SPDK NVMe-oF TCP (Target & Initiator) Performance Report Release 19.10

Testing Date: November 2019

Performed by: Karol Latecki (karol.latecki@intel.com)
Maciej Wawryk (maciejx.wawryk@intel.com)

Acknowledgments:

John Kariuki (john.k.kariuki@intel.com)

Benjamin Walker (benjamin.walker@intel.com)

Seth Howell (seth.howell@intel.com)

Vishal Verma (vishal4.verma@intel.com)

Contents

Contents 2
Audience and Purpose 3
Test setup 4
- Target Configuration 4
- Initiator 1 Configuration 5
- Initiator 2 Configuration 5
- BIOS settings 6
- TCP configuration 6
- Kernel & BIOS spectre-meltdown information 6
- BIOS version information 6

Introduction to SPDK NVMe-oF (Target & Initiator) 8
Test Case 1: SPDK NVMe-oF TCP Target I/O core scaling 10
- 4k Random Read Results 14
- 4k Random Write Results 15
- 4k Random Read-Write Results 16
- Large Sequential I/O Performance 17
- Conclusions 21

Test Case 2: SPDK NVMe-oF TCP Initiator I/O core scaling 22
- 4k Random Read Results 25
- 4k Random Write Results 26
- 4k Random Read-Write Results 27
- Conclusions 28

Test Case 3: Linux Kernel vs. SPDK NVMe-oF TCP Latency 29
- SPDK vs Kernel NVMe-oF Target Latency Results 32
- SPDK vs Kernel NVMe-oF TCP Initiator Latency Results 33
- SPDK vs Kernel NVMe-oF Latency Results 34
- Conclusions 35

Test Case 4: NVMe-oF Performance with increasing # of connections 36
- 4k Random Read Results 38
- 4k Random Write Results 39
- 4k Random Read-Write Results 40
- Conclusions 42

Summary 43
Appendix A 44

Audience and Purpose

This report is intended for people who are interested in evaluating SPDK NVMe-oF (Target & Initiator) performance as compared to the Linux Kernel NVMe-oF (Target & Initiator). This report compares the performance and efficiency of the SPDK NVMe-oF Target and Initiator vs. the Linux Kernel NVMe-oF Target and Initiator. This report covers the TCP transport only.

The purpose of reporting these tests is not to imply a single "correct" approach, but rather to provide a baseline of well-tested configurations and procedures that produce repeatable results. This report can also be viewed as information regarding best known method/practice when performance testing SPDK NVMe-oF (Target & Initiator).

Test setup

Target Configuration

Item	Description
Server Platform	SuperMicro SYS-2029U-TN24R4T 
CPU	Intel® Xeon® Gold 6230 Processor (27.5MB L3, 2.10 GHz) Number of cores 20, number of threads 40
Memory	10 x 32GB Hynix HMA84GR7AFR4N-VK, DDR4, 2666MHz Total of 320GB Memory channel population:

P1	P2
P1-DIMMA1	P2-DIMMA1
P1-DIMMB1	P2-DIMMB1
P1-DIMMC1	P2-DIMMC1
P1-DIMMD1	P2-DIMMD1
P1-DIMME1	P2-DIMME1

Operating System	Fedora 29
BIOS	3.1a 07/19/2019
Linux kernel version	5.2.7-100.fc29
SPDK version	SPDK 19.10 (dfe1678b7)
Storage	**OS:** 1x 120GB Intel SSDSC2BB120G4 **Storage Target**: 16x Intel® P4600™ P4600x 2.0TB (FW: QDV10150) (8 on each CPU socket)
NIC	2x 100GbE Mellanox® ConnectX-5 NICs. Both ports connected. 1 NIC per CPU socket.

Initiator 1 Configuration

Item	Description
Server Platform	SuperMicro SYS-2028U TN24R4T+
CPU	Intel® Xeon® CPU E5-2699 v4 @ 2.20GHz (55MB Cache, 2.20 GHz) Number of cores 22, number of threads 44 per socket (Both sockets populated)
Memory	8 x 8GB Samsung M393A1G40EB1-CRC, DDR4, 2400MHz Total 64GBs Memory channel population:

P1	P2
P1-DIMMA1	P2-DIMME1
P1-DIMMB1	P2-DIMMF1
P1-DIMMC1	P2-DIMMG1
P1-DIMMD1	P2-DIMMH1

Operating System	Fedora 29
BIOS	3.1 06/08/2018
Linux kernel version	5.2.7-100.fc29
SPDK version	SPDK 19.10 (e660235c9)
Storage	**OS:** 1x 240GB INTEL SSDSC2BB240G6
NIC	1x 100GbE Mellanox® ConnectX-4 NIC. Both ports connected to Target server.

Initiator 2 Configuration

Item	Description
Server Platform	SuperMicro SYS-2028U TN24R4T+
CPU	Intel® Xeon® CPU E5-2699 v4 @ 2.20GHz (55MB Cache, 2.20 GHz) Number of cores 22, number of threads 44 per socket (Both sockets populated)
Memory	8 x 8GB Samsung M393A1G40EB1-CRC, DDR4, 2400MHz Total 64GBs Memory channel population:

P1	P2
P1-DIMMA1	P2-DIMME1
P1-DIMMB1	P2-DIMMF1
P1-DIMMC1	P2-DIMMG1
P1-DIMMD1	P2-DIMMH1

Operating System	Fedora 29
BIOS	3.1 06/08/2018
Linux kernel version	5.2.7-100.fc29

SPDK version	SPDK 19.10 (e660235c9)
Storage	**OS:** 1x 240GB INTEL SSDSC2BB240G6
NIC	1x 100GbE Mellanox® ConnectX-4 NIC. Both ports connected to Target server.

BIOS settings

Item	Description
BIOS ***(Applied to all 3 systems)***	Hyper threading Enabled CPU Power and Performance Policy: • "Extreme Performance" for Target • "Performance" for Initiators CPU C-state No Limit CPU P-state Enabled Enhanced Intel® SpeedStep® Tech Enabled Turbo Boost Enabled

TCP configuration

Note that the SPDK NVMe-oF target and initiator use the Linux Kernel TCP stack. We tuned the Linux Kernel TCP stack for storage workloads over 100 Gbps NIC by settings the following parameters using sysctl:

```
# Set 256MB buffers
net.core.rmem_max = 268435456
net.core.wmem_max = 268435456
# Increase autotuning TCP buffer limits
# min, max and default settings
# auto-tuning allowed to 128MB
net.ipv4.tcp_rmem = 4096 87380 134217728
net.ipv4.tcp_wmem = 4096 65536 134217728
# MTU probing for Jumbo Frames
net.ipv4.tcp_mtu_probing = 1
```

Additionally, the NIC ports were configured to use Jumbo Frames using network-scripts (/etc/sysconfig/network-scripts for RHEL-based distributions) and setting MTU=9000.

Kernel & BIOS spectre-meltdown information

All three systems used Fedora 5.2.7-100.fc29 kernel from DNF repository with default patches for spectre-meltdown enabled.

The BIOS on all systems was updated to post spectre-meltdown versions as well.

BIOS version information

During testing it was observed that using BIOS version 3.1 for platform SYS-2029U-TN24R4T with the server power policy set to "Maximum performance" or "Extreme performance" resulted in a Kernel

panic. We learned that enabling one of these settings on this particular combination of CPU, BIOS version and OS / Kernel version caused a kernel panic during OS boot. This issue was resolved by using latest BIOS version 3.1a (07/19/2019) for SYS-2029U-TN24R4T server platform.

Introduction to SPDK NVMe-oF (Target & Initiator)

The NVMe over Fabrics (NVMe-oF) protocol extends the parallelism and efficiencies of the NVM Express* (NVMe) block protocol over network fabrics such as RDMA (iWARP, RoCE), InfiniBand™, Fibre Channel, TCP and Intel® Omni-Path. SPDK provides both a user space NVMe-oF target and initiator that extends the software efficiencies of the rest of the SPDK stack over the network. The SPDK NVMe-oF target uses the SPDK user-space, polled-mode NVMe driver to submit and complete I/O requests to NVMe devices which reduces the software processing overhead. Likewise, it pins connections to CPU cores to avoid synchronization and cache thrashing so that the data for those connections is kept close to the CPU.

The SPDK NVMe-oF target and initiator uses the underlying transport layer API which in case of TCP are POSIX sockets. In case of RDMA-capable NICs Infiniband/RDMA verbs API is used which should work on all flavors of RDMA transports, but is currently tested against RoCEv2, iWARP, and Omni-Path NICs. SPDK provides a user-space, lockless, polled-mode NVMe-oF initiator. The host system uses the initiator to establish a connection and submit I/O requests to an NVMe subsystem within an NVMe-oF target. NVMe subsystems contain namespaces, each of which maps to a single block device exposed via SPDK's bdev layer. SPDK's bdev layer is a block device abstraction layer and general-purpose block storage stack akin to what is found in many operating systems. Using the bdev interface completely decouples the storage media from the front-end protocol used to access storage. Users can build their own virtual bdevs that provide complex storage services and integrate them with the SPDK NVMe-oF target with no additional code changes. There can be many subsystems within an NVMe-oF target and each subsystem may hold many namespaces. Subsystems and namespaces can be configured dynamically via a JSON-RPC interface.

Figure 1 shows a high-level schematic of the systems used for testing in the rest of this report. The set up consists of three systems (two used as initiators and one used as the target). The NVMe-oF target is connected to both initiator systems point-to-point using QSFP28 cables without any switches. The target system has sixteen Intel® SSD DC P4600 SSDs which were used as block devices for NVMe-oF subsystems and two 100GbE Mellanox ConnectX®-5 NICs that provide up to 200GbE of network bandwidth. Each Initiator system has one Mellanox ConnectX®-4 100GbE NIC connected directly to the target without any switch.

One goal of this report was to make clear the advantages and disadvantages inherent to the design of the SPDK NVMe-oF components. These components are written using techniques such as run-to completion, polling, and asynchronous I/O. The report covers four real-world use cases.

For performance benchmarking the fio tool is used with two storage engines:

1) Linux Kernel libaio engine
2) SPDK bdev engine

Performance numbers reported are aggregate I/O per second, average latency, and CPU utilization as a percentage for various scenarios. Aggregate I/O per second and average latency data is reported from fio and CPU utilization was collected using sar (systat).

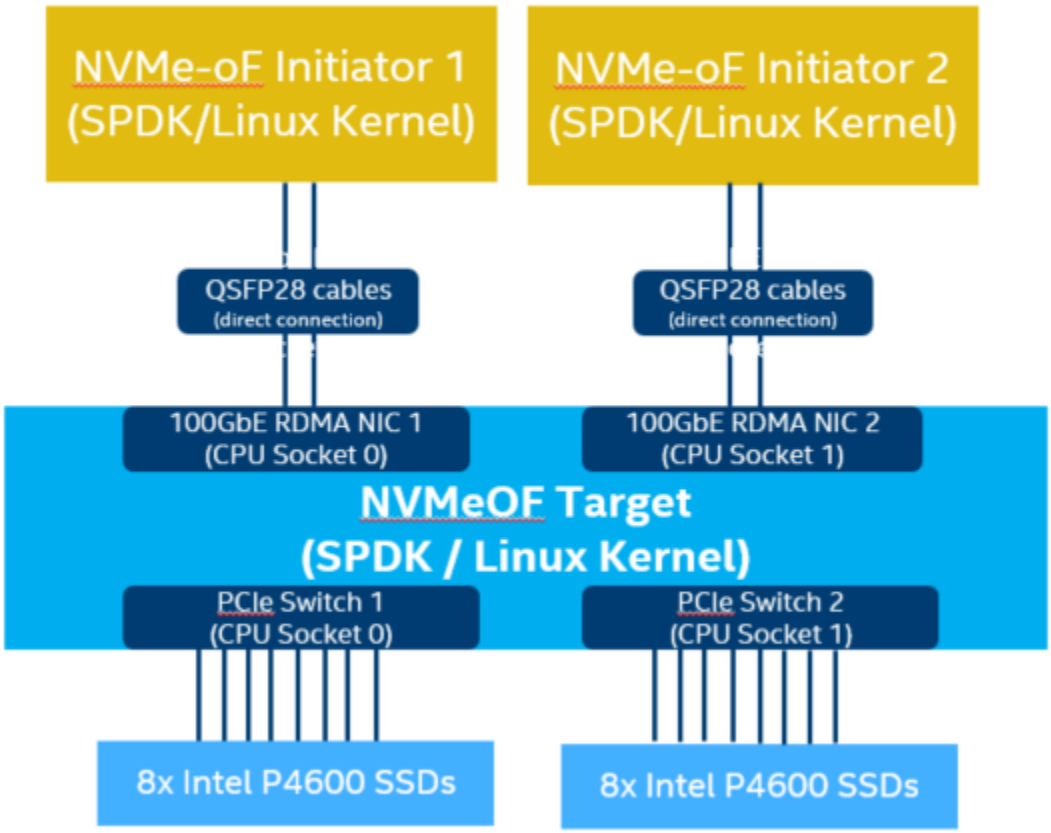

Figure 1: High-Level NVMe-oF TCP performance testing setup

Test Case 1: SPDK NVMe-oF TCP Target I/O core scaling

This test case was performed in order to understand the performance of the SPDK TCP NVMe-oF target with I/O core scaling.

The SPDK NVMe-oF TCP target was configured to run with 16 NVMe-oF subsystems. Each NVMe-oF subsystem ran on top of an individual NVMe bdev backed by a single Intel P4600 device. Each of the 2 host systems was connected to 8 NVMe-oF subsystems which were exported by the SPDK NVMe-oF Target over 1x 100GbE NIC. The SPDK bdev FIO plugin was used to target 8 NVMe-oF bdevs on each of the host. The SPDK Target was configured to use 1, 4, 8, 12, 16, 20, 24, 32, 36 and 40 CPU cores. We ran the following workloads on each initiator:

- 4KB 100% Random Read
- 4KB 100% Random Write
- 4KB Random 70% Read 30% Write

We scaled the fio jobs using fio configuration parameter numjob=3, in order to generate more I/O requests.

For detailed configuration please refer to the table below. The actual SPDK NVMe-oF configuration was performed using JSON-RPC and the table contains the sequence of commands used by spdk/scripts/rpc.py script rather than a configuration file. The SPDK NVMe-oF Initiator (bdev fio_plugin) still uses plain configuration files.

Each workload was run three times at each CPU count and the reported results are the average of the 3 runs. For workloads which need preconditioning (4KB rand write and 4KB 70% read 30% write we ran preconditioning once before running all of the workload to ensure that NVMe devices reached higher IOPS so that we can saturate the network.

Item	Description
Test Case	Test SPDK NVMe-oF Target I/O core scaling
SPDK NVMe-oF Target configuration	All the commands below were executed with spdk/scripts/rpc.py script. construct_nvme_bdev -t PCIe -b Nvme0 -a 0000:60:00.0 construct_nvme_bdev -t PCIe -b Nvme1 -a 0000:61:00.0 construct_nvme_bdev -t PCIe -b Nvme2 -a 0000:62:00.0 construct_nvme_bdev -t PCIe -b Nvme3 -a 0000:63:00.0 construct_nvme_bdev -t PCIe -b Nvme4 -a 0000:64:00.0 construct_nvme_bdev -t PCIe -b Nvme5 -a 0000:65:00.0 construct_nvme_bdev -t PCIe -b Nvme6 -a 0000:66:00.0 construct_nvme_bdev -t PCIe -b Nvme7 -a 0000:67:00.0 construct_nvme_bdev -t PCIe -b Nvme8 -a 0000:b5:00.0 construct_nvme_bdev -t PCIe -b Nvme9 -a 0000:b6:00.0 construct_nvme_bdev -t PCIe -b Nvme10 -a 0000:b7:00.0

```
construct_nvme_bdev -t PCIe -b Nvme11 -a 0000:b8:00.0
construct_nvme_bdev -t PCIe -b Nvme12 -a 0000:b9:00.0
construct_nvme_bdev -t PCIe -b Nvme13 -a 0000:ba:00.0
construct_nvme_bdev -t PCIe -b Nvme14 -a 0000:bb:00.0
construct_nvme_bdev -t PCIe -b Nvme15 -a 0000:bc:00.0

nvmf_create_transport -t TCP
(creates TCP transport layer with default values:
trtype: "TCP"
max_queue_depth: 128
max_qpairs_per_ctrlr: 64
in_capsule_data_size: 4096
max_io_size: 131072
"io_unit_size": 131072,
max_aq_depth: 128
num_shared_buffers: 4096
buf_cache_size: 32,
"c2h_success": true,
"dif_insert_or_strip": false,
"sock_priority": 0
)

nvmf_subsystem_create nqn.2018-09.io.spdk:cnode1 -s SPDK001 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode2 -s SPDK002 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode3 -s SPDK003 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode4 -s SPDK004 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode5 -s SPDK005 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode6 -s SPDK006 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode7 -s SPDK007 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode8 -s SPDK008 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode9 -s SPDK009 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode10 -s SPDK0010 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode11 -s SPDK0011 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode12 -s SPDK0012 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode13 -s SPDK0013 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode14 -s SPDK0014 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode15 -s SPDK0015 -a -m 8
nvmf_subsystem_create nqn.2018-09.io.spdk:cnode16 -s SPDK0016 -a -m 8

nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode1 Nvme0n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode2 Nvme1n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode3 Nvme2n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode4 Nvme3n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode5 Nvme4n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode6 Nvme5n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode7 Nvme6n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode8 Nvme7n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode9 Nvme8n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode10 Nvme9n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode11 Nvme10n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode12 Nvme11n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode13 Nvme12n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode14 Nvme13n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode15 Nvme14n1
nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode16 Nvme15n1
```

	nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode1 -t tcp -f ipv4 -s 4420 -a 20.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode2 -t tcp -f ipv4 -s 4420 -a 20.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode3 -t tcp -f ipv4 -s 4420 -a 20.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode4 -t tcp -f ipv4 -s 4420 -a 20.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode5 -t tcp -f ipv4 -s 4420 -a 20.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode6 -t tcp -f ipv4 -s 4420 -a 20.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode7 -t tcp -f ipv4 -s 4420 -a 20.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode8 -t tcp -f ipv4 -s 4420 -a 20.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode9 -t tcp -f ipv4 -s 4420 -a 10.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode10 -t tcp -f ipv4 -s 4420 -a 10.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode11 -t tcp -f ipv4 -s 4420 -a 10.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode12 -t tcp -f ipv4 -s 4420 -a 10.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode13 -t tcp -f ipv4 -s 4420 -a 10.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode14 -t tcp -f ipv4 -s 4420 -a 10.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode15 -t tcp -f ipv4 -s 4420 -a 10.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode16 -t tcp -f ipv4 -s 4420 -a 10.0.1.1
SPDK NVMe-oF Initiator - FIO plugin configuration	**BDEV.conf** [Nvme] TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode1" Nvme0 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode2" Nvme1 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode3" Nvme2 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode4" Nvme3 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode5" Nvme4 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode6" Nvme5 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode7" Nvme6 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode8" Nvme7 **FIO.conf** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_conf=/tmp/spdk/bdev.conf thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={1, 8, 16, 32, 64} time_based=1 numjobs=3 ramp_time=60 runtime=300 [filename0] filename=Nvme0n1 [filename1] filename=Nvme1n1 [filename2] filename=Nvme2n1 [filename3] filename=Nvme3n1 [filename4]

	filename=Nvme4n1 [filename5] filename=Nvme5n1 [filename6] filename=Nvme6n1 [filename7] filename=Nvme7n1

4k Random Read Results

Test Result: 4K 100% Random Read IOPS, QD=64

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1128.26	288.8	10632.6
4 cores	4412.28	1129.5	2718.3
8 cores	8295.54	2123.6	1452.7
12 cores	11610.08	2972.2	1037.3
16 cores	14780.72	3783.9	809.7
20 cores	15040.61	3850.4	794.0
24 cores	16605.15	4250.9	718.9
28 cores	17759.47	4546.4	672.0
32 cores	17602.98	4506.4	678.6
36 cores	16993.95	4350.4	706.1
40 cores	19164.14	4906.0	622.4

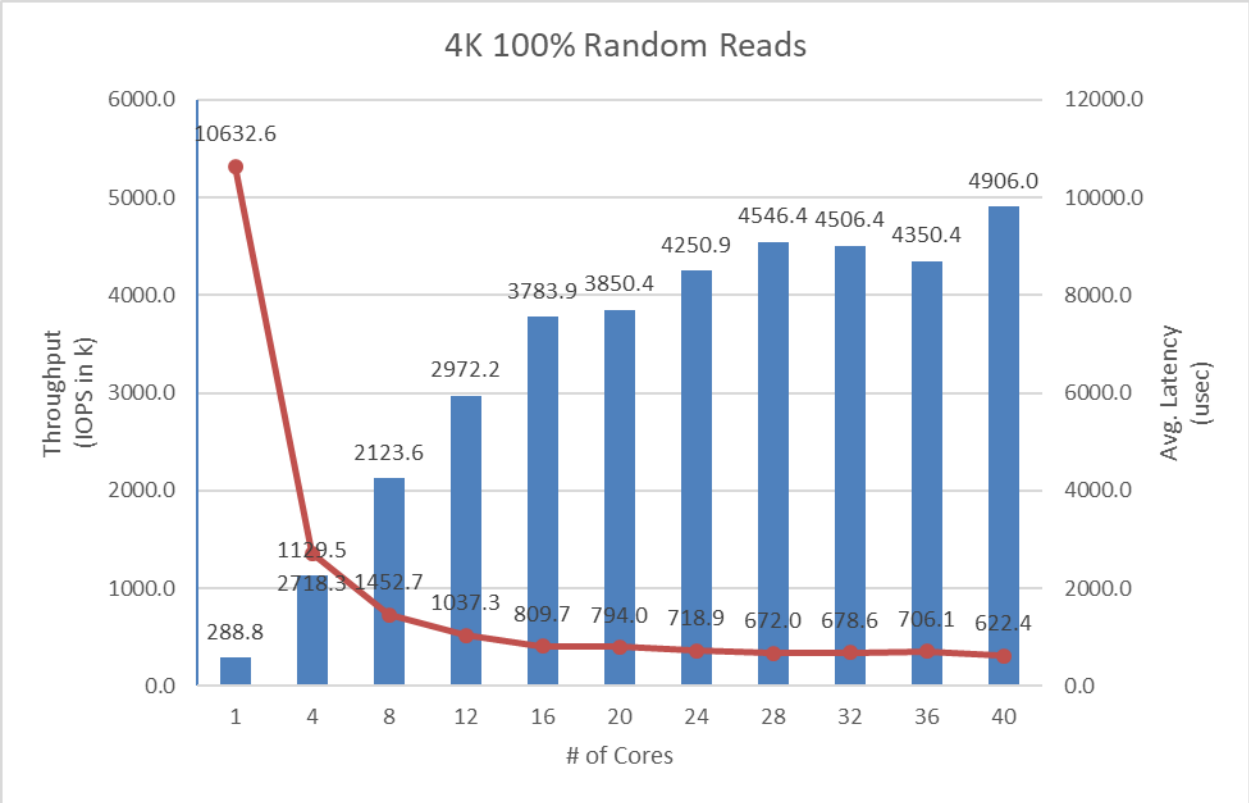

Figure 2: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Read workload at QD = 64

4k Random Write Results

Test Result: 4K 100% Random Writes IOPS, QD=64

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	877.4	224.6	13665.8
4 cores	3421.7	876.0	3498.0
8 cores	6649.7	1702.3	1794.8
12 cores	8589.4	2198.9	1381.7
16 cores	10055.6	2574.2	1177.8
20 cores	10681.9	2734.5	1108.2
24 cores	10675.2	2732.8	1110.0
28 cores	11722.3	3000.9	1013.5
32 cores	11773.8	3014.1	1007.4
36 cores	11773.8	3014.1	1007.4
40 cores	11955.6	3060.6	990.7

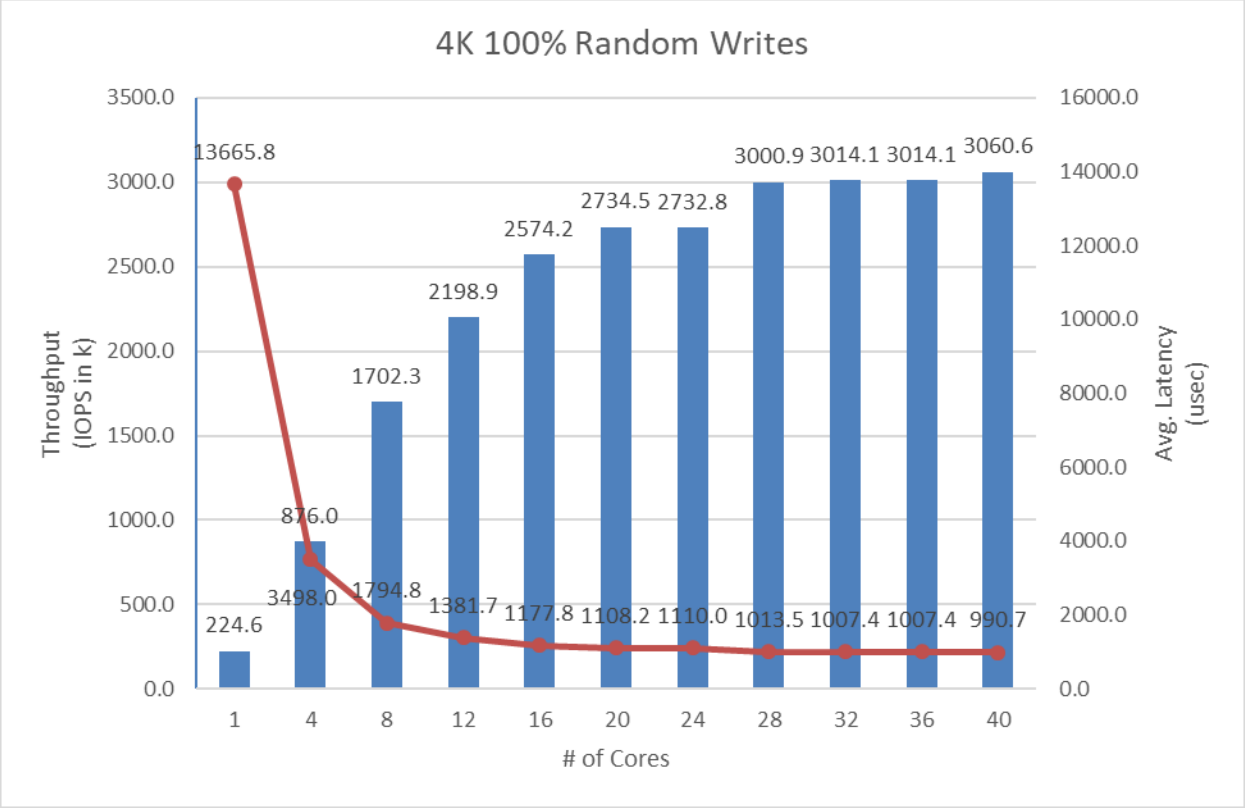

Figure 3: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write Workload at QD=64

4k Random Read-Write Results

Test Result: 4K Random Read/Write 70%/30% IOPS, QD=64

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1001.4	256.3	11979.5
4 cores	4026.6	1030.8	2975.9
8 cores	7702.7	1971.9	1555.4
12 cores	10621.8	2719.2	1129.7
16 cores	12860.2	3292.2	930.8
20 cores	13616.3	3485.8	877.4
24 cores	15111.4	3868.5	789.4
28 cores	15279.0	3911.4	780.8
32 cores	15803.7	4045.7	754.0
36 cores	16821.2	4306.2	707.7
40 cores	15320.3	3922.0	783.4

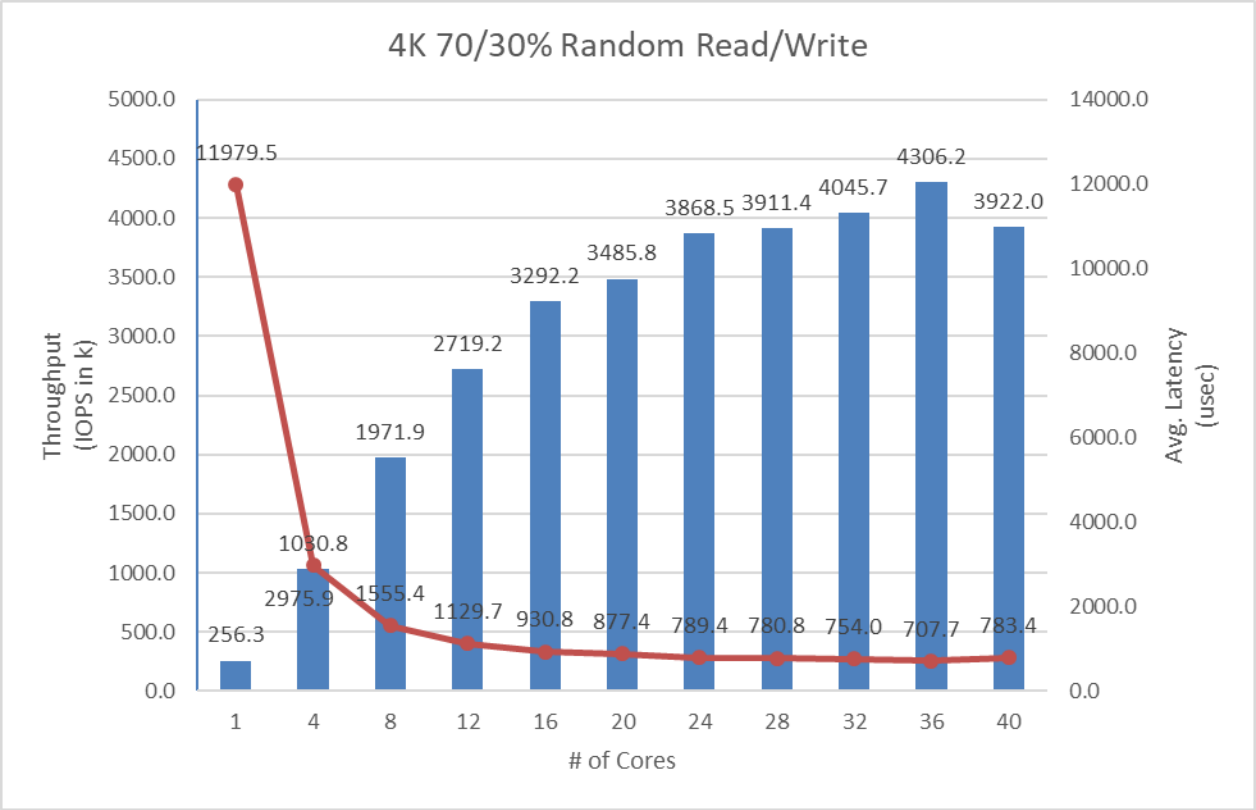

Figure 4: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 4KB Random 70/30 read/write workload at QD=64

Large Sequential I/O Performance

We measured the performance of large block I/O workloads by performing sequential I/Os of size 128K s at queue depth 4. We used iodepth=4 because higher queue depth resulted in negligible bandwidth gain and a significant increase in the latency. The rest of the FIO configuration is similar to the 4K test case in the previous part of this document.

Test Result: 128K 100% Sequential Reads, QD=4

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4056.95	32.5	3976.3
4 cores	10394.81	83.2	1540.8
8 cores	13024.24	104.2	1228.9
12 cores	15534.21	124.3	1031.6
16 cores	17454.89	139.6	916.5
20 cores	17424.48	139.4	918.3
24 cores	16722.94	133.8	959.1
28 cores	17888.10	143.1	900.3
32 cores	19031.09	152.2	840.3
36 cores	19102.97	152.8	837.3
40 cores	18931.42	151.5	844.7

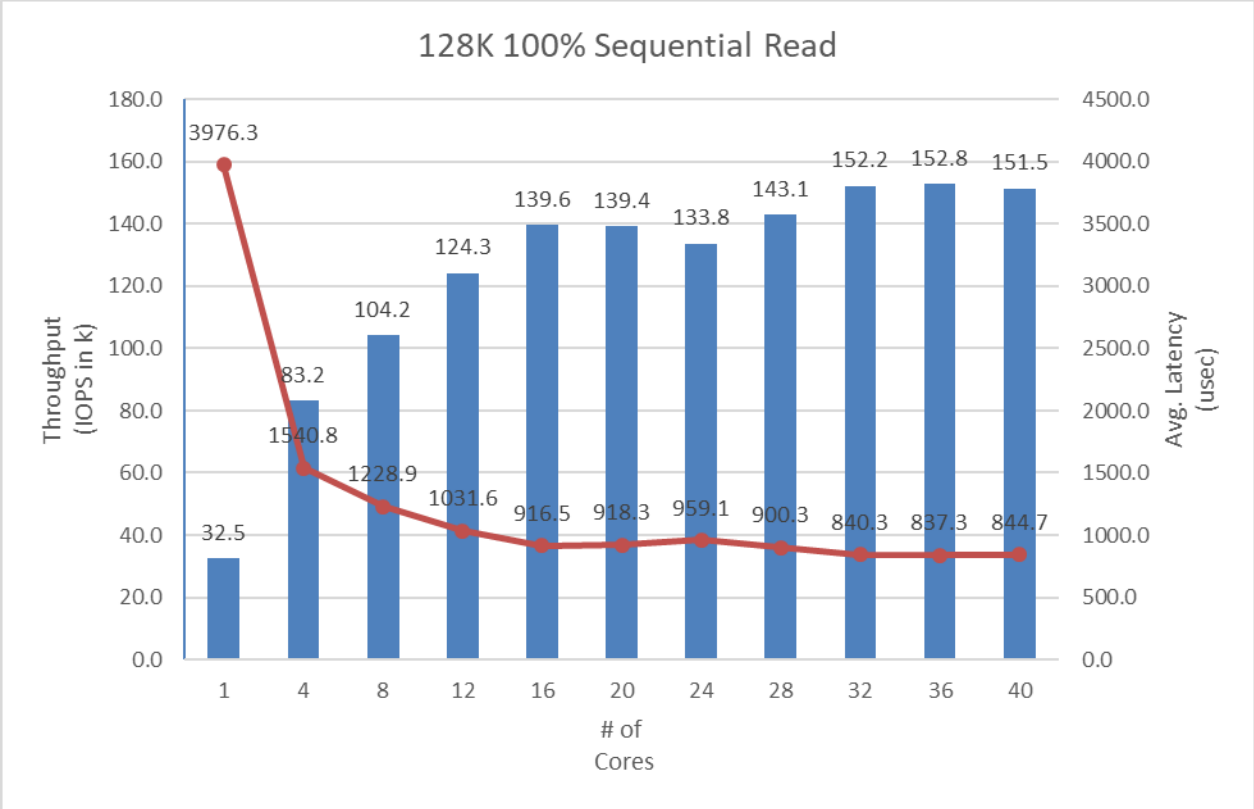

Figure 5: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 128KB 100% Sequential Read Workload at QD=4 and initiator FIO numjobs=2

Test Result: 128K 100% Sequential Writes, QD=4

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	2688.76	21.5	5959.6
4 cores	9517.05	76.1	1681.8
8 cores	14315.91	114.5	1120.6
12 cores	16395.60	131.2	977.7
16 cores	17706.50	141.7	907.7
20 cores	17921.98	143.4	894.5
24 cores	17967.11	143.7	893.7
28 cores	16939.03	135.5	973.4
32 cores	18601.91	148.8	892.9
36 cores	17425.27	139.4	929.4
40 cores	15812.46	126.5	1055.1

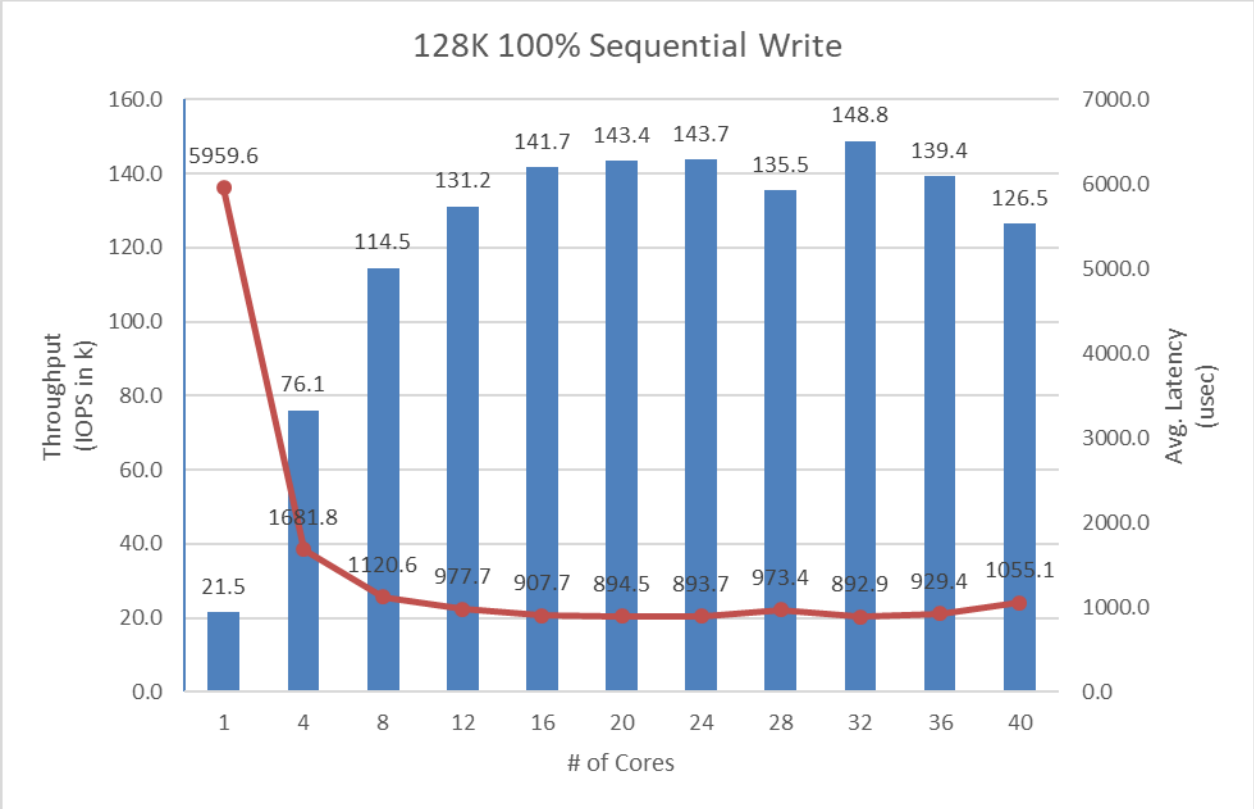

Figure 6: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 128KB 100% Sequential Write Workload at QD=4 and Initiator FIO numjobs=2

Test Result: 128K Sequential 70% Reads 30% Writes, QD=4

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	3241.50	25.9	4954.8
4 cores	10162.53	81.3	1577.2
8 cores	13218.42	105.7	1211.3
12 cores	15111.09	120.9	1086.7
16 cores	15132.42	121.1	1058.0
20 cores	15308.28	122.5	1049.0
24 cores	16473.76	131.8	972.8
28 cores	17432.35	139.5	920.5
32 cores	18190.14	145.5	878.6
36 cores	15394.35	123.2	1071.1
40 cores	15603.66	124.8	1056.8

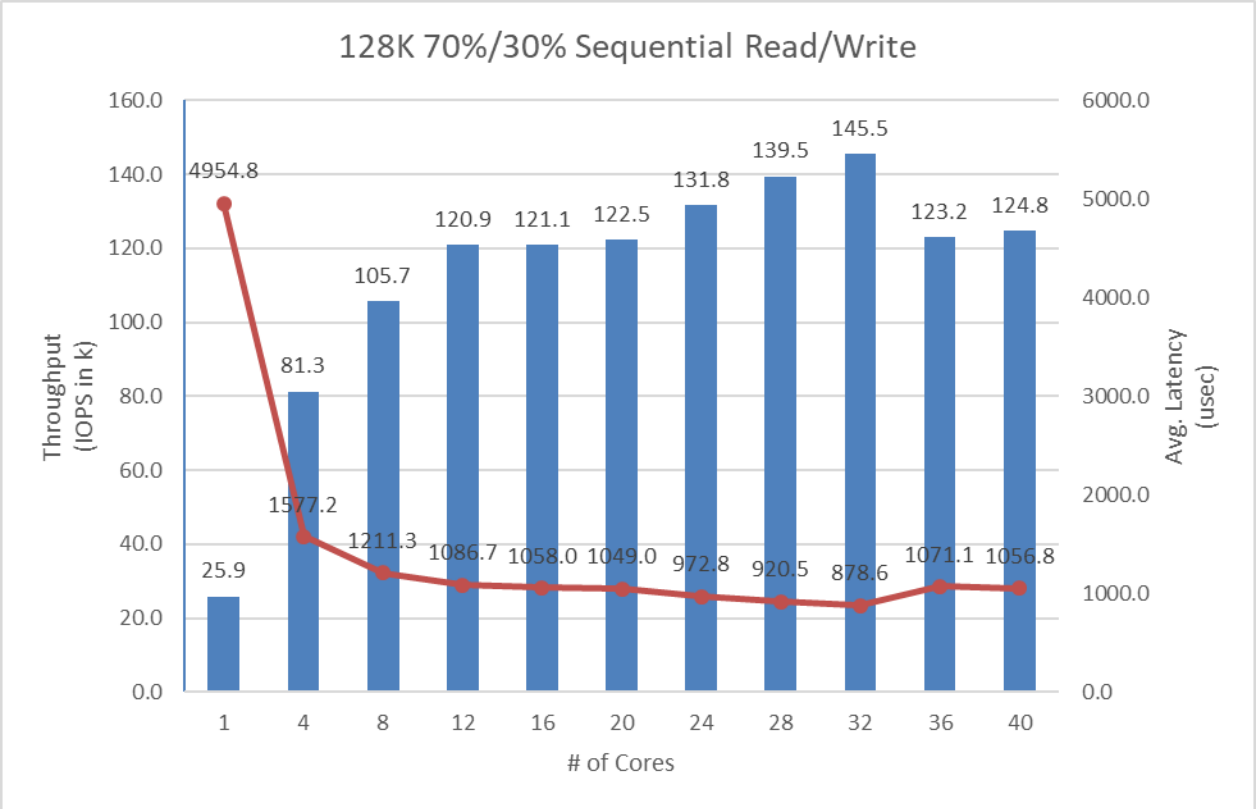

Figure 7: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 128KB Sequential 70% Read 30% Write Workload at QD=4 and Initiator FIO numjobs=2

Conclusions

1. The SPDK NVMe-oF TCP Target IOPS throughput scales up almost linearly up to 12-16 CPU cores for 4K random workloads. Beyond that IOPS gains become smaller and non-linear.

2. The best trade-off between CPU Efficiency and Network Saturation was observed when the Target was configured with 16 CPU cores. The performance we achieved fully saturated a 100Gbps NIC connection between Target and Initiator for Random Read and Random Read/Write workloads. We added another 16 CPU cores but could not saturate a 200Gbps Network.

3. For the 4k Random Write workload, NVMe drives were completely saturated.

4. For other 4k Random workloads (Random Read, Random Read-Write) we were unable to fully saturate the total available NIC bandwidth (200Gbps) or Target's PCIe switches throughput (around 5.8M IOPS) due to some other unidentified bottleneck.

5. For the Sequential 128k Read and Write workloads, the IOPS throughput scaled up with addition of CPU cores up to 16 CPU cores and remained constant as we added more CPU cores. The network bandwidth reported by FIO was about 150Gbps, which is close to network saturation considering the network overhead.

Test Case 2: SPDK NVMe-oF TCP Initiator I/O core scaling

This test case was performed in order to understand the performance of SPDK NVMe-oF TCP Initiator as the number of CPU cores is scaled up.

The test setup for this test case is slightly different than the set up described in introduction chapter, we used just a single SPDK NVMe-oF TCP Initiator to make it easier to understand of how the number of CPUs affects initiator IOPS throughput. The Initiator was connected to Target server with 100 Gbps network link.

The SPDK NVMe-oF TCP Target was configured similarly as in test case 1, using 20 cores. We used 20 CPU cores based on results of the previous test case which show that the target can easily serve over 3 million IOPS; that is enough IOPS to saturate 100 Gbps network connection

The SPDK bdev FIO plugin was used to target 16 individual NVMe-oF subsystems exported by the Target. The number of total CPU threads used by the FIO process was managed by setting the FIO job sections and numjobs parameter and ranged from 1 to 40 CPUs. For detailed FIO job configuration see table below. FIO was run with following workloads:

- 4KB 100% Random Read
- 4KB 100% Random Write
- 4KB Random 70% Read 30% Write

Item	Description
Test Case	Test SPDK NVMe-oF TCP Initiator I/O core scaling
SPDK NVMe-oF Target configuration	Same as in Test Case #1, using 20 CPU cores.
SPDK NVMe-oF Initiator 1 - FIO plugin configuration	**BDEV.conf** [Nvme] TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode1" Nvme0 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode2" Nvme1 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode3" Nvme2 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode4" Nvme3 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode5" Nvme4 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode6" Nvme5 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode7" Nvme6 TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode8" Nvme7 TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode1" Nvme8 TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode2" Nvme9 TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode3" Nvme10 TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode4" Nvme11 TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode5" Nvme12 TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode6" Nvme13 TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode7" Nvme14 TransportId "trtype:TCP adrfam:IPv4 traddr:10.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode8" Nvme15

	FIO.conf **For 1 CPU initiator configuration:** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_conf=/tmp/spdk/bdev.conf thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={32, 64, 128, 256} time_based=1 ramp_time=60 runtime=300 numjobs=1 [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 filename=Nvme4n1 filename=Nvme5n1 filename=Nvme6n1 filename=Nvme7n1 filename=Nvme8n1 filename=Nvme9n1 filename=Nvme10n1 filename=Nvme11n1 filename=Nvme12n1 filename=Nvme13n1 filename=Nvme14n1 filename=Nvme15n1
	FIO.conf **For X*4 CPU (up to 40) initiator configuration:** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_conf=/tmp/spdk/bdev.conf thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={32, 64, 128, 256} time_based=1 ramp_time=60 runtime=300 numjobs=X

	[filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 [filename1] filename=Nvme4n1 filename=Nvme5n1 filename=Nvme6n1 filename=Nvme7n1 [filename2] filename=Nvme8n1 filename=Nvme9n1 filename=Nvme10n1 filename=Nvme11n1 [filename3] filename=Nvme12n1 filename=Nvme13n1 filename=Nvme14n1 filename=Nvme15n1

4k Random Read Results

Test Result: 4K 100% Random Read, QD=64

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	505.63	129.4	481.7
4 cores	2180.42	558.2	455.5
8 cores	4443.42	1137.5	441.8
12 cores	5738.94	1469.2	516.4
16 cores	7635.36	1954.6	516.2
20 cores	9309.80	2383.3	529.4
24 cores	10507.20	2689.8	563.6
28 cores	11085.17	2837.8	624.6
32 cores	10539.57	2698.1	752.8
36 cores	10708.10	2741.3	834.6
40 cores	10466.69	2679.5	950.0

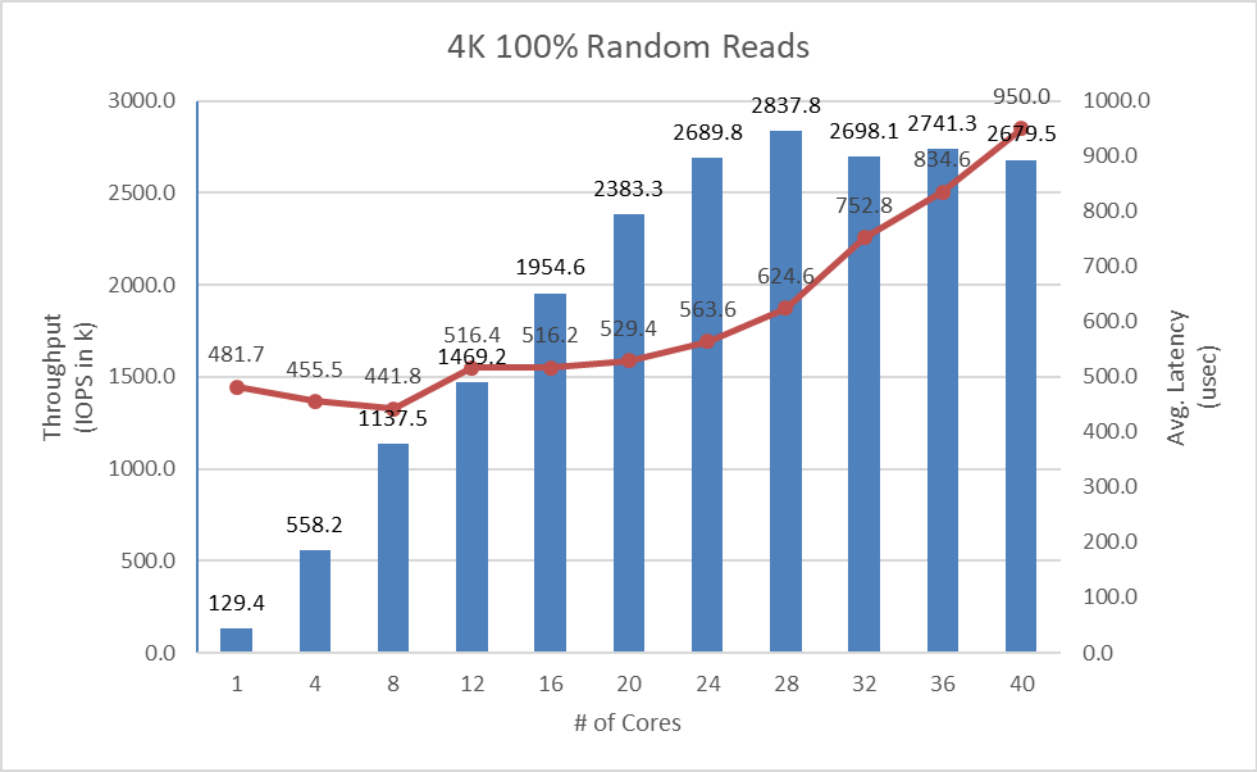

Figure 8: SPDK NVMe-oF TCP Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Read QD=64 workload

4k Random Write Results

Test Result: 4K 100% Random Write, QD=64

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	672.26	172.1	357.6
4 cores	3048.06	780.3	317.3
8 cores	5266.33	1348.2	370.9
12 cores	6934.66	1775.3	424.0
16 cores	8423.74	2156.5	467.2
20 cores	8825.77	2259.4	560.4
24 cores	8614.80	2205.4	691.4
28 cores	8231.85	2107.3	846.1
32 cores	8480.08	2170.9	939.4
36 cores	8171.61	2091.9	1097.4
40 cores	8177.83	2093.5	1218.8

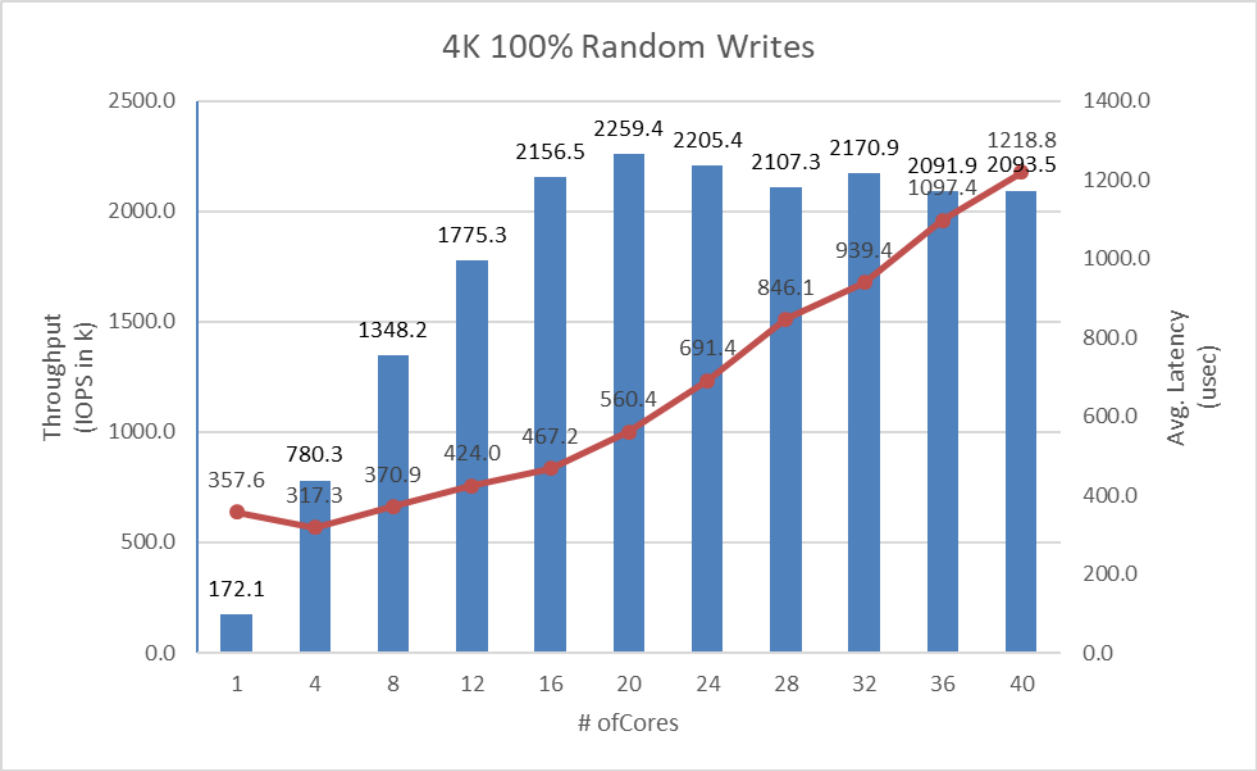

Figure 9: SPDK NVMe-oF TCP Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write Workload at QD=64

4k Random Read-Write Results

Test Result: 4K 70% Random Read 30% Random Write, QD=64

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	550.44	140.9	440.6
4 cores	2351.35	601.9	418.2
8 cores	4429.98	1134.1	442.5
12 cores	6105.67	1563.0	483.2
16 cores	7358.33	1883.7	536.0
20 cores	8471.92	2168.8	583.1
24 cores	9397.78	2405.8	631.9
28 cores	9886.86	2531.0	702.0
32 cores	10023.76	2566.1	792.9
36 cores	10040.79	2570.4	890.9
40 cores	10088.01	2582.5	986.2

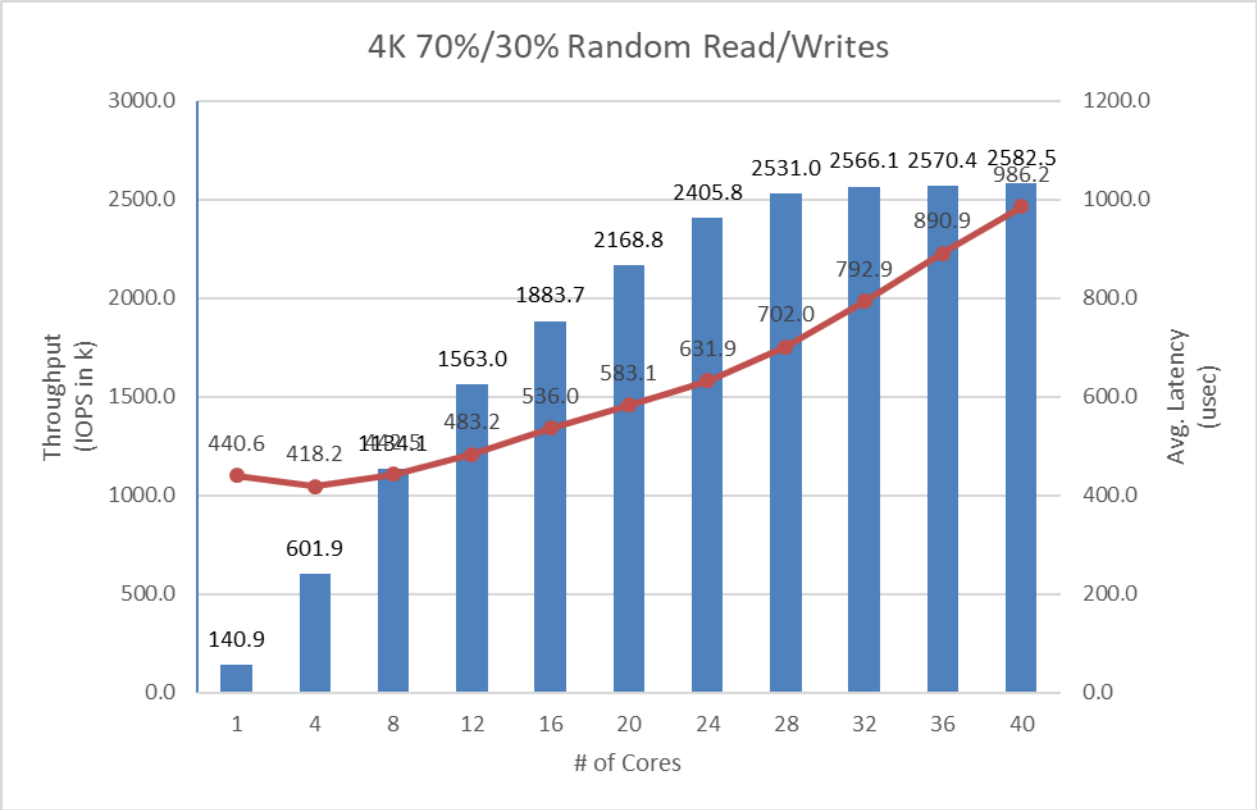

Figure 10: SPDK NVMe-oF TCP Initiator I/O core scaling: IOPS vs. Latency while running 4KB Random 70% Read 30% Write Workload at QD=64

Conclusions

1. The SPDK NVMe-oF TCP Initiator IOPS throughput scales almost linearly until the network is almost saturated for 4K Random Read and 4KB 70/30 Random Read/Write workloads.

2. As the IOPS throughput gets closer to the Network bandwidth the performance improvement is not linear.

3. The 4K Random Write workload does not saturate NIC or disks capabilities. There is a noticeable, slight degradation (by about 200k IOPS) as compared to SPDK 19.07.

Test Case 3: Linux Kernel vs. SPDK NVMe-oF TCP Latency

This test case was designed to understand latency characteristics of SPDK NVMe-oF TCP Target and Initiator vs. the Linux Kernel NVMe-oF TCP Target and Initiator implementations on a single NVMe-oF subsystem. The average I/O latency and p99 latency was compared between SPDK NVMe-oF (Target/Initiator) vs. Linux Kernel (Target/Initiator). Both SPDK and Kernel NVMe-oF Targets were configured to run on a single core, with a single NVMe-oF subsystem backed by a *Null Block Device*. The null block device (bdev) was chosen as the backend block device to eliminate the media latency during these tests.

Item	Description
Test Case	Linux Kernel vs. SPDK NVMe-oF Latency
	Test configuration
SPDK NVMe-oF Target configuration	All the commands below were executed with spdk/scripts/rpc.py script. nvmf_create_transport -t TCP (creates TCP transport layer with default values: trtype: "TCP" max_queue_depth: 128 max_qpairs_per_ctrlr: 64 in_capsule_data_size: 4096 max_io_size: 131072 io_unit_size: 8192 max_aq_depth: 128 num_shared_buffers: 4096 buf_cache_size: 32) construct_null_bdev Nvme0n1 10240 4096 nvmf_subsystem_create nqn.2018-09.io.spdk:cnode1 -s SPDK001 -a -m 8 nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode1 Nvme0n1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode1 -t tcp -f ipv4 -s 4420 -a 20.0.0.1

Kernel NVMe-oF Target configuration	Target configuration file loaded using nvmet-cli tool.
	{
	"ports": [
	{
	"addr": {
	"adrfam": "ipv4",
	"traddr": "20.0.0.1",
	"trsvcid": "4420",
	"trtype": "tcp"
	},
	"portid": 1,
	"referrals": [],
	"subsystems": [
	"nqn.2018-09.io.spdk:cnode1"
	]
	}

	]
	"hosts": [],
	"subsystems": [
	{
	"allowed_hosts": [],
	"attr": {
	"allow_any_host": "1",
	"version": "1.3"
	},
	"namespaces": [
	{
	"device": {
	"path": "/dev/nullb0",
	"uuid": "621e25d2-8334-4c1a-8532-b6454390b8f9"
	},
	"enable": 1,
	"nsid": 1
	}
	],
	"nqn": "nqn.2018-09.io.spdk:cnode1"
	}
	]
	}

FIO configuration	
SPDK NVMe-oF Initiator FIO plugin configuration	**BDEV.conf** [Nvme] TransportId "trtype:TCP adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode1" Nvme0 **FIO.conf** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_conf=/tmp/spdk/bdev.conf thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth=1 time_based=1 ramp_time=60 runtime=300 [filename0] filename=Nvme0n1
Kernel initiator configuration	**Device config** Done using nvme-cli tool. modprobe nvme-fabrics nvme connect –n nqn.2018-09.io.spdk:cnode1 –t tcp –a 20.0.0.1 –s 4420 **FIO.conf** [global] ioengine=libaio thread=1

	group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth=1 time_based=1 numjobs=1 ramp_time=60 runtime=300 [filename0] filename=/dev/nvme0n1

SPDK vs Kernel NVMe-oF TCP Target Latency Results

This following data was collected using the Linux Kernel initiator against both SPDK & Linux Kernel NVMe-oF TCP target.

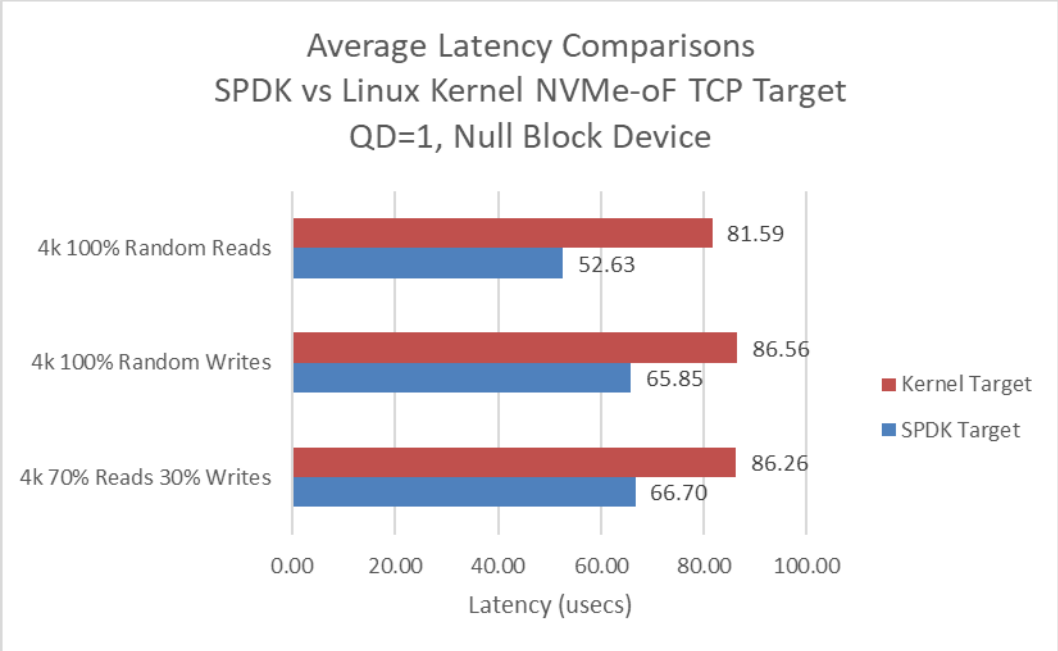

Figure 11: SPDK vs. Kernel NVMe-oF TCP Average I/O Latency for various workloads run using the Kernel Initiator

SPDK NVMe-oF TCP Target Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	52.63	18878	70.5
4K 100% Random Writes IOPS	65.85	14877	108.7
4K 100% Random 70% Reads 30% Writes IOPS	66.70	14672	126.2

Linux Kernel NVMe-oF TCP Target Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	81.59	12074	97.5
4K 100% Random Writes IOPS	85.56	11369	116.2
4K 100% Random 70% Reads 30% Writes IOPS	86.26	11429	105.2

SPDK vs Linux Kernel NVMe-oF TCP Initiator Latency Results

This following data was collected using Kernel & SPDK initiator against an SPDK target.

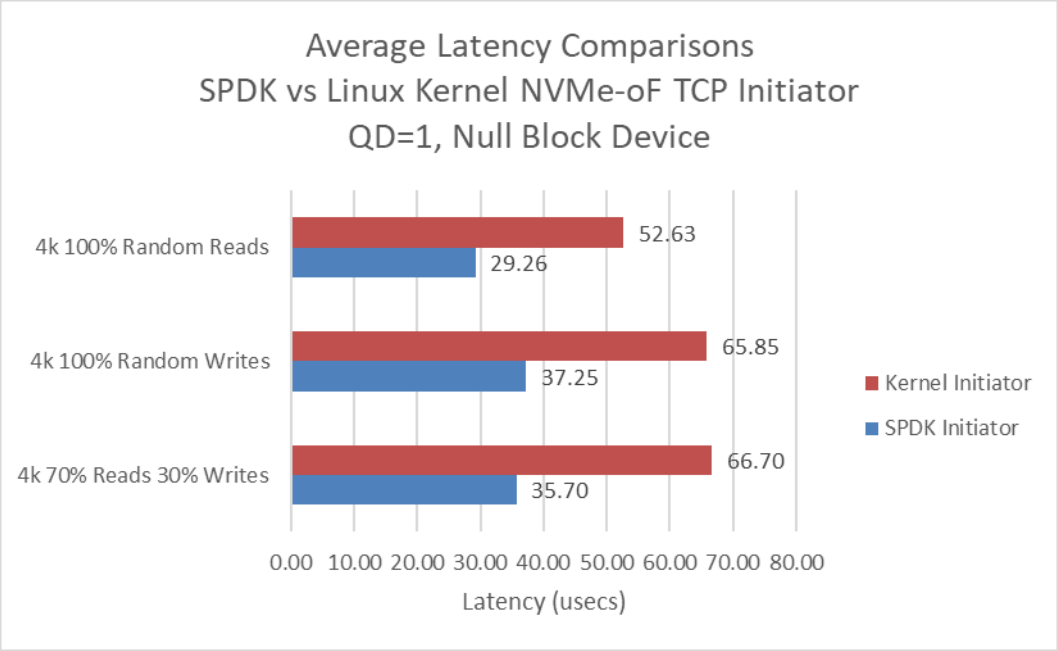

Figure 12: SPDK vs. Kernel NVMe-oF TCP Average I/O Latency for various workloads against SPDK Target

Linux Kernel NVMe-oF TCP Initiator Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	52.63	18878	70.5
4K 100% Random Writes IOPS	65.85	14877	119.3
4K 100% Random 70% Reads 30% Writes IOPS	66.70	14672	126.2

SPDK NVMe-oF TCP Initiator Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	29.26	33900	45.5
4K 100% Random Writes IOPS	37.25	26642	71.9
4K 100% Random 70% Reads 30% Writes IOPS	35.70	27873	71.8

SPDK vs Kernel NVMe-oF TCP Latency Results

Following data was collected using SPDK Target with SPDK Initiator and Linux Target with Linux Initiator.

SPDK NVMe-oF TCP Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	35.70	27873	71.82
4K 100% Random Writes IOPS	37.25	26642	71.85
4K 100% Random 70% Reads 30% Writes IOPS	29.26	33900	45.48

Linux Kernel NVMe-oF TCP Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	86.26	11429	105.20
4K 100% Random Writes IOPS	86.56	11369	116.22
4K 100% Random 70% Reads 30% Writes IOPS	81.59	12074	97.45

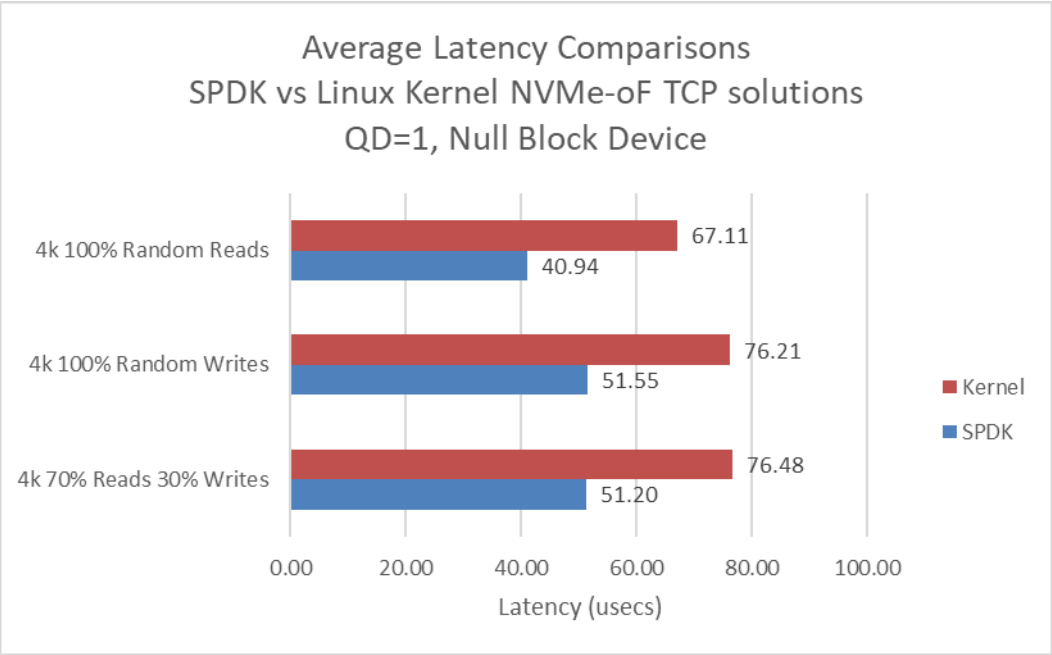

Figure 13: SPDK vs. Kernel NVMe-oF TCP solutions Average I/O Latency for various workloads

Conclusions

1. The SPDK NVMe-oF Target reduces the NVMe-oF (TCP) average round trip I/O latency (reads/writes) by up to 29 usec vs. the Linux Kernel NVMe-oF target for the TCP transport. This is entirely software overhead.

2. SPDK NVMe-oF Initiator reduces the average latency by up to 31 usec vs. the Linux Kernel NVMe-oF Initiator for the TCP transport, which eliminates up to 50% NVMe-oF software overhead.

3. The SPDK NVMe-oF TCP target and initiator reduced the average latency by up to 60% vs. the Linux Kernel NVMe-oF target and initiator for the TCP transport.

4. The SPDK NVMe-oF Initiator reduces the p99 latency by 35% and 60% for the 4K random reads and write workloads respectively.

Test Case 4: NVMe-oF Performance with increasing # of connections

This test case was performed in order to understand throughput and latency capabilities of SPDK vs. Linux Kernel NVMe-oF TCP Target under increasing number of connections per subsystem. The number of connections (or I/O queue pairs) per NVMe-oF subsystem were varied and corresponding aggregated IOPS and number of CPU cores metrics were reported. The number of CPU cores metric was calculated from %CPU utilization measured using the sar utility from the Linux systat package. The SPDK NVMe-oF TCP Target was configured to run on 30 CPU cores, exporting 16 NVMe-oF subsystems (1 per Intel P4600). Two initiators were used, both running I/Os to 8 separate NVMe-oF subsystems using the Linux Kernel NVMe-oF TCP initiator. We ran the following workloads on the host systems:

- 4KB 100% Random Read
- 4KB 100% Random Write
- 4KB Random 70% Read 30% Write

Item	Description
Test Case	NVMe-oF Target performance under varying # of connections
SPDK NVMe-oF Target configuration	Same as in Test Case #1, using 30 CPU cores.
Kernel NVMe-oF Target configuration	Target configuration file loaded using nvmet-cli tool. For detail configuration file contents please see Appendix A.
Kernel NVMe-oF Initiator #1	**Device config** Performed using nvme-cli tool. modprobe nvme-fabrics nvme connect –n nqn.2018-09.io.spdk:cnode1 –t tcp –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode2 –t tcp –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode3 –t tcp –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode4 –t tcp –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode5 –t tcp –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode6 –t tcp –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode7 –t tcp –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode8 –t tcp –a 20.0.1.1 –s 4420
Kernel NVMe-oF Initiator #2	**Device config** Performed using nvme-cli tool. modprobe nvme-fabrics nvme connect –n nqn.2018-09.io.spdk:cnode9 –t tcp –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode10 –t tcp –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode11 –t tcp –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode12 –t tcp –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode13 –t tcp –a 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode14 –t tcp –a 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode15 –t tcp –a 10.0.1.1 –s 4420

	nvme connect –n nqn.2018-09.io.spdk:cnode16 –t tcp –a 10.0.1.1 –s 4420
FIO configuration (used on both initiators)	**FIO.conf** [global] ioengine=libaio thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={8, 16, 32, 64, 128} time_based=1 ramp_time=60 runtime=300 numjobs={1, 4, 16, 32} [filename1] filename=/dev/nvme0n1 [filename2] filename=/dev/nvme1n1 [filename3] filename=/dev/nvme2n1 [filename4] filename=/dev/nvme3n1 [filename5] filename=/dev/nvme4n1 [filename6] filename=/dev/nvme5n1 [filename7] filename=/dev/nvme6n1 [filename8] filename=/dev/nvme7n1

The number of CPU cores used while running the SPDK NVMe-oF TCP target was 30, whereas for the Linux Kernel NVMe-oF TCP target there was no CPU core limitation applied.

The numbers in the graph represent relative performance in IOPS/core which was calculated based on total aggregate IOPS divided by total CPU cores used while running that specific workload. For the Kernel NVMe-oF target, total CPU cores was calculated from % CPU utilization which was measured using the sar utility in Linux.

4k Random Read Results

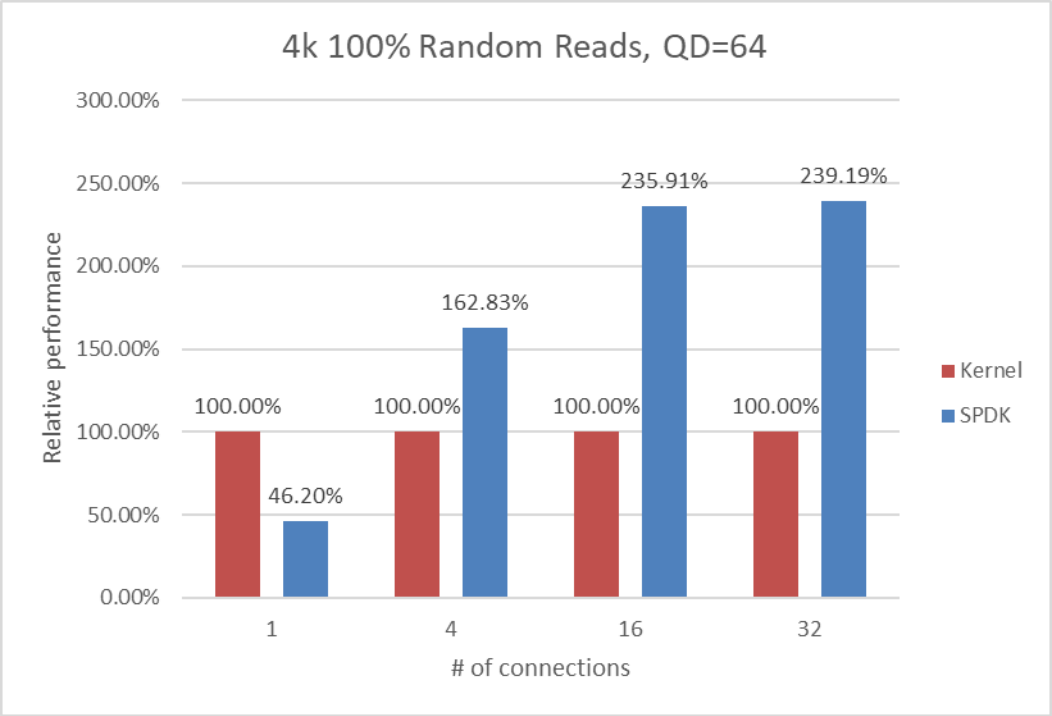

Figure 14: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF TCP Target for 4K 100% Random Reads using the Kernel Initiator

Linux Kernel NVMe-oF TCP Target: 4K 100% Random Reads, QD=64

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	4058.69	1039.0	985.3	13.1
4	14250.22	3348.0	1122.6	47.0
16	16711.26	4278.0	3829.2	69.6
32	16547.56	4236.0	7734.6	72.5

SPDK NVMe-oF TCP Target: 4K 100% Random Reads, QD=64

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	4289.66	1098.1	932.7	30.0
4	14813.56	3792.3	1079.9	30.0
16	17002.24	4352.5	3763.7	30.0
32	16385.30	4194.5	7812.1	30.0

4k Random Write Results

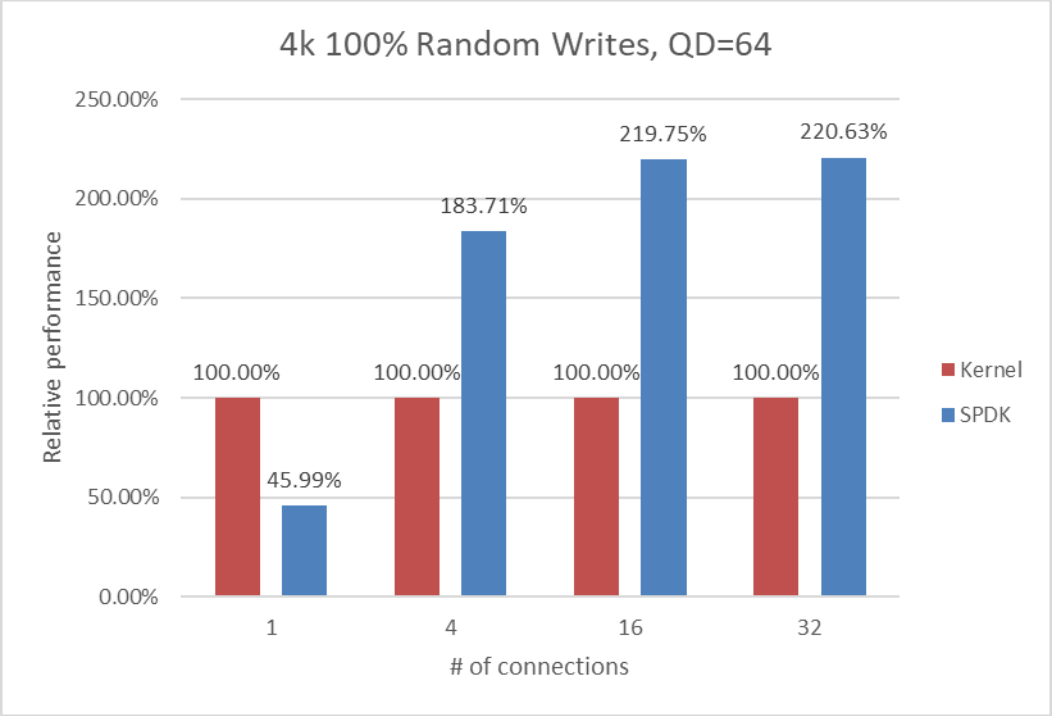

Figure 15: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF TCP Target for 4K 100% Random Writes

Note: Drives were not pre-conditioned while running 100% Random write I/O Test

Linux Kernel NVMe-oF TCP Target: 4K 100% Random Writes, QD=64

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	4326.62	1107.6	924.2	13.6
4	13288.72	3401.9	1203.4	56.4
16	13302.78	3405.4	4812.8	68.8
32	12773.11	3269.8	10022.0	70.0

SPDK NVMe-oF TCP Target: 4K 100% Random Writes, QD=64

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	4378.88	1121.0	914.4	30.0
4	12993.09	3326.2	1230.9	30.0
16	12756.08	3265.5	5019.9	30.0
32	12079.04	3092.1	10602.0	30.0

4k Random Read-Write Results

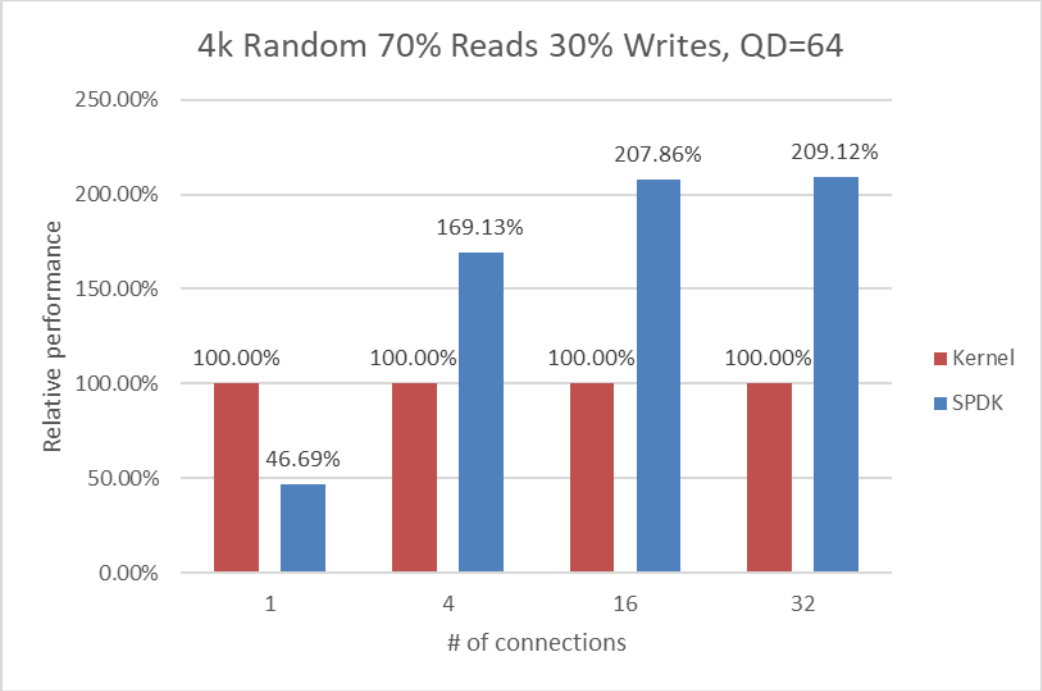

Figure 16: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF TCP Target for 4K Random 70% Reads 30% Writes

Linux Kernel NVMe-oF TCP Target: 4K 70% Random Read 30% Random Write, QD=64

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	3988.85	1021.1	877.1	13.6
4	13353.91	3418.6	926.7	50.9
16	14491.60	3709.8	3731.4	68.2
32	14079.35	3604.2	7970.7	71.6

SPDK NVMe-oF TCP Target: 4K 70% Random Read 30% Random Write, QD=64

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	4096.96	1048.8	818.0	30.0
4	13313.90	3408.3	926.7	30.0
16	13241.83	3389.8	2279.5	30.0
32	12329.43	3156.2	3724.2	30.0

Low Connections Results

During testing it was observed that relative performance of SPDK Target is about 45% of Kernel Target performance at low number of connections (1 connection per subsystem) because SPDK uses a fixed number of CPU cores and does not have a mechanism to decrease the number of cores on the fly if workload does not use all the CPU resources.

The test cases with 1 connection per subsystems were re-run with SPDK using only 4 CPU cores.

SPDK and Linux Kernel NVMe-oF TCP Target relative performance comparison for various workloads, QD=64, 1 connection per subsystem.

Workload	Target	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
Random Read	Linux	4058.69	1039.0	985.3	13.1
	SPDK	3663.92	938.0	1093.8	4.0
Random Write	Linux	4326.62	1107.6	924.2	13.6
	SPDK	2982.57	763.5	1341.8	4.0
Random Read/Write	Linux	3988.85	1021.1	1934.0	13.6
	SPDK	3307.05	846.6	2331.9	4.0

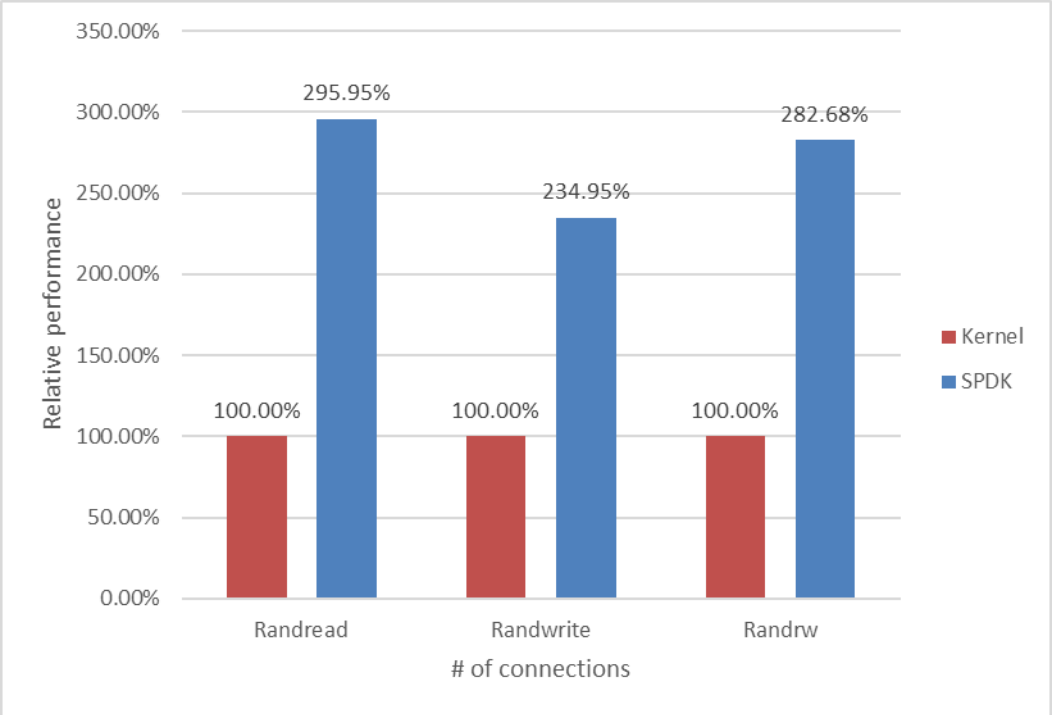

Figure 17: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF TCP Target for various workloads, 16 NVMe-oF Subsystems, 1 connection per subsystem, SPDK Target configured with 4CPU Cores

Conclusions

1. The performance of SPDK NVMe-oF TCP Target for Random Write and Random Read-Write workloads peaked when the number of connections per subsystem was 4. For Random-Read peak performance was observer at higher number of connections per subsystem which was 116. These values seem to be the optimum configurations and allow for best performance in setup used for this testing.

2. The relative performance for the Linux Kernel NVMe-oF TCP Target was better than SPDK in when there was just one connection per subsystem because SPDK uses a fixed number of CPU cores and has no mechanism to dynamically decrease the number of cores based on the workload.

   For this reason, we've re-run the test cases where number of connections per subsystem was 1 with a lower number of SPDK Target CPU cores(4 CPU cores) and added the results to the tables.

3. SPDK NVMe-oF TCP target relative performance was between 1.6 - 2.4 times better than the Linux Kernel NVMe-oF TCP target in all other test cases where each subsystem had multiple connections.

Summary

This report showcased performance results with SPDK NVMe-oF TCP target and initiator under various test cases, including:

- I/O core scaling
- Average I/O latency
- Performance with increasing number of connections per subsystems

It compared performance results while running Linux Kernel NVMe-oF (Target/Initiator) against the accelerated polled-mode driven SPDK NVMe-oF (Target/Initiator) implementation.

Throughput scales up and latency decreases almost linearly with the scaling of SPDK NVMe-oF target cores when serving 4K random workloads until the network traffic reaches around 100 Gbps at about 16 CPU cores. Beyond that the trend becomes non-linear. At 40 CPU cores for the target we did not reach 200G network saturation.

For the SDPK NVMe-oF TCP Initiator, the IOPS throughput scales almost linearly with addition of CPU cores until the network was almost saturated, however, as we got closer to network saturation it was observed that the throughput scaling becomes non-linear. A single initiator was able to almost saturate 100Gb link.

For the NVMe-oF TCP latency comparison, the SPDK NVMe-oF Target and Initiator average latency is up to 60% lower than their Kernel counterparts when testing against null bdev based backend.

The SPDK NVMe-oF TCP Target performed up to 2.4 times better w.r.t IOPS/core than Linux Kernel NVMe-oF target while running 4K 100% random read workload with increasing number of connections per NVMe-oF subsystem.

This report provides information regarding methodologies and practices while benchmarking NVMe-oF using SPDK, as well as the Linux Kernel. It should be noted that the performance data showcased in this report is based on specific hardware and software configurations and that performance results may vary depending on the hardware and software configurations.

Appendix A

Example Kernel NVMe-oF TCP Target configuration for Test Case 4.


```
{
  "ports": [
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.0.1",
        "trsvcid": "4420",
        "trtype": "tcp"
      },
      "portid": 1,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode1"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.0.1",
        "trsvcid": "4421",
        "trtype": "tcp"
      },
      "portid": 2,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode2"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.0.1",
        "trsvcid": "4422",
        "trtype": "tcp"
      },
      "portid": 3,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode3"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.0.1",
        "trsvcid": "4423",
        "trtype": "tcp"
      },
      "portid": 4,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode4"
```

```
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.1.1",
        "trsvcid": "4424",
        "trtype": "tcp"
      },
      "portid": 5,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode5"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.1.1",
        "trsvcid": "4425",
        "trtype": "tcp"
      },
      "portid": 6,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode6"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.1.1",
        "trsvcid": "4426",
        "trtype": "tcp"
      },
      "portid": 7,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode7"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.1.1",
        "trsvcid": "4427",
        "trtype": "tcp"
      },
      "portid": 8,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode8"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
```

```
        "traddr": "10.0.0.1",
        "trsvcid": "4428",
        "trtype": "tcp"
      },
      "portid": 9,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode9"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.0.1",
        "trsvcid": "4429",
        "trtype": "tcp"
      },
      "portid": 10,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode10"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.0.1",
        "trsvcid": "4430",
        "trtype": "tcp"
      },
      "portid": 11,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode11"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.0.1",
        "trsvcid": "4431",
        "trtype": "tcp"
      },
      "portid": 12,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode12"
      ]
    },
{
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4432",
        "trtype": "tcp"
      },
      "portid": 13,
```

```
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode13"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4433",
        "trtype": "tcp"
      },
      "portid": 14,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode14"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4434",
        "trtype": "tcp"
      },
      "portid": 15,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode15"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4435",
        "trtype": "tcp"
      },
      "portid": 16,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode16"
      ]
    }
  ],
"hosts": [],
 "subsystems": [
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme0n1",
```

```
          "uuid": "b53be81d-6f5c-4768-b3bd-203614d8cf20"
        },
        "enable": 1,
        "nsid": 1
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode1"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme1n1",
          "uuid": "12fcf584-9c45-4b6b-abc9-63a763455cf7"
        },
        "enable": 1,
        "nsid": 2
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode2"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme2n1",
          "uuid": "ceae8569-69e9-4831-8661-90725bdf768d"
        },
        "enable": 1,
        "nsid": 3
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode3"
  },
 {
   "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme3n1",
          "uuid": "39f36db4-2cd5-4f69-b37d-1192111d52a6"
        },
        "enable": 1,
```

```
          "nsid": 4
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode4"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme4n1",
            "uuid": "984aed55-90ed-4517-ae36-d3afb92dd41f"
          },
          "enable": 1,
          "nsid": 5
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode5"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme5n1",
            "uuid": "d6d16e74-378d-40ad-83e7-b8d8af3d06a6"
          },
          "enable": 1,
          "nsid": 6
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode6"
    },
   {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme6n1",
            "uuid": "a65dc00e-d35c-4647-9db6-c2a8d90db5e8"
          },
          "enable": 1,
          "nsid": 7
        }
      ],
```

```
    "nqn": "nqn.2018-09.io.spdk:cnode7"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme7n1",
          "uuid": "1b242cb7-8e47-4079-a233-83e2cd47c86c"
        },
        "enable": 1,
        "nsid": 8
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode8"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme8n1",
          "uuid": "f12bb0c9-a2c6-4eef-a94f-5c4887bbf77f"
        },
        "enable": 1,
        "nsid": 9
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode9"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme9n1",
          "uuid": "40fae536-227b-47d2-bd74-8ab76ec7603b"
        },
        "enable": 1,
        "nsid": 10
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode10"
  },
  {
```

```
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme10n1",
            "uuid": "b9756b86-263a-41cf-a68c-5cfb23c7a6eb"
          },
          "enable": 1,
          "nsid": 11
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode11"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme11n1",
            "uuid": "9d7e74cc-97f3-40fb-8e90-f2d02b5fff4c"
          },
          "enable": 1,
          "nsid": 12
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode12"
    },
{
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme12n1",
            "uuid": "d3f4017b-4f7d-454d-94a9-ea75ffc7436d"
          },
          "enable": 1,
          "nsid": 13
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode13"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
```

```
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme13n1",
            "uuid": "6b9a65a3-6557-4713-8bad-57d9c5cb17a9"
          },
          "enable": 1,
          "nsid": 14
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode14"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme14n1",
            "uuid": "ed69ba4d-8727-43bd-894a-7b08ade4f1b1"
          },
          "enable": 1,
          "nsid": 15
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode15"
    },
  {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme15n1",
            "uuid": "5b8e9af4-0ab4-47fb-968f-b13e4b607f4e"
          },
          "enable": 1,
          "nsid": 16
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode16"
    }
  ]
}
```

DISCLAIMERS

§