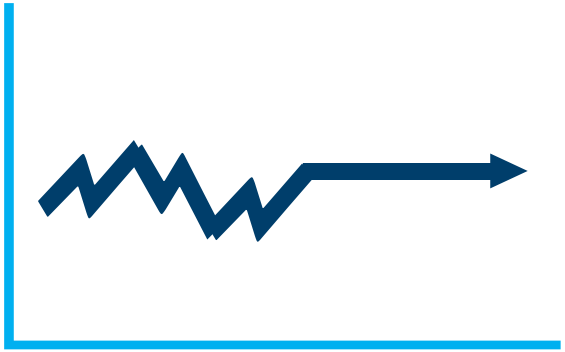
Intel® Ethernet 800 Series Can…

Improve Application Performance
with Application Device Queues (ADQ)

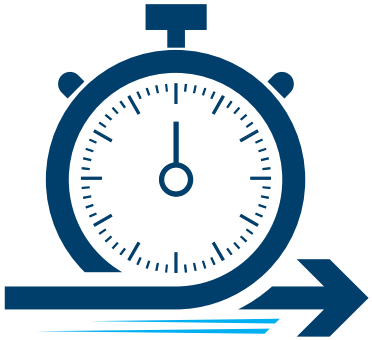
Improve Packet Processing Efficiency
with Dynamic Device Personalization (DDP)

Scale-out Application Performance Parameters

PREDICTABILITY



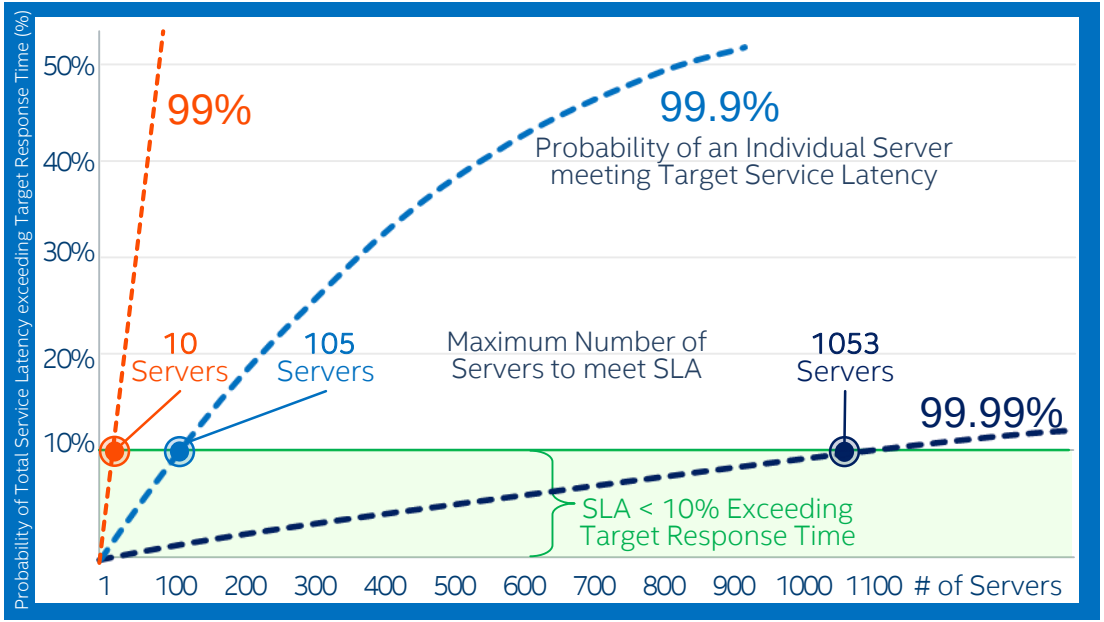
LATENCY



THROUGHPUT

Why Application Response Predictability Matters

Predictability of Data Center Application Performance[1]



"The Tail at Scale" – Communications of the ACM. February 2013
Jeffrey Dean – Google Senior Fellow and Luiz Andrée Barroso – Google Fellow / VP of Engineering
https://cseweb.ucsd.edu/~gmporter/classes/fa17/cse124/post/schedule/p74-dean.pdf

Meeting the Scale-out Challenge

- Reducing variability in application response time (jitter) improves throughput and reduces latency

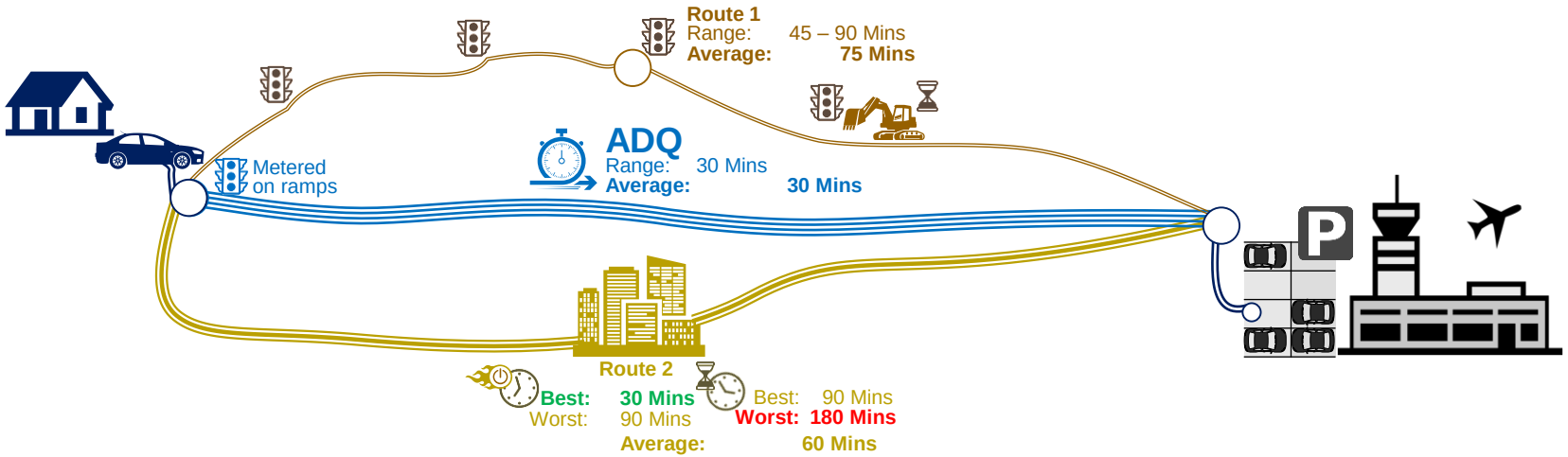
Benefits of Reducing Jitter

- More servers can be added to parallelize task
- Support more end-users with existing hardware

Higher predictability enables more servers working in parallel within a desired response time

How to Improve Predictability

Analogy: Time to Reach Airport for Flight



Application Device Queues (ADQ) Improves Predictability with Dedicated Lanes and Rate Limiting

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

What is ADQ?

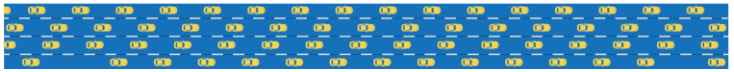
- An application specific queuing and steering technology

How does ADQ work?

- Filters 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

With ADQ

Application traffic to a dedicated set of queues



Without ADQ

Application traffic intermixed with other traffic types

What are the benefits of ADQ?

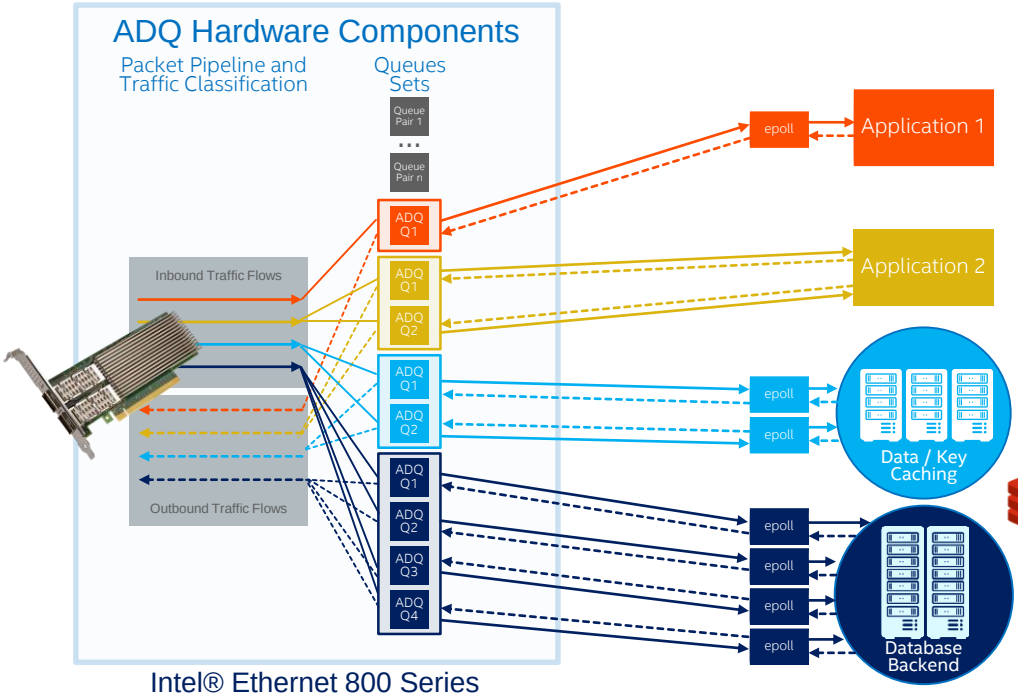
INCREASES APPLICATION **PREDICTABILITY**

REDUCES APPLICATION **LATENCY**

IMPROVES APPLICATION **THROUGHPUT**

Intel® Ethernet -- Application Device Queues (ADQ)[1]

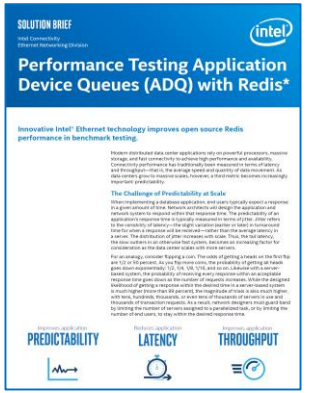
Latest Network Technology Innovation for Intel® Ethernet 800 Series



Intel® Ethernet 800 Series Controllers and Network Adapters

ADQ Basics

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



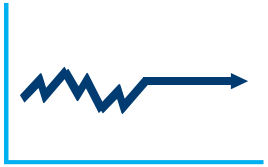
Intel® Ethernet 800 Series with ADQ on Redis Solution Brief

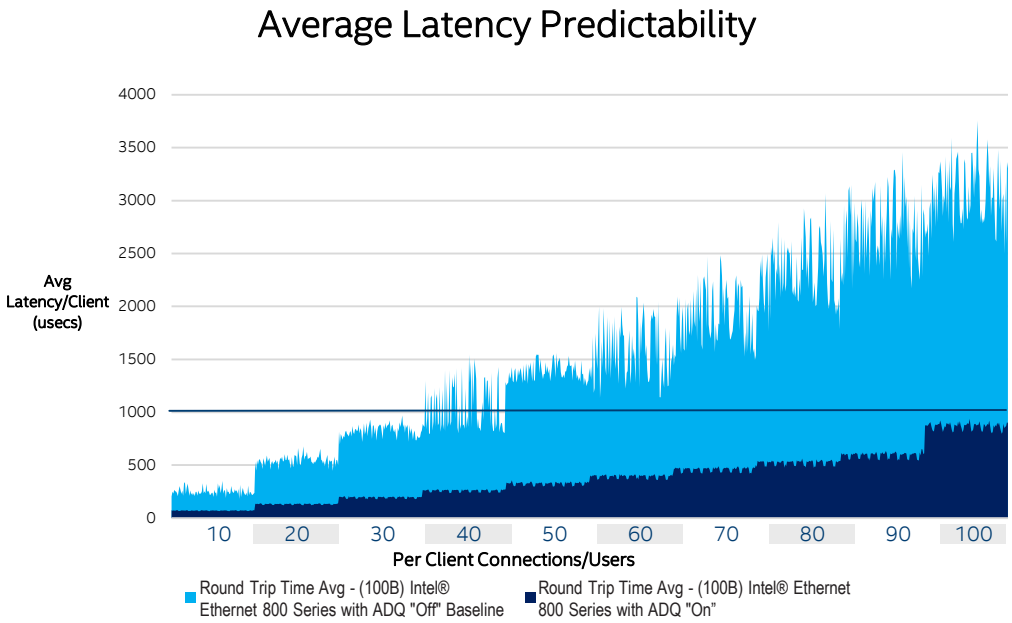
[1]Features & schedule are subject to change. All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Application Device Queues (ADQ) – Redis* Open Source Database

Performance Results

INCREASES APPLICATION PREDICTABILITY





Lower is Better

>50% PREDICTABILITY INCREASE WITH OPEN SOURCE REDIS

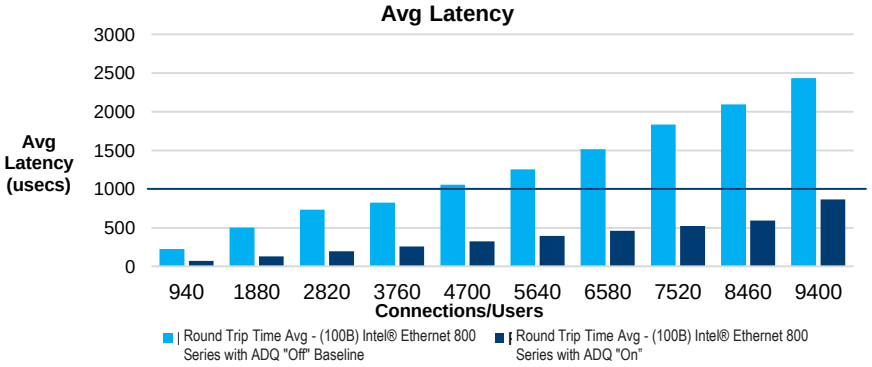
redis*

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 complete information visit http://www.intel.com/performance. Source: Performance results are based on Intel internal testing as of February 2019, and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure. Tests performed using Redis Open Source on 2nd Generation Intel® Xeon® Scalable processors and Intel® Ethernet 800 series 100GbE on Linux 4.19.18 kernel (see backup) Calculation: (new - old) / old x 100% for reduction in variance of Standard Deviation of Rtt Average Latency across all runs (10 to 100) for baseline vs ADQ (229-739)/739 * 100% = -69% Reduction in Variance

Application Device Queues (ADQ) – Redis* Open Source Database

Performance Results

REDUCES APPLICATION LATENCY

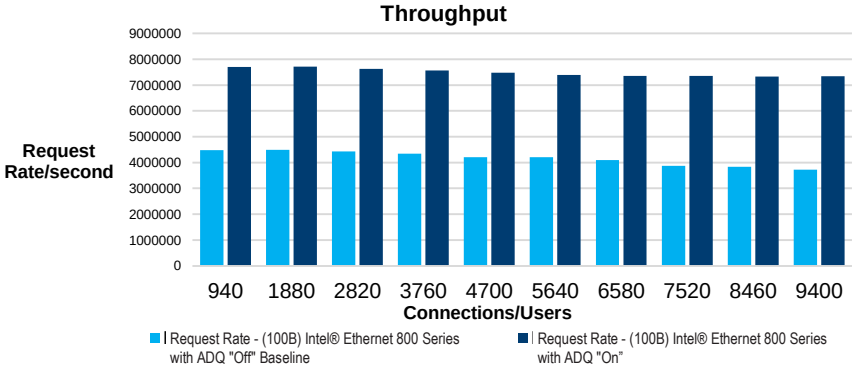


Lower is Better

>45% LATENCY REDUCTION WITH OPEN SOURCE REDIS

redis*

IMPROVES APPLICATION THROUGHPUT



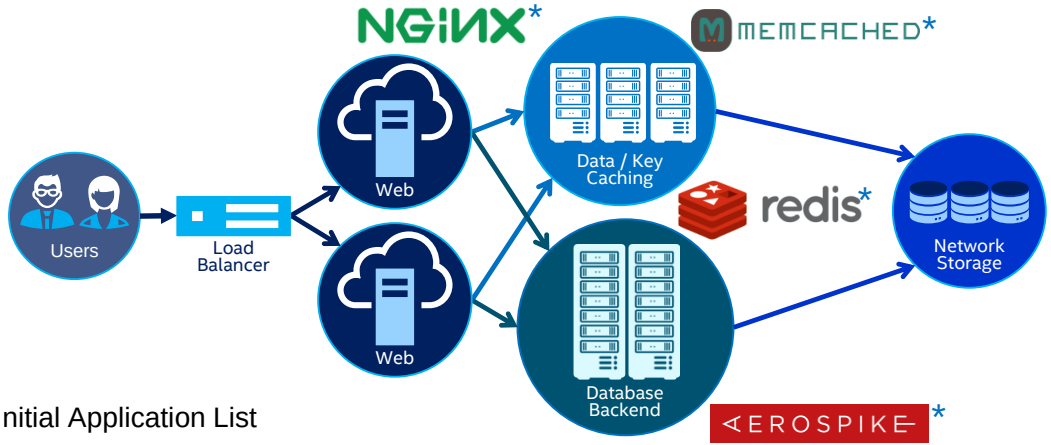
Higher is Better

>30% THROUGHPUT IMPROVEMENT WITH OPEN SOURCE REDIS

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 complete information visit http://www.intel.com/performance. Source: Performance results are based on Intel internal testing as of February 2019, and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure. Tests performed using Redis Open Source on 2nd Generation Intel® Xeon® Scalable processors and Intel® Ethernet 800 series 100GbE on Linux 4.19.18 kernel (see backup) Calculation: (new - old) / old x 100% Rtt Average Latency across all runs for baseline vs ADQ (382-1249)/1249 * 100% = -69% Reduction in Rtt Average Latency & Transaction Request Rate across all runs for baseline vs ADQ (79601-44345)/44345 * 100% = 80% Throughput Improvement

Application Device Queues (ADQ)[1]

Performance Improvements across Multiple Tiers



Initial Application List

NGINX

"NGINX is excited to collaborate with Intel on delivering a significant reduction in latency. Together we help to scale NGINX based cloud services with the new Intel® Ethernet 800 Series with Application Device Queues"

Christine Puccio VP, Global Strategic Alliances & Partnerships

EROSPIKE

"Getting useful insights in real-time out of Big Data comes with a set of major challenges such as predictable low latency and maximum throughput at the network layer. Aerospike, as always, is at the forefront of addressing these challenges. We expect the Intel® Ethernet 800 Series with Application Device Queues (ADQ) coupled with Aerospike Enterprise will help get predictable performance, higher data throughput and lower latency. We are pleased to work closely with Intel to bring this exciting new technology to our customers."

Srini Srinivasan, Founder and Chief Product Officer

Significantly Improves Predictability, Latency and Throughput

[1]Features & schedule are subject to change. All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Ongoing work: ADQ support in SPDK?

- Leverage the ADQ (application device queue) feature of Intel's 100Gb NIC (i.e., E810). Benefit: **High IOPS with improved tail latency.**
  - ADQ is an application specific queuing and steering technology that dedicates and isolates application specific hardware NIC queues.
  - These queues are then connected optimally to application specific threads of execution.
- Technique requirement:
  - Kernel & driver: Busy polling; Socket option for NAPI_ID (SO_INCOMING_NAPI_ID); symmetric polling; ….
  - Application: epoll threads watch the socket with same NAPI_ID
- Hardware:
  - Application level filtering & traffic shaping; Flow based queue steering and load balance.

Agenda

- Why NVMe-oF & Why SPDK?
- SPDK NVMe-oF development history & status
- SPDK TCP transport introduction
- Intel NIC update
- **Conclusion**

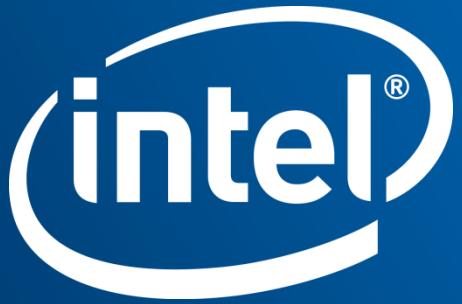
Conclusion

- SPDK NVMe-oF solution is well adopted by the industry. In this presentation, followings are introduced, i.e.,
  - The development status of SPDK NVMe-oF solution
  - SPDK TCP transport development status.
  - The status of Intel' NIC.
- Call for activity in community
  - Welcome to bug submission, idea discussion and patch submission for NVMe-oF

Conclusion: Further development area

Ingredients	Goal
Spec compliance	Continue following the NVMe-oF spec
Interoperability with kernel	Continue the interoperability test with Linux kernel solution.
Performance	Continue performance enhancements and integration with other solutions.
Advanced feature	Continuing extracting the common features from customers and put in the roadmap

Q & A ?


intel®