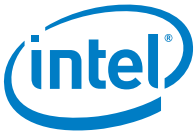


SPDK NVMe-oF RDMA (Target & Initiator) Performance Report Release 19.10

Testing Date: December 2019

Performed by: Karol Latecki (karol.latecki@intel.com)

Maciej Wawryk (maciejx.wawryk@intel.com)

Acknowledgments:

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

James Harris (james.r.harris@intel.com)

Contents

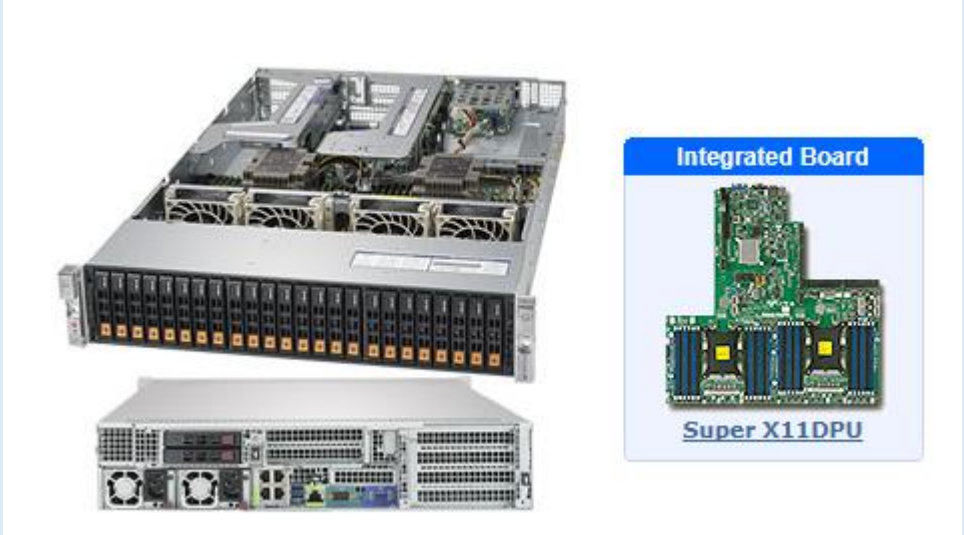
Audience and Purpose

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 contains performance and efficiency information between SPDK NVMe-oF (Target & Initiator) vs. Linux Kernel NVMe-oF (Target & Initiator) on a set of underlying block devices under various test cases. This report covers the RDMA 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:
Operating System	Fedora 29
BIOS	3.1a
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: QDV10190) (8 on each CPU socket)
NIC	2x 100GbE Mellanox ConnectX-5 NICs. Both ports connected. 1 NIC per CPU socket.

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

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
Storage	**OS:** 1x 240GB INTEL SSDSC2BB240G6
NIC	1x 100GbE Mellanox ConnectX-4 NIC. Both ports connected to Target server. (connected to CPU socket 0)

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
Storage	**OS:** 1x 240GB INTEL SSDSC2BB240G6
NIC	1x 100GbE Mellanox ConnectX-4 NIC. Both ports connected to Target server. (connected to CPU socket 0)

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

Kernel & BIOS spectre-meltdown information

All three server systems use Fedora 5.2.7-100.fc29 kernel version available from DNF repository with default patches for spectre-meltdown issue enabled.

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

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 as close to the CPU cache as possible.

The SPDK NVMe-oF target and initiator uses the Infiniband/RDMA verbs API to access an RDMA-capable NIC. These should work on all flavors of RDMA transports, but are currently tested against RoCEv2, iWARP, and Omni-Path NICs. Similar to the SPDK NVMe driver, 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 individual 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 P4600 SSDs which were used as block devices for NVMe-oF subsystems and two 100GbE Mellanox ConnectX-5 NICs connected to 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).

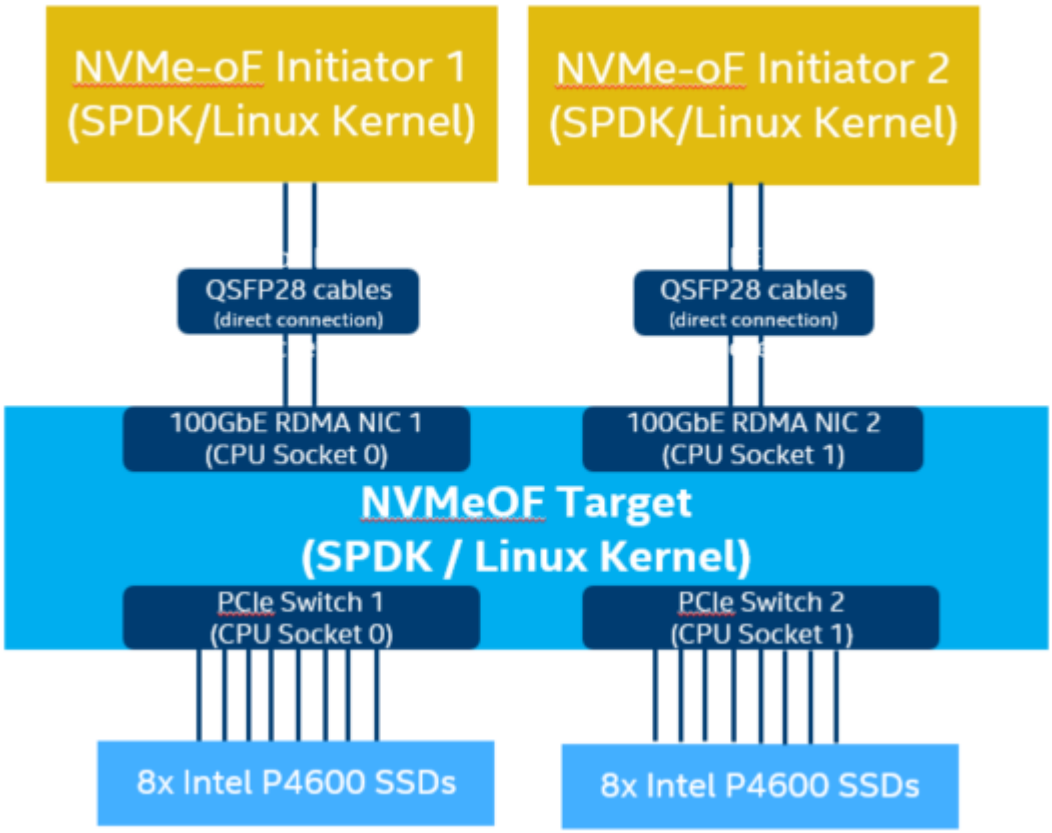


Figure 1: High level NVMe-oF performance testing setup

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

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

SPDK NVMe-oF RDMA target was configured to run with 16 NVMe-oF subsystems. Each NVMe-oF subsystem ran on top of an individual bdev backed by a single Intel P4600 device. Each of the 2 initiators were connected to 8 individual NVMe-oF subsystems which were exposed via SPDK NVMe-oF Target over 1x 100GbE NIC. SPDK bdev FIO plugin was used to target 8 individual NVMe-oF bdevs on each of the initiators. SPDK Target Reactor Mask was configured to use 1, 2, 3, 4, 5 and 6 cores tests while running following workloads on each initiator:

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

For detailed configuration please refer to table below. Actual configuration of SPDK NVMe-oF Target in test was done using JSON-RPC and the table contains a sequence of commands used by spdk/scripts/rpc.py script rather than a configuration file. 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 of below commands are 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 RDMA
(creates RDMA transport layer with default values:
trtype: "RDMA"
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)

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 rdma -f ipv4 -s 4420 -a 20.0.0.1
nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode2 -t rdma -f ipv4 -s 4420 -a 20.0.0.1
nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode3 -t rdma -f ipv4 -s 4420 -a 20.0.0.1
nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode4 -t rdma -f ipv4 -s 4420 -a 20.0.0.1
nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode5 -t rdma -f ipv4 -s 4420 -a 20.0.1.1
nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode6 -t rdma -f ipv4 -s 4420 -a 20.0.1.1
```

	nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode7 -t rdma -f ipv4 -s 4420 -a 20.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode8 -t rdma -f ipv4 -s 4420 -a 20.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode9 -t rdma -f ipv4 -s 4420 -a 10.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode10 -t rdma -f ipv4 -s 4420 -a 10.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode11 -t rdma -f ipv4 -s 4420 -a 10.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode12 -t rdma -f ipv4 -s 4420 -a 10.0.0.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode13 -t rdma -f ipv4 -s 4420 -a 10.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode14 -t rdma -f ipv4 -s 4420 -a 10.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode15 -t rdma -f ipv4 -s 4420 -a 10.0.1.1 nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode16 -t rdma -f ipv4 -s 4420 -a 10.0.1.1
SPDK NVMe-oF Initiator - FIO plugin configuration	**BDEV.conf** [Nvme] TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode1" Nvme0 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode2" Nvme1 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode3" Nvme2 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode4" Nvme3 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode5" Nvme4 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode6" Nvme5 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode7" Nvme6 TransportId "trtype:RDMA 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} time_based=1 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	4438.27	1136.2	900.8
2 cores	10441.66	2673.1	383.3
3 cores	16188.98	4144.4	246.8
4 cores	20086.32	5142.1	199.3
5 cores	20870.73	5342.9	191.4
6 cores	22195.29	5682.0	180.2

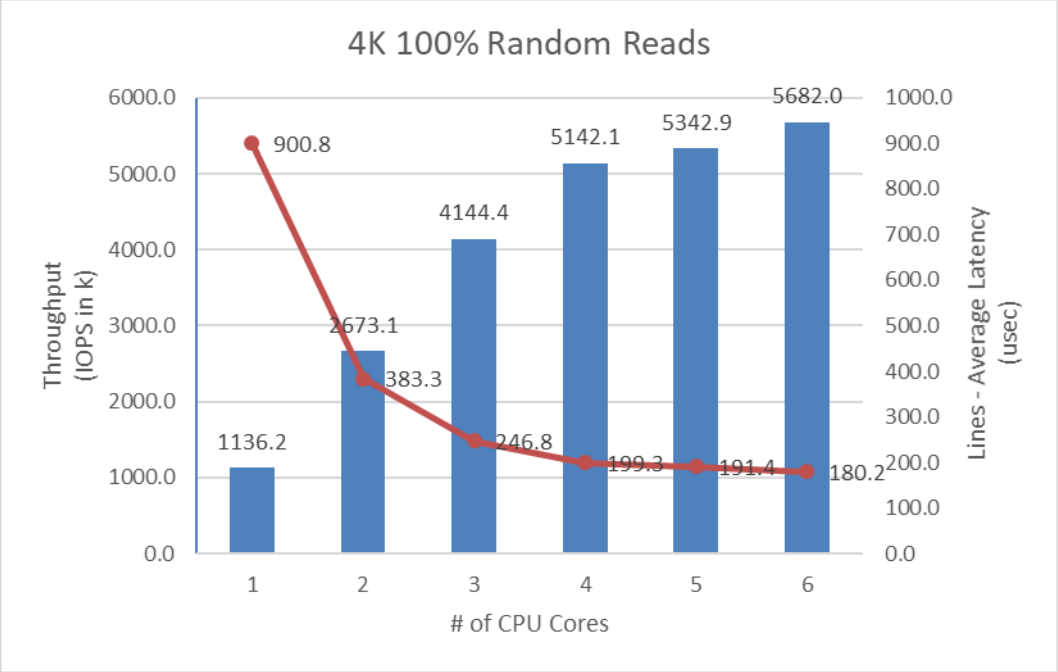


Figure 2: SPDK NVMe-oF RDMA Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Read workload at QD=64

4k Random Write results

Test Result: 4K 100% Random Writes IOPS QD=32

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5999.95	1536.0	664.0
2 cores	13349.26	3417.4	296.5
3 cores	18824.98	4819.2	211.1
4 cores	21441.12	5488.9	184.6
5 cores	21265.96	5444.1	187.1
6 cores	21900.01	5606.4	181.6

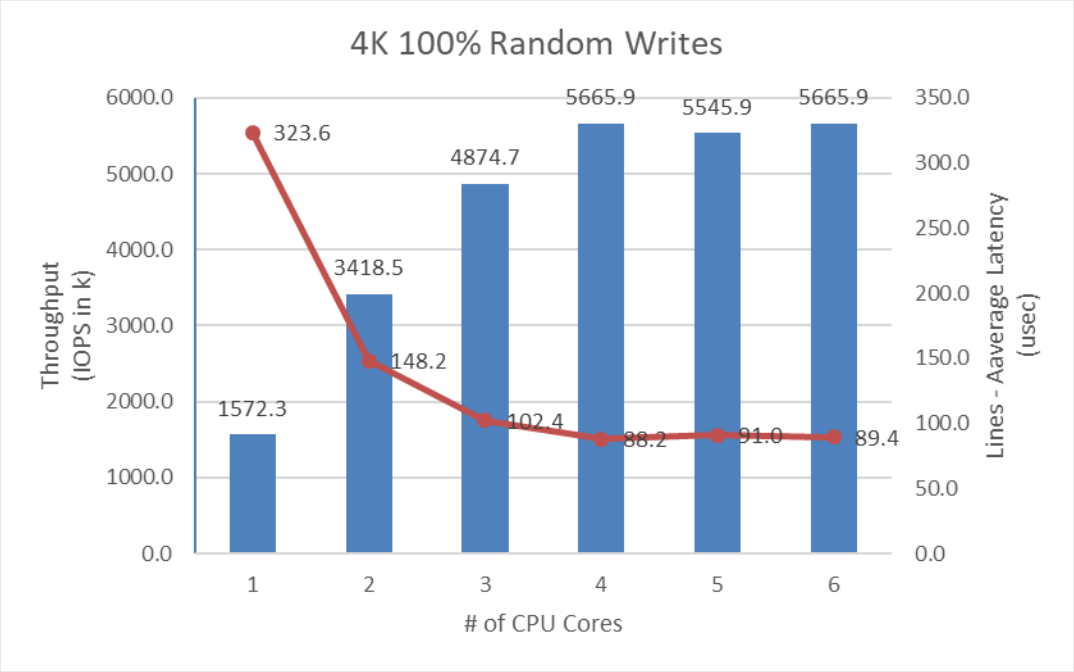


Figure 3: SPDK NVMe-oF RDMA Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write workload at QD=64

4k Random Read-Write results

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

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4494.45	1150.6	889.2
2 cores	10319.46	2641.8	386.4
3 cores	13276.88	3398.9	300.4
4 cores	13879.24	3553.1	287.7
5 cores	14066.23	3601.0	283.9
6 cores	14164.25	3626.0	282.0

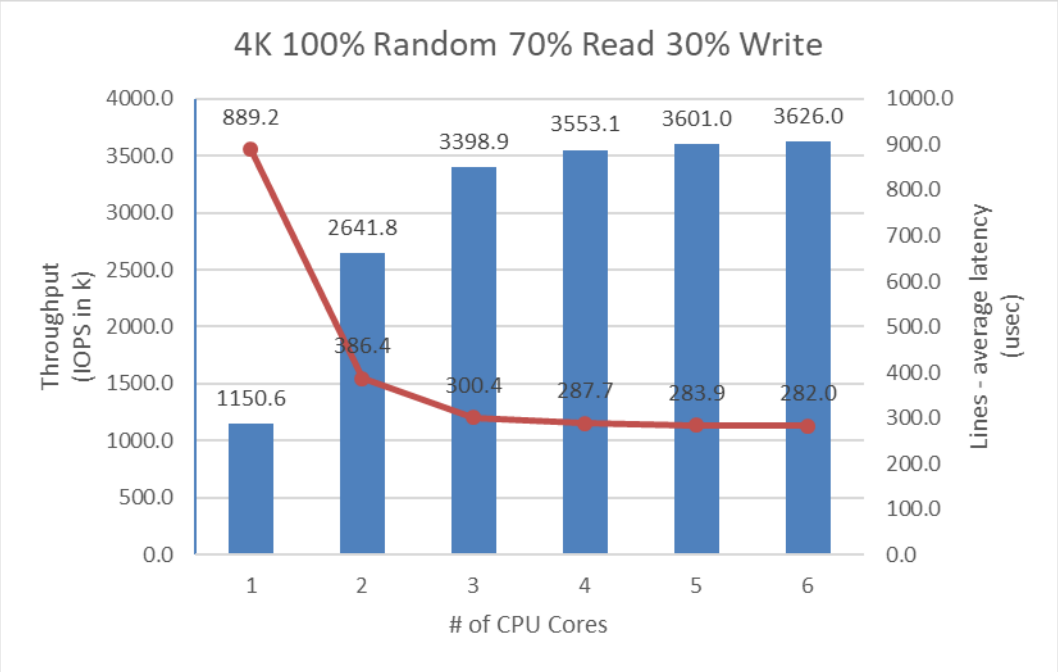


Figure 4: SPDK NVMe-oF RDMA Target I/O core scaling: IOPS vs. Latency while running 4KB Random 70% read 30% write workload at QD=64

Conclusions

1. For 100% Random Reads, throughput scales up and latency decreases almost linearly with the scaling of SPDK NVMe-oF Target I/O cores until transitioning from 4 to 5 CPU cores, when benefit of adding another core is almost halved. Increasing from 5 to 6 CPU cores does not show any benefit as we have reached network saturation.

2. For 100% Random Write workload, throughput scales up and latency decreases linearly with the scaling of SPDK NVMe-oF Target I/O cores until the network is saturated at just 4 CPU cores. The SSDs were not preconditioned prior to executing the test to ensure the 4K random write workload saturated the newtwork; preconditioning the SSDs limits the workload performance to the SSDs steady state performance.

3. For 4K Random 70/30 Reads/Writes, performance doesn't scale beyond 3 cores due to some other bottleneck. It was observed that running this test case locally without going over the network the storage can achieve 4M IOPS. This is close to the maximum performance of 3.6M-3.8M IOPS that was observed over NVMe-oF for the RDMA transport.

Large Sequential I/O Performance

128K block size I/O tests were performed with sequential I/O workloads at queue depth 8. The rest of the FIO configuration is similar to 4K test case in the previous part of this document. We used iodepth=8 because higher queue depth resulted in negligible bandwidth gain and a significant increase in the latency.

Test Result: 128K 100% Sequential Reads QD=8

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	19708.43	157.7	819.3
2 cores	19845.44	158.8	813.9
3 cores	19964.63	159.7	809.9
4 cores	20058.59	160.5	807.0

Test Result: 128K 100% Sequential Writes QD=8

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	17703.95	141.6	903.6
2 cores	17823.34	142.6	897.6
3 cores	17992.11	143.9	889.6
4 cores	17853.47	142.8	896.0

Test Result: 128K 70% Reads 30% Writes QD=8

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	17689.01	141.5	1584.7
2 cores	17845.20	142.8	1562.3
3 cores	17950.74	143.6	1551.4
4 cores	18168.04	145.3	1523.5

Conclusions

1. We were not able to saturate the network with any workload used in this test case. This suggests some performance degradation in comparison to SPDK 19.04 where network saturation with sequential workloads was possible with low queue depths and just a single core on the Target side.

2. Using a single CPU core on the NVMe-oF Target we achieved approximately ¾ of our network throughput (about 150Gbps).

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

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

The SPDK NVMe-oF RDMA Target was configured using 6 cores; all the other configurations are similar to test case 1. The SPDK bdev FIO plugin was used to target 8 NVMe-oF bdevs on each of the 2 initiators. Following workloads from both of the initiators were run using fio client-server mode.

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

Depending on the number of initiators and initiator cores, we varied the number of NVMe-oF subsystems exported by the target, instead of exposing all 16 NVMe-oF subsystems. This was done to avoid a situation where single initiator would connect to all 16 target subsystems, which resulted in unnatural latency spike in the 1 CPU core test case.

1 core: 1 initiator running on a single core connecting to 4 subsystems.

2 cores: 2 initiators, each running on a single core. Each initiator connected to 4 subsystems.

3 cores: 2 initiators, the first one running on 1 core and other one running on 2 cores. Initiator 1 connected to 4 subsystems and initiator 2 connected to 8 subsystems.

4 cores: 2 initiators, both running on 2 cores each. Both initiators connected to 8 subsystems.

Item	Description
Test Case	Test SPDK NVMe-oF InitiatorI/O core scaling
SPDK NVMe-oF Target configuration	Same as in Test Case #1, using 6 CPU cores.
SPDK NVMe-oF Initiator 1 - FIO plugin configuration	**BDEV.conf** [Nvme] TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode1" Nvme0 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode2" Nvme1 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode3" Nvme2 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode4" Nvme3 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode5" Nvme4 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode6" Nvme5 TransportId "trtype:RDMA adrfam:IPv4 traddr:20.0.1.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode7" Nvme6 TransportId "trtype:RDMA 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={32, 64, 128, 256} time_based=1 ramp_time=60 runtime=300 {filename_section} **FIO.conf filename section for 1 & 2 CPU test run** [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 **FIO.conf filename section for 3 & 4 CPU test run** [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 [filename1] filename=Nvme4n1 filename=Nvme5n1 filename=Nvme6n1 filename=Nvme7n1
SPDK NVMe-oF Initiator 2 - FIO plugin configuration	**BDEV.conf** [Nvme] TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode9" Nvme0 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode10" Nvme1 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode11" Nvme2 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.0.1 trsvcid:4420 subnqn:nqn.2018-09.io.spdk:cnode12" Nvme3 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.1.1 trsvcid:4421 subnqn:nqn.2018-09.io.spdk:cnode13" Nvme4 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.1.1 trsvcid:4421 subnqn:nqn.2018-09.io.spdk:cnode14" Nvme5 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.1.1 trsvcid:4421 subnqn:nqn.2018-09.io.spdk:cnode15" Nvme6 TransportId "trtype:RDMA adrfam:IPv4 traddr:10.0.1.1 trsvcid:4421 subnqn:nqn.2018-09.io.spdk:cnode16" Nvme7 **FIO.conf** Similar as Initiator 1. The only differences are were in the filename section which are shown below. **FIO.conf filename section for 1 CPU test run** N/A (only Initiator 1 is used) **FIO.conf filename section for 2 CPU test run** [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1

	filename=Nvme3n1 **FIO.conf filename section for 3 CPU test run** [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 **FIO.conf filename section for 4 CPU test run** [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 [filename1] filename=Nvme4n1 filename=Nvme5n1 filename=Nvme6n1 filename=Nvme7n1

4k Random Read results

Test Result: 4K 100% Random Read, QD=256, SPDK Target 6 CPU cores

# of Initiator CPU Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5125.35	1312.1	150.1
2 cores	10314.84	2640.6	165.8
3 cores	14533.23	3720.5	171.9
4 cores	19810.16	5071.4	180.1

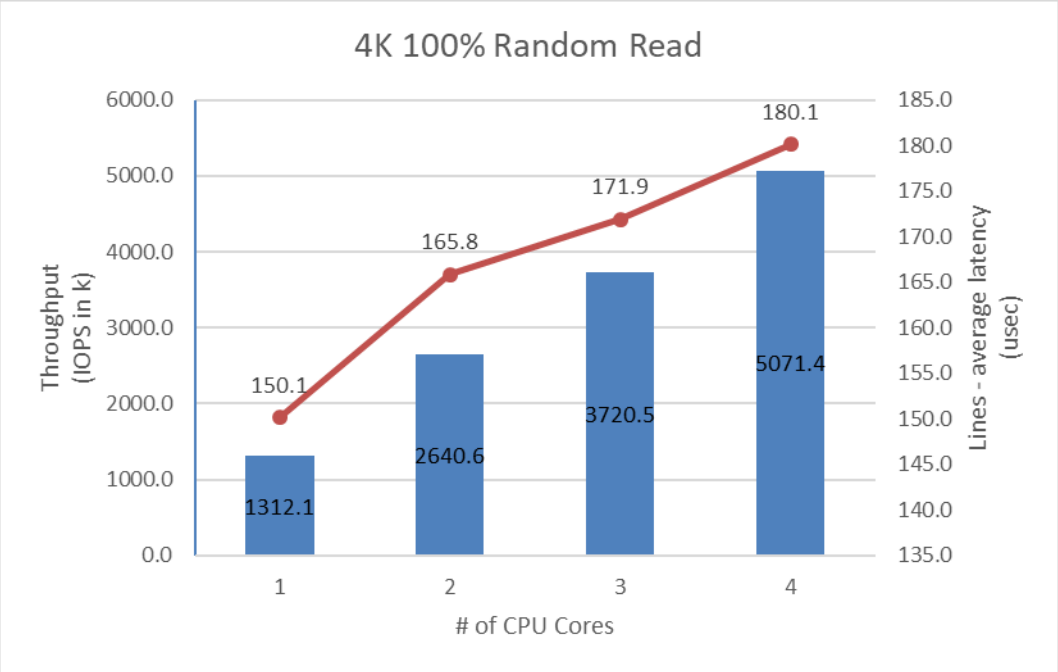


Figure 5: SPDK NVMe-oF RDMA Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Read workload at QD=256

4k Random Write results

Note: The SSDs were pre-conditioned just once with 2x Sequential Write workload before running all the 100% Random Write test cases. This allowed the throughput to scale to the 2x 100GbE network bandwidth when testing with 3 and 4 CPU cores rather than limiting the workload performance to the storage bottleneck (which is approx. 3.2M IOPS).

Test Result: 4K 100% Random Write, QD=256, SPDK Target 6 CPU Cores

# of Initiator CPU Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5002.04	1280.5	112.7
2 cores	9624.68	2463.9	121.0
3 cores	13761.39	3522.9	128.9
4 cores	18239.39	4669.3	133.7

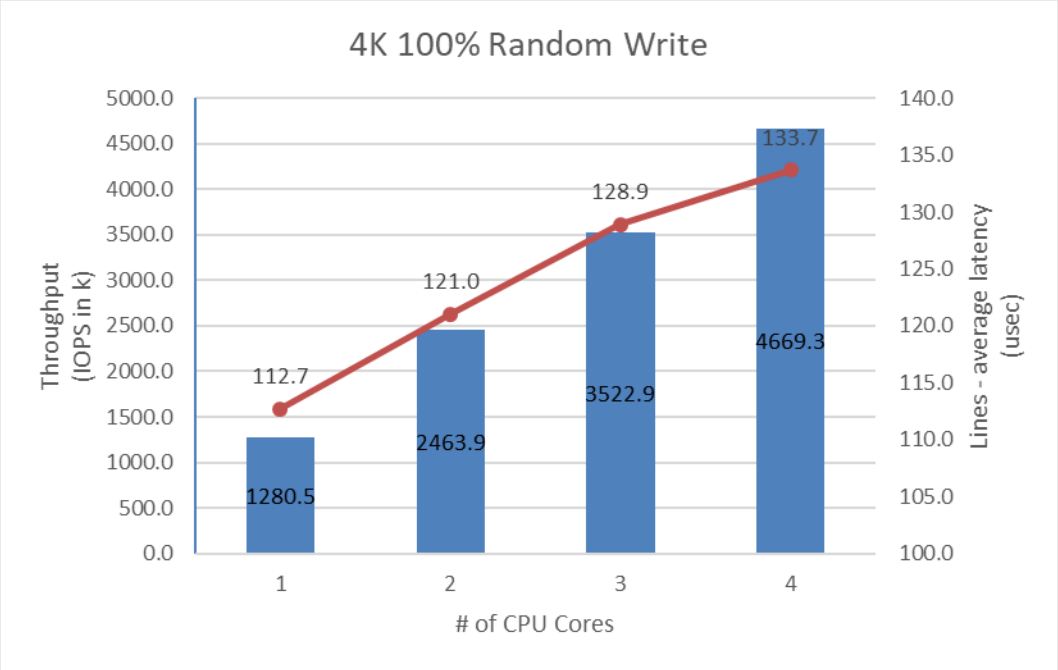


Figure 6: SPDK NVMe-oF RDMA Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write workload at QD=256

4k Random Read-Write results

Test Result: 4K 70% Random Read 30% Random Write QD=256, SPDK Target 6 CPU Cores

# of Initiator CPU Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	3514.00	899.6	104.2
2 cores	7315.66	1872.8	128.2
3 cores	11320.90	2898.1	127.1
4 cores	13946.09	3570.2	138.5

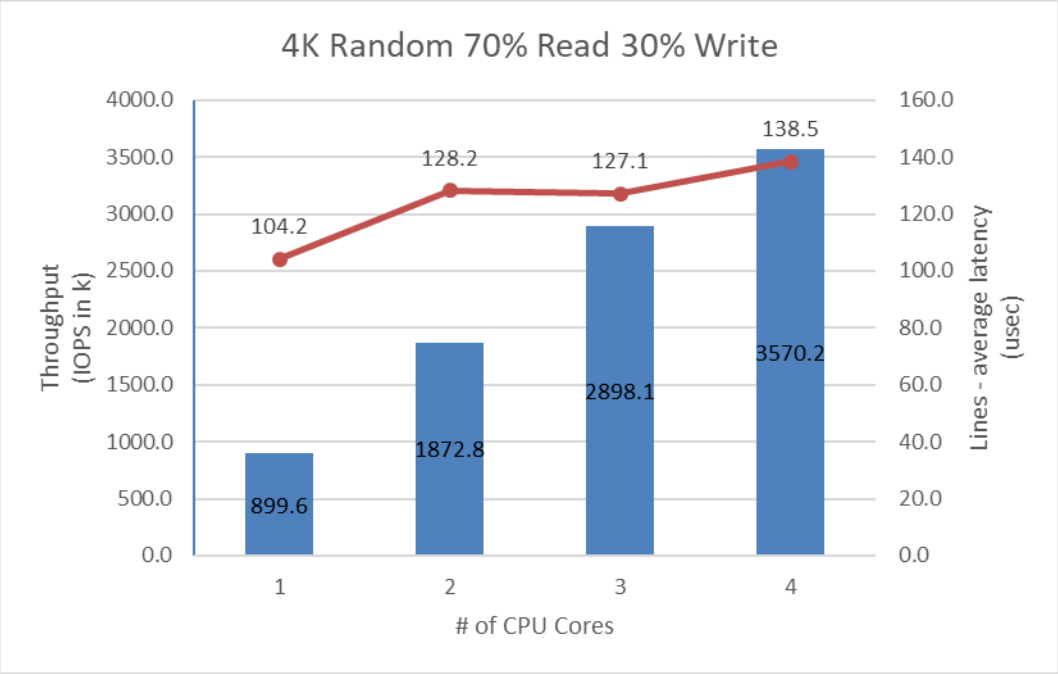
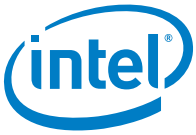


Figure 7: SPDK NVMe-oF RDMA Initiator I/O core scaling: IOPS vs. Latency while running 4KB Random 70% Read 30% Write workload at QD=256



Conclusions

1. The 4K 100% Random Reads and Random Writes workloads IOPS scaled linearly with each CPU core added to the initiator. At 4 CPU cores the workload is close to saturating the network bandwidth.

2. The 4K 70% Random Reads / 30% Random Writes IOPS scales linearly up to 4 CPU cores. The results when using 4 CPUs are similar to these from Test Case 1 which suggest some kind of network or platform bottleneck.

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

This test case was designed to understand latency characteristics of SPDK NVMe-oF RDMA Target and Initiator vs. the Linux Kernel NVMe-oF RDMA 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 containing 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 of below commands are executed with spdk/scripts/rpc.py script. nvmf_create_transport -t RDMA (creates RDMA transport layer with default values: trtype: "RDMA" 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 rdma -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": "rdma"
	},
	"portid": 1,
	"referrals": [],
	"subsystems": [

<table>
<tr><td></td><td><pre>
        "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"
    }
  ]
}
</pre></td></tr>
<tr><td colspan="2">FIO configuration</td></tr>
<tr><td>SPDK NVMe-oF Initiator FIO plugin configuration</td><td>BDEV.conf
[Nvme]
TransportId "trtype:RDMA 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</td></tr>
<tr><td>Kernel initiator configuration</td><td>Device config
Done using nvme-cli tool.
modprobe nvme-fabrics
nvme connect –n nqn.2018-09.io.spdk:cnode1 –t rdma –a 20.0.0.1 –s 4420

FIO.conf</td></tr>
</table>

	[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 ramp_time=60 runtime=300 [filename0] filename=/dev/nvme0n1

SPDK vs Kernel NVMe-OF RDMA Target results

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

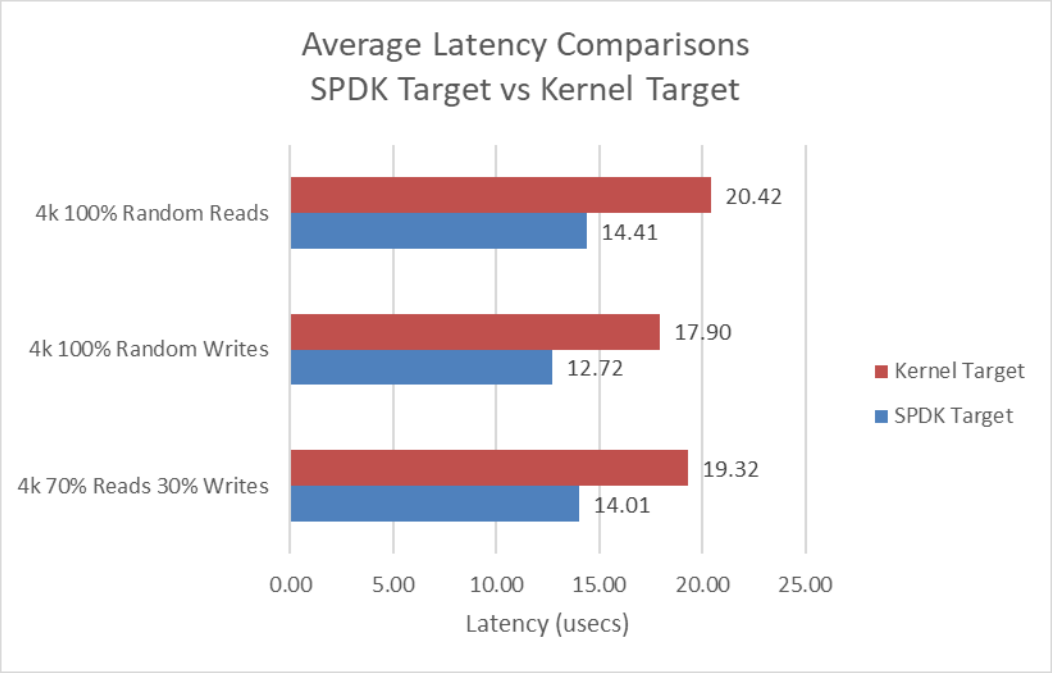


Figure 8: Average I/O Latency comparisons at QD=1 between SPDK and Linux Kernel NVMe-oF RDMA Target for various workloads using the Linux Kernel Initiator

SPDK NVMe-oF RDMA Target QD=1 Latency for a Null block device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Read	14.41	67764	24.7
4K 100% Random Write	12.72	76336	22.6
4K 100% Random 70% Read 30% Write	14.01	69429	24.4

Linux Kernel NVMe-oF RDMA Target QD=1 Latency for a Null block device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads	20.42	48025	29.1
4K 100% Random Writes	17.90	54753	26.9
4K 100% Random 70% Read 30% Write	19.32	50697	28.7

Conclusions

1. For the RDMA transport, the SPDK NVMe-oF Target reduces the NVMe-oF average round trip I/O latency (reads/writes) by up to 6 usec vs. the Linux Kernel NVMe-oF target. This is entirely software overhead, therefore, using the SPDK NVMe-oF target reduces the NVMe-oF software overhead by approximately 30% vs. the Linux Kernel NVMe-oF target

SPDK vs Kernel NVMe-oF RDMA Initiator results

This following data was collected using the Linux Kernel and SPDK NVMe-oF RDMA initiator against an SPDK NVMe-OF RDMA target.

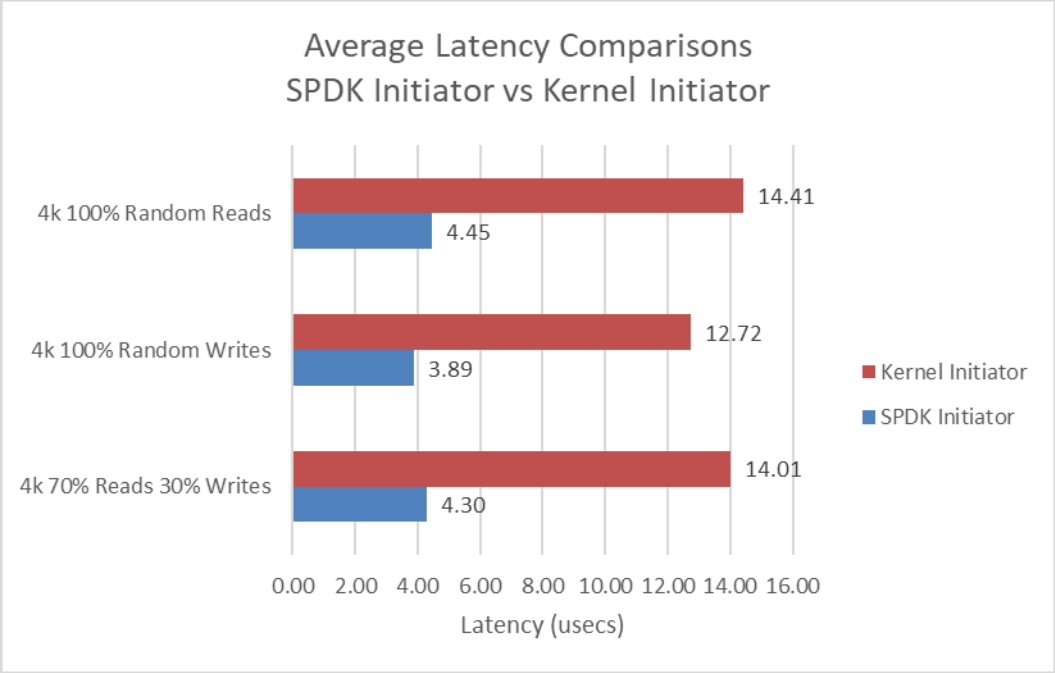


Figure 9: Average I/O Latency Comparisons at QD=1 between SPDK and Linux Kernel NVMe-oF RDMA Initiator for various workloads against SPDK NVMe-OF Target

Linux Kernel NVMe-oF RDMA Initiator QD=1 Latency for a Null block device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	14.41	67764	24.7
4K 100% Random Writes IOPS	12.72	76336	23.4
4K 100% Random 70% Reads 30% Writes IOPS	14.01	69429	24.4

SPDK NVMe-oF RDMA Initiator QD=1 Latency for a Null block device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	4.45	215693	16.2
4K 100% Random Writes IOPS	3.89	245606	14.6
4K 100% Random 70% Reads 30% Writes IOPS	4.30	222143	14.1

Conclusions

1. The SPDK NVMe-oF initiator reduces the NVMe-oF software overhead by up to 3.25x vs. the Linux Kernel NVMe-oF Initiator for the RDMA transport.

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

This test case was performed in order to understand throughput and latency capabilities of SPDK NVMe-oF RDMA Target vs. Linux Kernel NVMe-oF RDMA Target under increasing number of connections per subsystem. 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. Number of CPU cores metric was calculated from %CPU utilization measured using sar (systat package in Linux). SPDK NVMe-oF RDMA Target was configured to run on 4 cores, 16 NVMe-oF subsystems (1 per Intel P4600) and 2 initiators were used both running I/Os to 8 separate subsystems using Kernel NVMe-oF RDMA initiator.

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

Item	Description
Test Case	NVMe-oF RDMA Target performance under varying # of connections
SPDK NVMe-oF Target configuration	Same as in Test Case #1, using 4 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 rdma –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode2 –t rdma –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode3 –t rdma –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode4 –t rdma –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode5 –t rdma –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode6 –t rdma –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode7 –t rdma –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode8 –t rdma –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 rdma –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode10 –t rdma –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode11 –t rdma –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode12 –t rdma –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode13 –t rdma –a 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode14 –t rdma –a 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode15 –t rdma –a 10.0.1.1 –s 4420

	nvme connect –n nqn.2018-09.io.spdk:cnode16 –t rdma –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 SPDK NVMe-oF Target was configured to use 4 CPU cores, whereas we did not limit the number of CPU cores for the Linux Kernel NVMe-oF target. The graph below shows the relative performance in terms of IOPS/core which was calculated by dividing the total aggregate IOPS by the total CPU cores used while running that specific workload. For the case of Kernel NVMe-oF target, total CPU cores was calculated from % CPU utilization which was measured using sar utility in Linux.

4k Random Read results

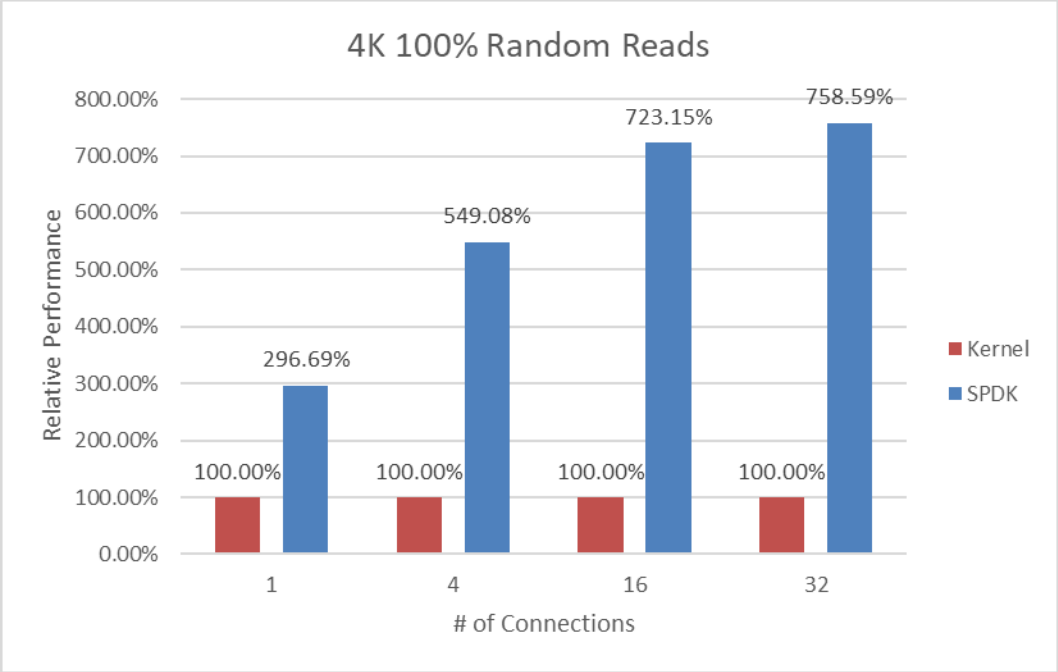


Figure 10: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF RDMA Target for 4K 100% Random Read QD=32, using Kernel Initiators

Linux Kernel NVMe-oF RDMA Target: 4K 100% Random Reads, QD=32

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	13128.49	3360.9	152.2	12.3
4	15865.47	4061.6	503.8	23.1
16	15039.53	3850.1	2126.9	33.6
32	15029.68	3847.5	4257.4	35.6

SPDK NVMe-oF RDMA Target: 4K 100% Random Reads, QD=32

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	12623.24	3231.5	158.2	4
4	15062.79	3856.1	530.7	4
16	12957.79	3317.2	2468.7	4
32	12807.11	3278.6	4996.4	4

4k Random Write results

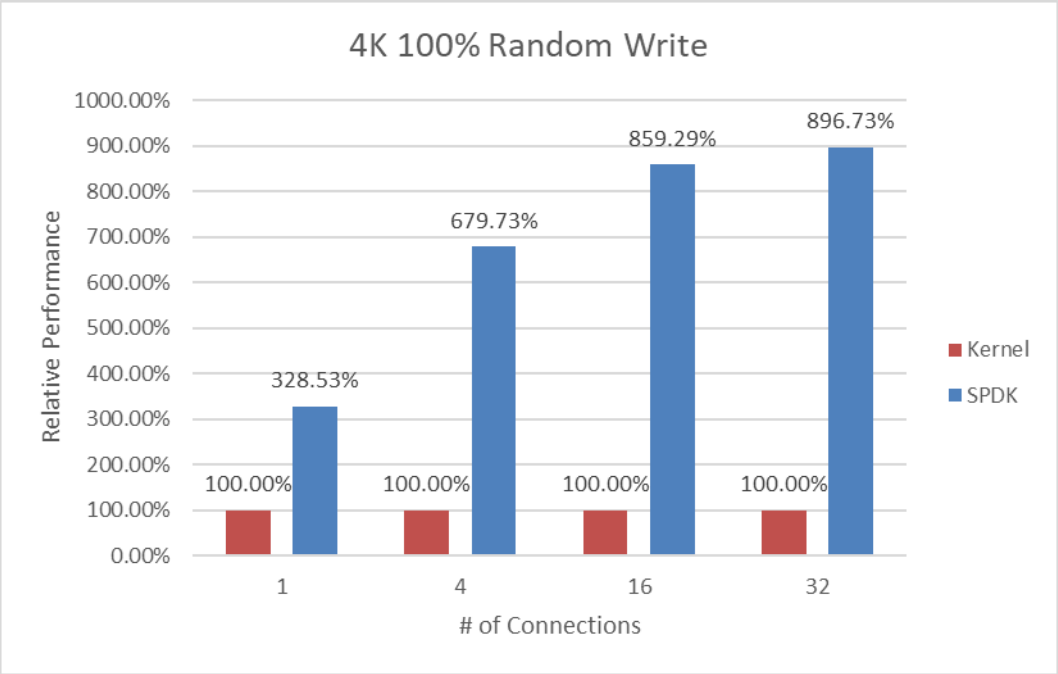


Figure 11: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF RDMA Target for 4K 100% Random Write QD=32, using Kernel Initiators

Note: The SSDs were not pre-conditioned before running 100% Random Write I/O test.

Linux Kernel NVMe-oF RDMA Target: 4K 100% Random Writes, QD=32

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	15457.49	3957.1	129.1	12.9
4	15229.33	3898.7	524.9	20.3
16	14282.56	3656.3	2239.8	27.0
32	14455.96	3700.7	4426.7	28.6

SPDK NVMe-oF RDMA Target: 4K 100% Random Writes, QD=32

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	15739.83	4029.4	126.8	4
4	20390.29	5219.9	392.2	4
16	18197.51	4658.5	1757.7	4
32	18132.29	4641.8	3528.9	4

4k Random Read-Write results

For 4K Random 70% reads, 30% writes workload it was noticed that relative performance didn't increase when going from 4 to 16 connections per subsystem.

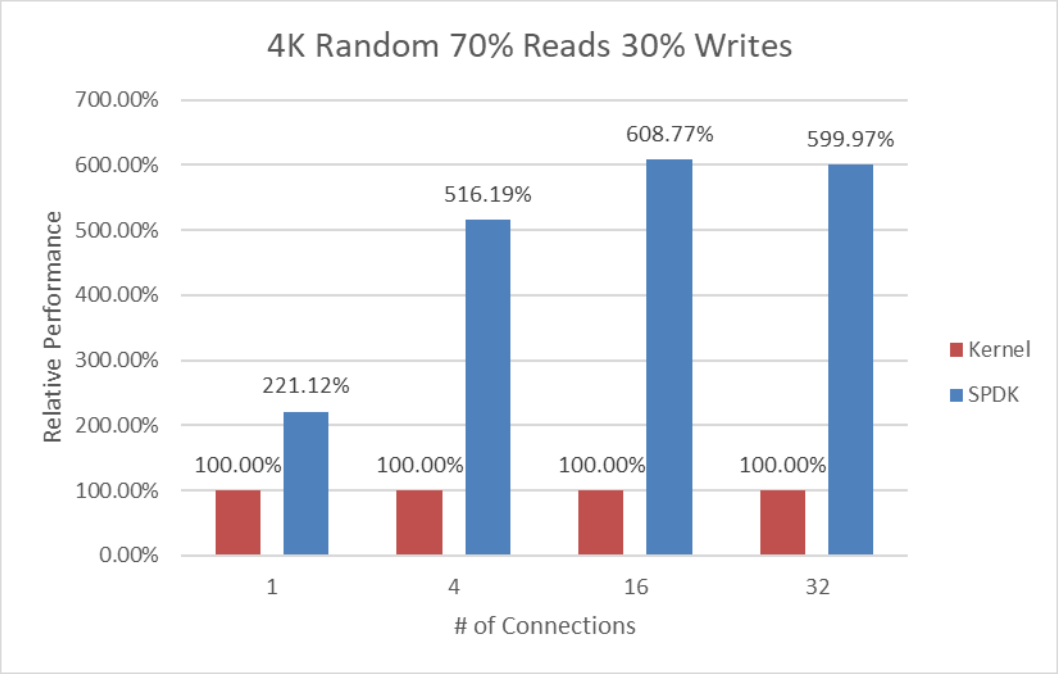


Figure 12: Relative Performance Comparison ofLinux Kernel vs. SPDK NVMe-oF RDMA Target for 4K Random 70% Read 30% Write QD=32, using Kernel Initiators

Linux Kernel NVMe-oF RDMA Target: 4K 70% Random Read 30% Random Write, QD=32

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	9585.10	2453.8	43.0	9.1
4	19032.34	4872.3	79.9	24.0
16	18856.80	4827.3	1327.1	34.9
32	18689.05	4784.3	2918.7	36.2

SPDK NVMe-oF RDMA Target: 4K 70% Random Read 30% Random Write, QD=32

Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	9359.04	2395.9	51.8	4
4	16407.02	4200.2	219.8	4
16	13172.05	3372.0	2225.0	4
32	12401.06	3174.6	4811.8	4

Conclusions

1. When the SPDK NVMe-oF Target was configured with 4 CPU cores the performance peaked when the number of connections was 4 per subsystem. Increasing the number of connection results in increase in average latency and IOPS drop.
2. Similarly, the performance for the Linux Kernel NVMe-oF Target peaks at 4 connections per subsystem; increasing the number of connections only increases the latency. The aggregate IOPS dropped is slightly and not as noticable as in the SPDK case.

3. For all workloads SDPK performance is similar to last 19.04 report in terms of numbers, but it's worth noting that this time a different CPU (less powerful) was used for the target platform. This can suggest that actual SPDK performance improved.

4. The SPDK NVMe-oF target shows 6-8x more IOPS/Core relative to the Linux Kernel NVMe-oF target as the number of connections per subsystem increased.

Summary

This report showcased performance results with SPDK NVMe-oF RDMA 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 the Linux Kernel NVMe-oF RDMA (Target/Initiator) against the accelerated polled-mode driven SPDK NVMe-oF RDMA (Target/Initiator) implementation. Like in the last report, throughput scales up and latency decreases almost linearly with the scaling of SPDK NVMe-oF target cores. In case of SDPK NVMe-OF Initiator core scaling it was observed that throughput scales almost linearly, but the latency remains rather stable which might suggest that the Target side might be able to process slightly more IO.

In case of 4k Random 70% Read 30% Write workload there was an unidentified bottleneck, which prevents from confirming proper throughput and latency scaling for this workload.

It was also observed that the SPDK NVMe-oF Target average latency is up to 6 usec lower than Kernel when testing against null bdev based backend. The advantage of SPDK is even greater when comparing NVMe-oF Initiators: the SPDK NVMe-oF RDMA average latency is up to 3.25 times lower than Kernel initiator.

The SPDK NVMe-oF Target performed up to 8 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.

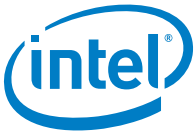
This report provides information regarding methodologies and practices while benchmarking NVMe-oF using SPDK, as well as the Linux Kernel. It should be noted that the performance data showcased in this report is based on specific hardware and software configurations and that performance results may vary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

Appendix A

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

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

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




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

```
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode13"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4433",
        "trtype": "rdma"
      },
      "portid": 14,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode14"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4434",
        "trtype": "rdma"
      },
      "portid": 15,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode15"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4435",
        "trtype": "rdma"
      },
      "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"
    }
  ]
}
```

Notices and Disclaimers

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. For more complete information about performance and benchmark results, visit http://www.intel.com/benchmarks.

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.
No product or component can be absolutely secure.

§