SPDK NVMe-oF RDMA (Target & Initiator) Performance Report Release 19.04

Testing Date: June 2019

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

Acknowledgments:

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

Revision History

Date	Revision	Comment
19/05/15	v. 1.0	Complete performance runs
19/05/15	v. 1.0	Review
19/05/27	v. 1.1	Reviews incorporated
19/06/07	v. 1.2	Finished test case 1 & 2 re-run for more CPU cores
19/06/10	v. 1.3	Review
19/06/17	v. 1.4	Reviews incorporated

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® Platinum 8180 Processor (38.5MB L3, 2.50 GHz) Number of cores 28, number of threads 56
Memory	Total 376 GBs
Operating System	Fedora 28
BIOS	2.1a 08/23/2018
Linux kernel version	5.0.9-100.fc28
SPDK version	SPDK 19.04 (f19fea15c)
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	Total 64GBs
Operating System	Fedora 28
BIOS	3.1 06/08/2018
Linux kernel version	5.0.9-100.fc28
SPDK version	SPDK 19.04 (f19fea15c)
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	Total 64GBs
Operating System	Fedora 28
BIOS	3.1 06/08/2018
Linux kernel version	5.0.9-100.fc28
SPDK version	SPDK 19.04 (f19fea15c)
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 <Performance> 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 5.0.9-100.fc28 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).

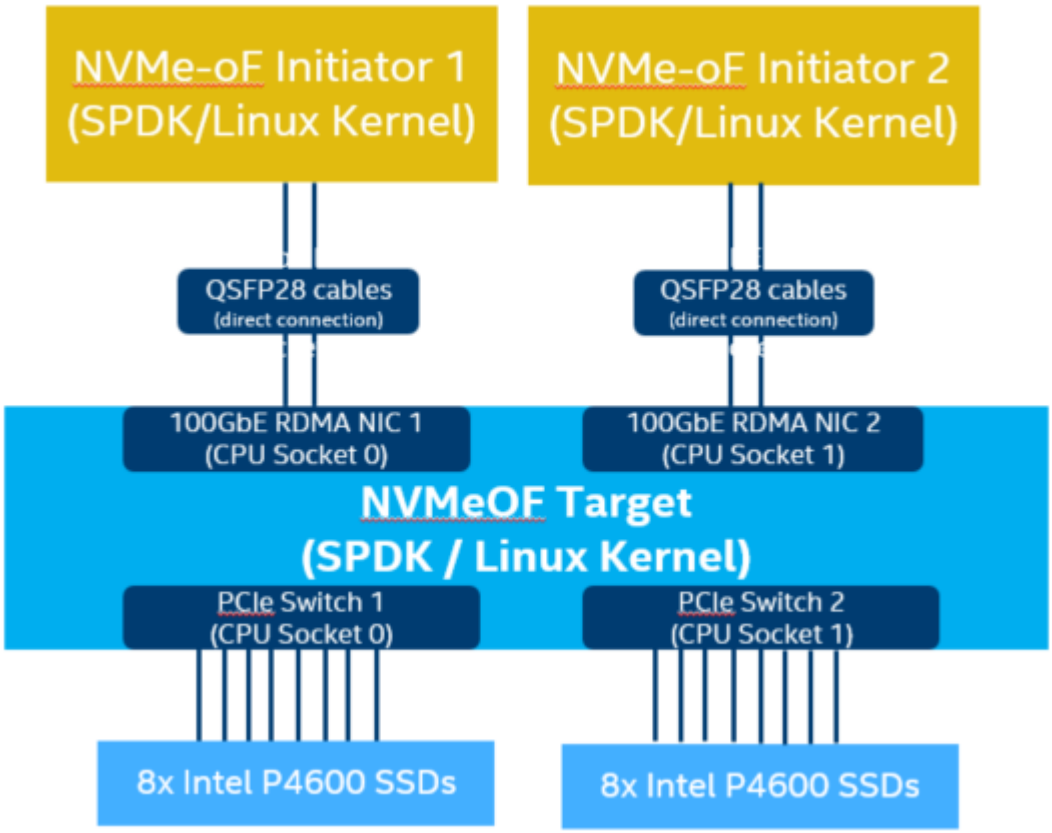

Figure 1: High level NVMe-oF performance testing setup

19.04 NVMe-OF New features

SRQ

Support for per-device Shared Receive Queues in the RDMA transport layer has been added in 19.04 release. It is enabled by default when creating a RDMA transport layer for any device that supports it. For purpose of creating this and any future SPDK NVMe-oF RDMA report this feature will be enabled.

SRQ reduces the number of communication buffers and improves memory utilization: Instead of using multiple per-sender queues a single receive queue is used, with its own set of buffers. Enabling SRQ comes with a slight performance drop as more operations need to be performed on incoming data in order to properly distribute. Performance is 1-10% lower than for a configuration without SRQ and is dependent on workload. Example of such drop was presented in Test Case 1 results.

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

This test case was performed in order to understand the performance of SPDK NVMe-oF target with I/O core scaling. SPDK NVMe-oF 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 ReactorMask 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=32

	SPDK v19.01.1			SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4851.77	1242.1	411.9	4483.50	1147.8	445.8
2 cores	10136.65	2595.0	197.5	10017.61	2564.5	201.0
3 cores	14306.15	3662.4	139.4	13841.95	3543.5	144.1
4 cores	16296.15	4171.8	122.5	15703.81	4020.2	127.1
5 cores	Not run	Not run	Not run	16280.91	4167.9	122.6
6 cores	Not run	Not run	Not run	16713.24	4278.6	119.4

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

	SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4255.57	1089.4	939.6
2 cores	9856.26	2523.2	405.3
3 cores	14963.67	3830.7	269.1
4 cores	19874.40	5087.8	201.3
5 cores	21560.06	5519.4	185.5
6 cores	21648.04	5541.9	184.9

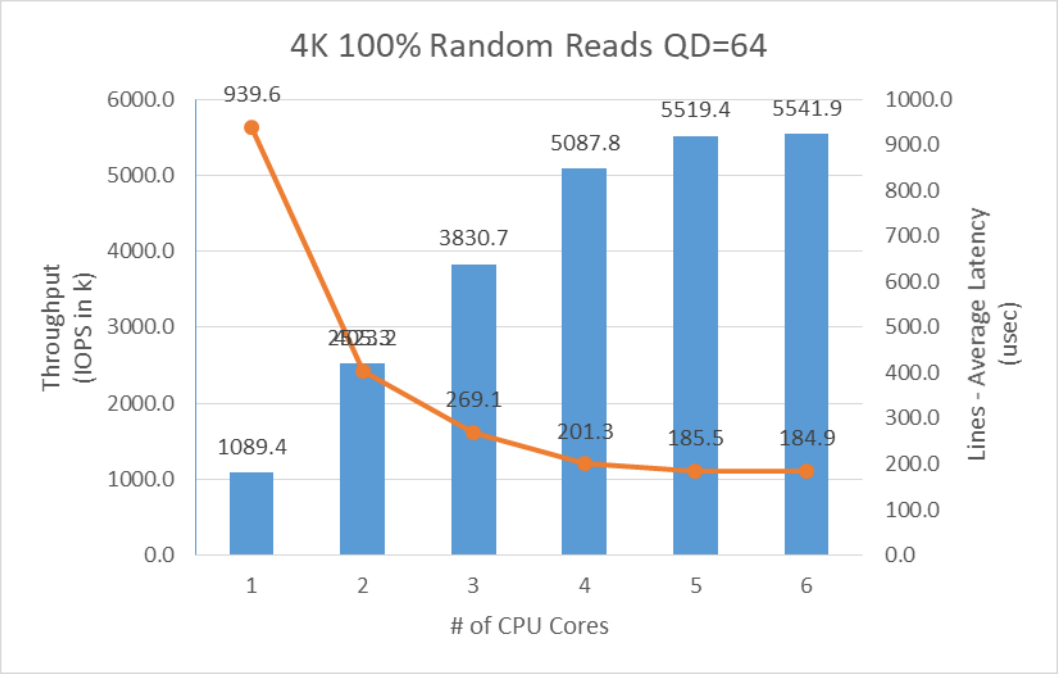

Figure 2: SPDK NVMe-oF Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random read workload

Test Result: 4K 100% Random Read IOPS QD=64, SRQ Disabled vs SRQ Enabled

	SPDK v19.04, SRQ Disabled			SPDK v19.04, SRQ Enabled			SRQ Enabled vs Disabled Relative Performance		
# of Cores	BW (MBps)	Throughput (IOPS k)	Avg. Lat. (usec)	BW (MBps)	Throughput (IOPS k)	Avg. Lat. (usec)	BW (%)	IOPS (%)	Avg. Lat. (%)
4 cores	21779.99	5575.7	183.4	19874.40	5087.8	201.3	91.25	91.25	109.72
5 cores	23259.03	5954.3	171.7	21560.06	5519.4	185.5	92.70	92.70	107.98
6 cores	24056.69	6158.5	166.0	21648.04	5541.9	184.9	89.99	89.99	111.38

4k Random Write results

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

	SPDK v19.01.1			SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	6148.64	1574.0	324.1	6092.40	1559.7	327.2
2 cores	12674.51	3244.7	156.0	13060.29	3343.4	151.7
3 cores	18907.93	4840.4	104.1	19395.47	4965.2	101.6
4 cores	22560.17	5775.4	87.4	22232.87	5691.6	89.0
5 cores	Not run	Not run	Not run	21785.05	5577.0	91.3
6 cores	Not run	Not run	Not run	22466.59	5751.4	88.6

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

	SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5986.38	1532.5	666.9
2 cores	12801.63	3277.2	310.7
3 cores	19599.50	5017.5	202.3
4 cores	21569.54	5521.8	184.6
5 cores	21424.65	5484.7	186.2
6 cores	22163.89	5674.0	180.1

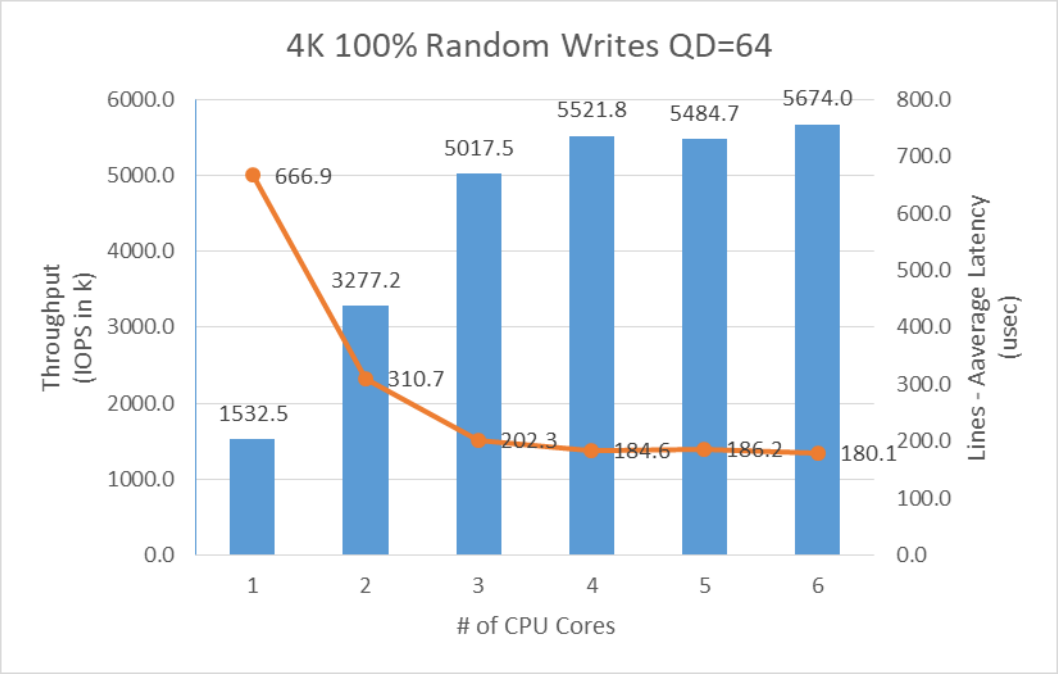

Figure 3: SPDK NVMe-oF Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random write workload

4k Random Read-Write results

Test Result: 4K 70% Read 30% Write IOPS, QD=32

	SPDK v19.01.1			SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5072.15	1298.5	393.8	4682.29	1198.7	426.8
2 cores	9866.31	2525.8	202.1	9478.18	2426.4	210.5
3 cores	10175.64	2605.0	196.2	10078.54	2580.1	198.1
4 cores	10400.38	2662.5	192.1	10160.75	2601.1	196.6
5 cores	Not run	Not run	Not run	10224.87	2617.6	195.3
6 cores	Not run	Not run	Not run	10242.21	2622.0	195.0

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

	SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4449.80	1139.1	898.6
2 cores	9860.58	2524.3	407.0
3 cores	13483.08	3451.7	296.1
4 cores	14206.02	3636.7	281.2
5 cores	14349.54	3673.5	278.5
6 cores	14413.66	3689.9	277.2

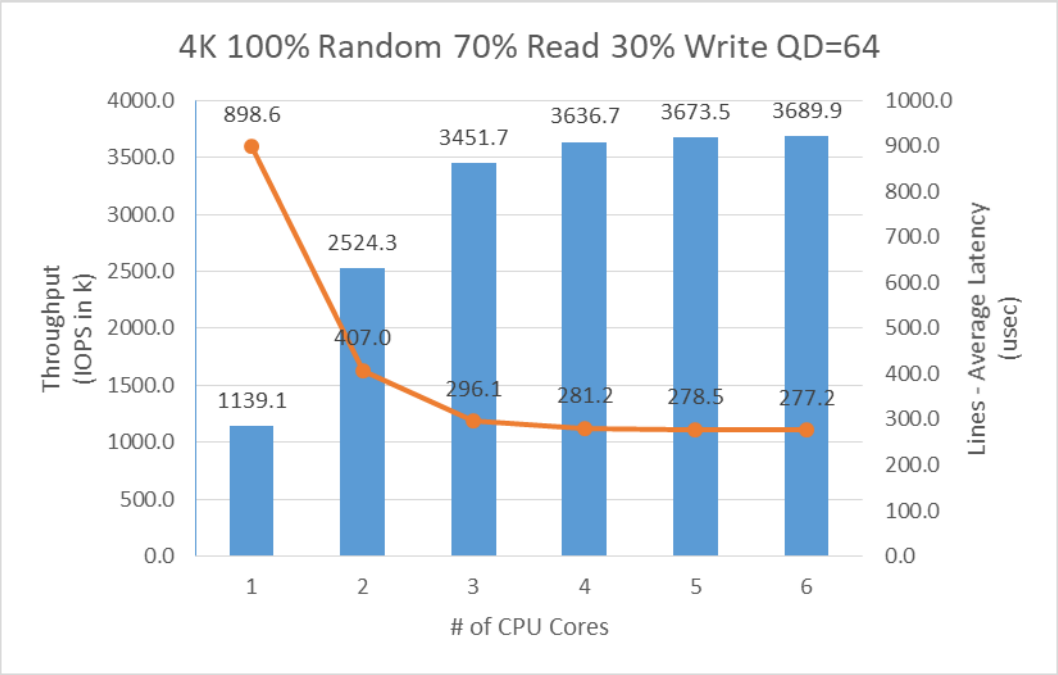

Figure 4: SPDK NVMe-oF Target I/O core scaling: IOPS vs. Latency while running 4KB Random 70% read 30% write workload

Conclusion:

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 halved. Increasing from 5 to 6 CPU cores does not show any benefit as we have reached network saturation.

2. For 100% Random Writes, the results are almost the same as for 100% Random Reads, except that saturation is reached quicker – at 4 CPU cores. Increasing to 5 or 6 CPU cores does not improve results.

3. For 4K Random 70/30 Reads/Writes, performance doesn't scale from 3 to more cores due to some other bottleneck. It was observed that while running this test case locally without involving any network it can hit > 4M aggregate IOPS. But, while running this over the network, it could only achieve 3.8M IOPS max. This points to some other platform or network bottleneck while running this workload.

4. The SRQ transport layer feature that was introduced in 19.04 caused a slight performance degradation vs. 19.01.1. SRQ is enabled by default in 19.04 can cause up to 10% of performance degradation. However, SRQ decreases memory utilization significantly so the small degradation in the IOPS and/or latency is a worthy trade-off. The benefits of using SRQ are realized when the NVMe-oF target has a high number of connections, which is not the case, nor aim of this document.

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

	SPDK v19.01.1			SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	24962.03	199.7	640.8	25053.41	200.4	638.4
2 cores	24947.14	199.6	641.1	25050.36	200.4	638.5
3 cores	24928.21	199.4	641.6	25047.21	200.4	638.6
4 cores	24910.79	199.3	642.1	25040.83	200.3	638.7

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

	SPDK v19.01.1			SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	23516.53	188.1	680.1	23610.35	188.9	677.4
2 cores	23606.75	188.9	677.5	23559.83	188.5	678.8
3 cores	23580.76	188.6	678.2	23593.33	188.7	677.9
4 cores	23543.86	188.4	679.3	23562.77	188.5	678.7

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

	SPDK v19.01.1			SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	29842.07	238.7	623.8	30123.13	241.0	625.4
2 cores	29857.59	238.9	620.3	30119.92	241.0	619.8
3 cores	29087.92	232.7	625.0	29629.22	237.0	621.4
4 cores	30261.42	242.1	623.6	29514.49	236.1	619.2

Conclusion:

1. A single CPU core saturated the network bandwidth. The SPDK NVMe-oF target running on 1 core does close to 24 GBps 100% Writes and 25 GBps of 100% Reads. The 70-30 Read/Write workload bandwidth was 29-30GBps which means 2x 100GbE NICs network bandwidth was saturated. Therefore, adding more CPU cores did not result in increased performance for these workloads because the network was the bottleneck.

2. Results comparable to 19.01.1.

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

This test case was performed in order to understand the performance of SPDK NVMe-oF Initiator with I/O core scaling. SPDK NVMe-oF Target was configured similar to test case 1, running using 6 cores. SPDK bdev FIO plugin was used to target 8 individual NVMe-oF bdevs on each of the 2 initiators. FIO cpumask was varied in order to run 1, 2, 3 and 4 cores tests while running following workloads from both of the initiators using fio client-server mode.

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

Depending on 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 available 16 target subsystems, which resulted in unnatural latency spike in the 1 CPU test case.

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

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

3 cores: 2 separate 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 separate initiators, both running 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. Different filename section settings (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

Results in the table represent aggregate performance (IOPS & Avg. latency) observed:

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

	SPDK v19.01.1			SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4968.21	1271.9	195.8	5683.05	1454.9	156.3
2 cores	10947.22	2802.5	175.1	11095.00	2840.3	161.1
3 cores	16797.94	4300.3	171.6	14734.99	3772.2	180.3
4 cores	19080.54	4884.6	210.0	17429.95	4462.1	225.5

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

	SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4915.75	1258.4	150.1
2 cores	10539.08	2698.0	154.1
3 cores	15838.41	4054.6	162.7
4 cores	20702.37	5299.8	176.5

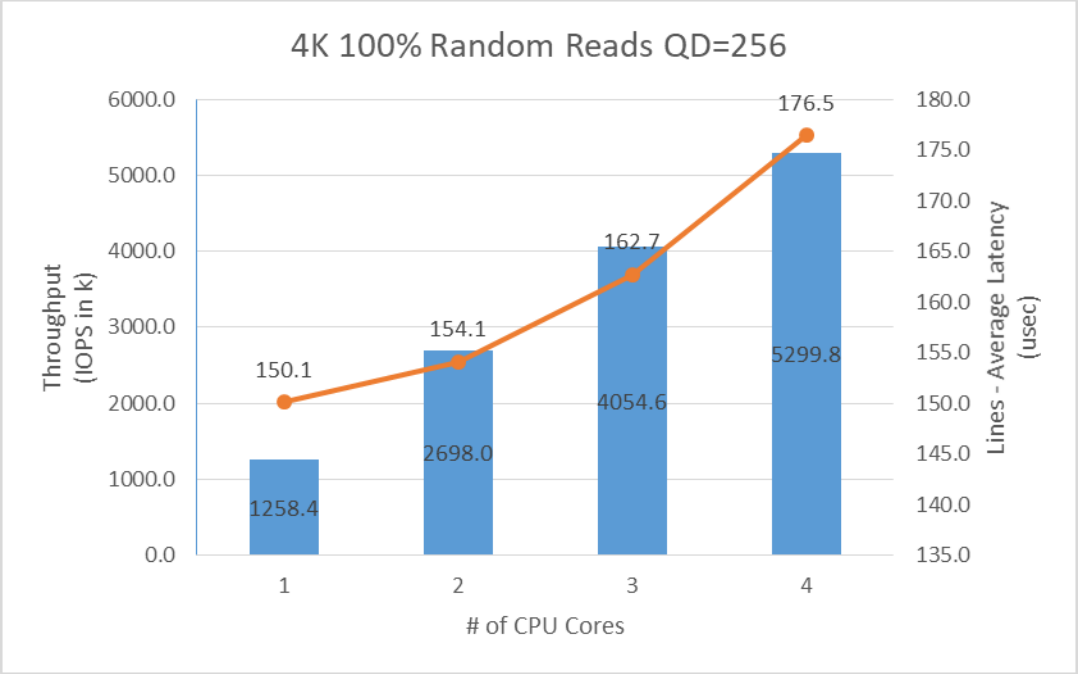

Figure 5: SPDK NVMe-oF Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random read workload

4k Random Write results

Note: The SSDs were pre-conditioned just once before running all the 100% Random Write test cases. This helped scale throughput to the 2x 100GbE network bandwidth when testing with 3 and 4 CPU cores rather than hitting the storage bottleneck

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

	SPDK v19.01.1			SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4826.91	1235.7	197.4	5305.06	1358.1	115.7
2 cores	10407.71	2664.4	174.7	10086.05	2582.0	119.0
3 cores	15685.55	4015.5	168.5	13897.14	3557.7	147.1
4 cores	20242.84	5182.2	170.6	18870.83	4830.9	160.5

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

	SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4613.72	1181.1	112.7
2 cores	9958.48	2549.4	115.1
3 cores	15075.57	3859.3	117.1
4 cores	19369.23	4958.5	128.4

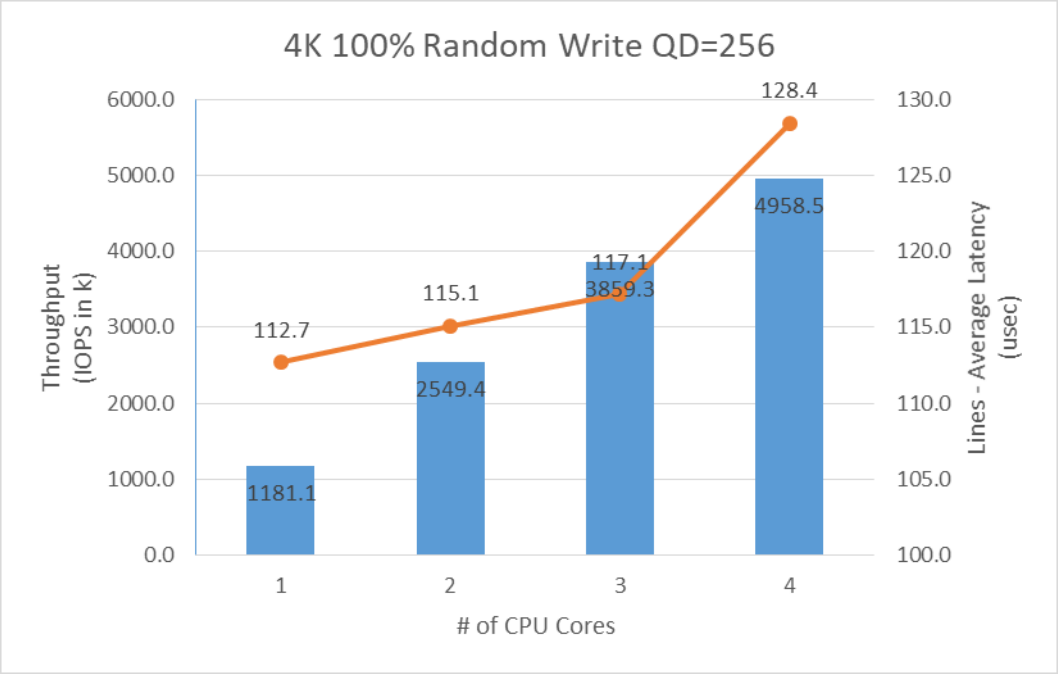

Figure 6: SPDK NVMe-oF Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write workload

4k Random Read-Write results

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

	SPDK v19.01.1			SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4570.25	1170.0	161.4	3649.75	934.3	126.8
2 cores	9257.49	2369.9	114.2	7198.95	1842.9	128.1
3 cores	11989.00	3069.2	119.5	11644.98	2981.1	127.9
4 cores	14385.84	3682.8	134.4	14243.11	3646.2	138.6

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

	SPDK v19.04		
# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	4252.29	1088.6	104.2
2 cores	9207.31	2357.1	106.3
3 cores	11963.40	3062.6	116.0
4 cores	14115.10	3613.5	134.2

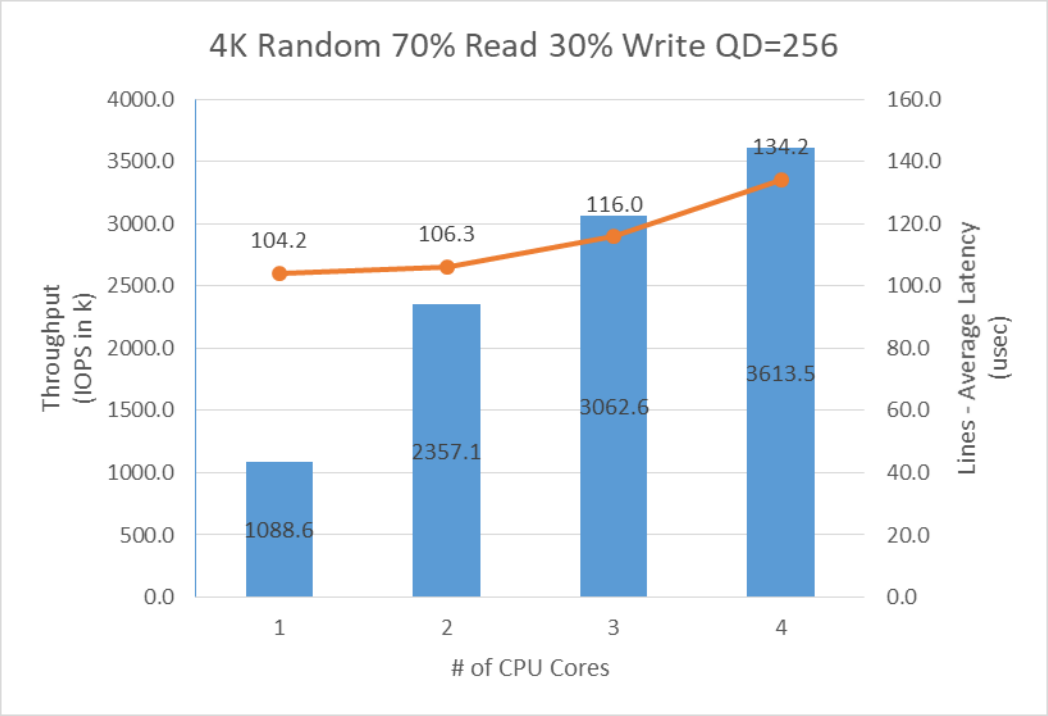

Figure 7: SPDK NVMe-oF Initiator I/O core scaling: IOPS vs. Latency while running 4KB Random 70% read 30% write workload

Conclusion:

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 scale linearly from 1 to 2 CPUs. For 3 and 4 CPUs the performance increase is notlinear, and the results 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 Latency

This test case was designed to understand latency characteristics of SPDK NVMe-oF Target and Initiator vs. the Linux Kernel NVMe-oF 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": [
	"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: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
Kernel initiator configuration	**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** [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 Target results

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

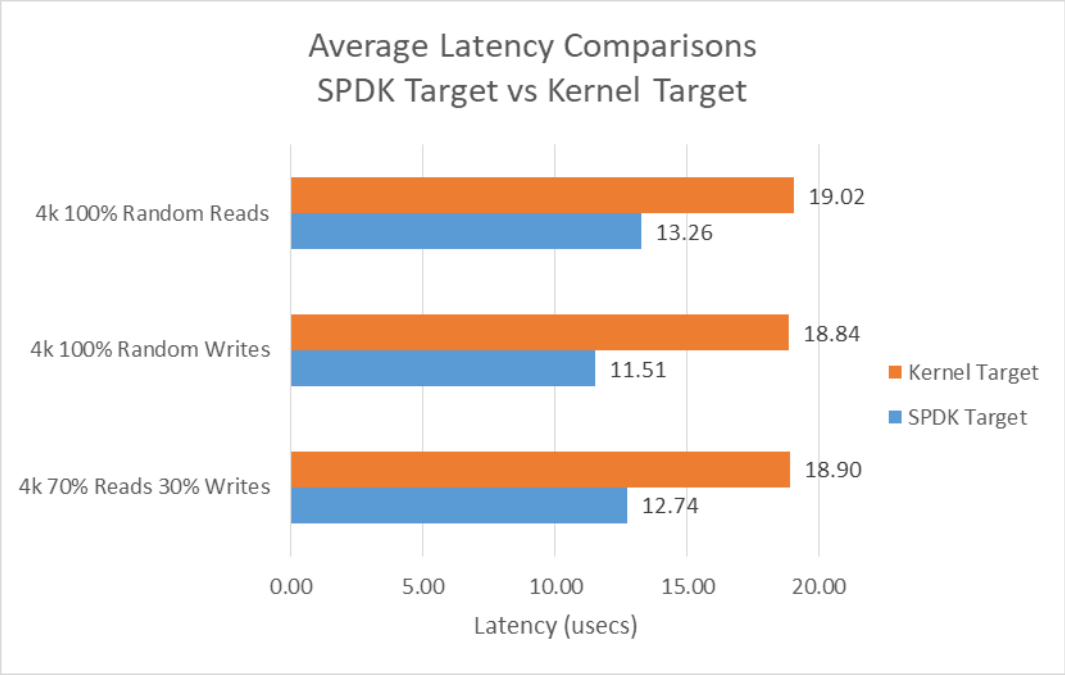

Figure 8: Average I/O Latency comparisons between SPDK and Kernel NVMe-oF Target for various workloads using the Kernel Initiator

SPDK NVMe-oF Target

	SPDK v19.01.1			SPDK v19.04		
Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	13.32	73122	24.0	12.74	76715	23.7
4K 100% Random Writes IOPS	11.75	82469	21.0	11.51	84682	22.1
4K 100% Random 70% Reads 30% Writes IOPS	13.38	73051	24.4	13.26	73846	24.7

Linux Kernel NVMe-oF Target

	Kernel 4.20			Kernel 5.0.9		
Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	19.66	49485	29.0	18.90	52007	27.6
4K 100% Random Writes IOPS	18.26	53176	24.9	18.84	52166	29.8

4K 100% Random 70% Reads 30% Writes IOPS	19.33	50355	24.8	19.02	51736	27.9

Conclusion:

1. The SPDK NVMe-oF Target reduces the NVMe-oF average round trip I/O latency (reads/writes) by up to 7 usec vs. the Linux Kernel NVMe-oF target. This is entirely software overhead.
2. Both Kernel and SPDK Target performance in this test case remains stable since 19.01.1 report.

SPDK vs Kernel Initiator results

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

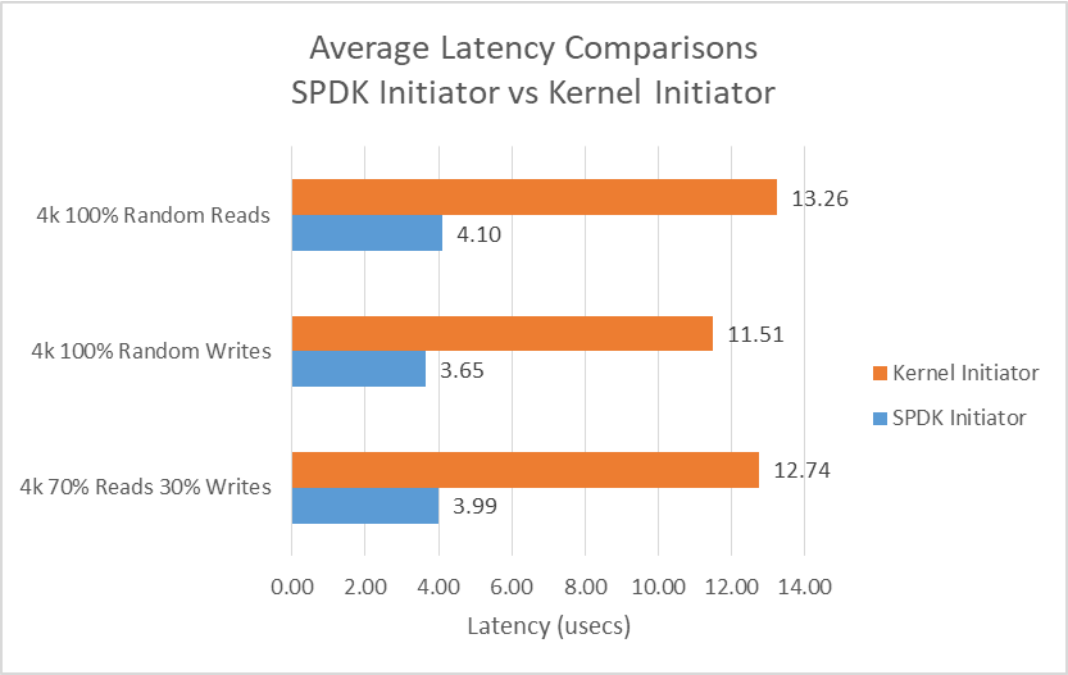

Figure 9: Average I/O Latency Comparisons between SPDK and Kernel NVMe-oF Initiator for various workloads against SPDK Target

Linux Kernel NVMe-oF Initiator

	Kernel 4.20			Kernel 5.0.9		
Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	13.32	73122	24.0	12.74	76715	23.7
4K 100% Random Writes IOPS	11.75	82469	22.7	11.51	84682	22.5
4K 100% Random 70% Reads 30% Writes IOPS	13.38	73051	24.4	13.26	73846	24.7

SPDK NVMe-oF Initiator

	SPDK v19.01.1			SPDK v19.04		
Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	Average Latency (usec)	IOPS	p99 (usec)
4K 100% Random Reads IOPS	4.23	226639	5.0	3.99	239247	4.1
4K 100% Random Writes IOPS	3.77	253694	5.6	3.65	260628	3.7
4K 100% Random 70% Reads 30% Writes IOPS	4.34	221712	4.8	4.10	233923	4.2

Conclusion:

1. SPDK NVMe-oF Initiator reduces the NVMe-oF SW overhead by up to 4x vs. the Linux Kernel NVMe-oF Initiator.

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 NVMe-oF Target vs. Linux Kernel NVMe-oF 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 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 initiator.

- 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 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

Number of CPU cores used while running SPDK NVMe-oF target were 4, whereas for the case of Linux Kernel NVMe-oF target there was no CPU core limitation applied.

Numbers in the graph represent relative performance which are in terms of IOPS/core which was calculated based on total aggregate IOPS divided by 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

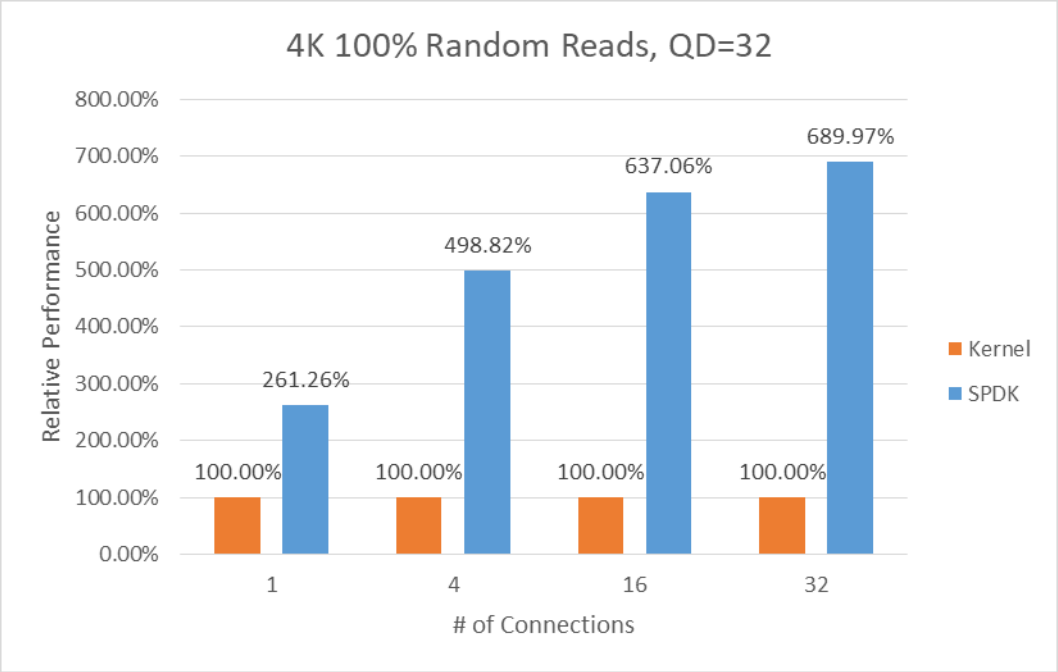

Figure 10: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF Target for 4K 100% Random Reads using the Kernel Initiator

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

	Kernel 4.20				Kernel 5.0.9			
Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	13153.28	3367.2	151.9	12.0	14443.91	3697.6	138.2	11.4
4	14866.64	3805.9	537.7	19.8	15992.04	4094.0	499.8	20.8
16	13823.73	3538.8	2314.5	26.1	15045.08	3851.5	2126.1	30.4
32	13287.02	3401.4	4816.2	30.4	15047.04	3852.0	4252.6	33.2

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

	SPDK v19.01.1				SPDK v19.04			
Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	13062.58	3344.0	153.0	4	13209.44	3381.6	151.1	4
4	15445.60	3954.1	517.8	4	15318.58	3921.5	521.8	4
16	12859.08	3291.9	2489.4	4	12615.75	3229.6	2535.8	4
32	11512.27	2947.1	5579.9	4	12508.68	3202.2	5115.8	4

4k Random Write results

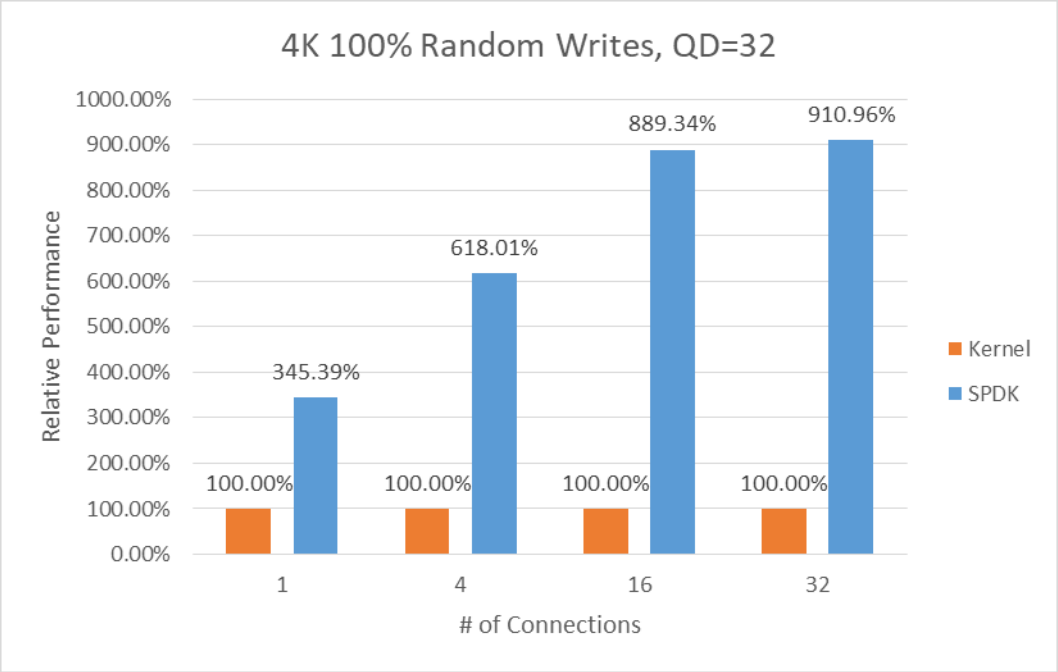

Figure 11: Relative Performance Comparison of Linux Kernel vs. SPDK NVMe-oF Target for 4K 100% Random Writes

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

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

	Kernel 4.20				Kernel 5.0.9			
Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	16219.94	4152.3	123.0	13.5	18558.84	4751.1	107.5	14.0
4	15008.11	3842.1	532.8	17.2	14334.90	3669.7	557.6	16.9
16	14271.00	3653.3	2241.9	23.4	13313.63	3408.3	2402.8	24.4
32	14070.91	3602.1	4560.0	29.1	13603.83	3482.5	4704.0	25.4

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

	SPDK v19.01.1				SPDK v19.04			
Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	11934.90	3055.3	172.0	4	18299.48	4684.7	109.1	4
4	20922.56	5356.2	382.3	4	20924.68	5356.7	382.0	4
16	17382.37	4449.9	1843.0	4	19430.11	4974.1	1646.3	4
32	16974.95	4345.5	3769.5	4	19520.97	4997.3	3277.8	4

4k Random Read-Write results

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

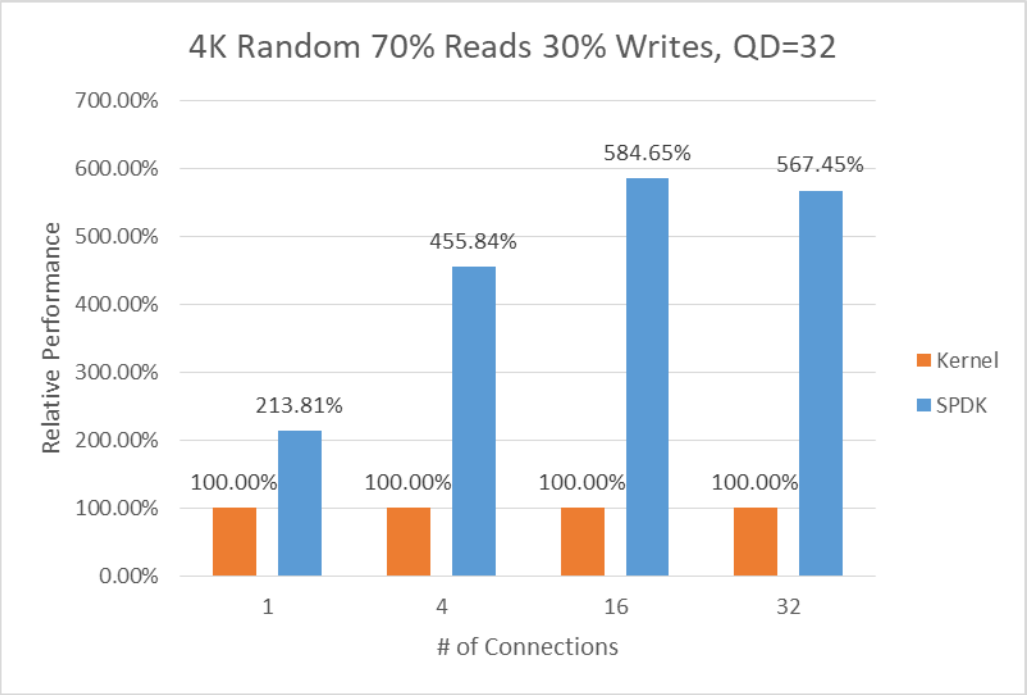

Figure 12: Relative Performance Comparison ofLinux Kernel vs. SPDK NVMe-oF Target for 4K Random 70% Reads 30% Writes

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

	Kernel 4.20				Kernel 5.0.9			
Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	9333.25	2389.3	46.2	8.4	9843.75	2520.0	34.4	8.7
4	19229.42	4922.7	121.0	21.1	19255.24	4929.3	77.9	21.6
16	16913.31	4329.8	1474.2	32.2	18629.71	4769.2	1353.0	32.2
32	16648.65	4262.0	3289.4	39.0	18653.88	4775.3	2922.1	32.6

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

	SPDK v19.01.1				SPDK v19.04			
Connections per subsystem	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores	Bandwidth (MBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	9909.67	2536.9	46.9	4	9689.06	2480.4	43.5	4
4	16768.54	4292.7	193.3	4	16235.37	4156.2	233.0	4

16	13405.55	3431.8	2132.7	4	13522.21	3461.7	2140.5	4
32	11601.18	2969.8	5241.8	4	12968.30	3319.8	4533.2	4

Conclusion:

1. The performance of the 4K Random 70/30 Read/Write workload peaked when the number of connections was 4 and iodepth of 4 per subsystem. This seem to be the optimum configuration that yield the maximum performance for the 4K 100% Random Reads workload as well.

2. For the 4K Random Read workload, the Linux Kernel NVMe-oF target performance in IOPS, BW and Latency improved, but at a cost of slightly higher CPU utilization. SPDK performance remains similar to 19.01.1 for low number of connections, however, there was a slight performance improvement observed at 16 and 32 connections. These changes are reflected in the chart: relative performance for 1 and 4 connections is similar to last report, while it has risen for 16 and 32 connections. SPDK NVMe-oF target performance is up to 7 times better than kernel.

3. For the 4K 100% random writes workloads both SPDK and Kernel NVMe-oF target showed increased performance for a single connection. As the number of connection increased, the Linux Kernel NVMe-oF target performance did not change, but generally the CPU utilization decreased slightly between Linux Kernel 4.20 and 5.0.9. SPDK NVMe-oF target performance has improved since 19.01.1, and the relative performance is up to 9 times better than the Linux Kernel NVMe-oF target.

4. For 70/30 Random Read/Write workload, the Linux Kernel overall performance increased noticeably. This could probably be a result of higher random read results (IOPS, BW and Latency values) and lower CPU utilization for random writes. SPDK performance is comparable to 19.01.1, and the relative performance is up to 5.8 times better than the Linux Kernel NVMe-oF target.

Summary

This report showcased performance results with SPDK NVMe-oF 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. 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 in case of Kernel vs SPDK comparison, the SPDK NVMe-oF Target average latency is up to 7 usec lower than Kernel when testing against null bdev based backend. The advantage of SPDK is even greater when comparing NVMe-oF Initiators – SPDK average latency is up to 4 times lower than Kernel Initiator.

SPDK NVMe-oF Target performed up to 6.5 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 (16) 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 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"
    }
  ]
}
```

DISCLAIMERS

§