SPDK NVMe-oF RDMA (Target & Initiator) Performance Report Release 23.05

Intel E810-CQDA2 RoCEv2 version

Testing Date: June 2023

Performed by:

Jaroslaw Chachulski (jaroslawx.chachulski@intel.com)

Karol Latecki (karol.latecki@intel.com)

Acknowledgments:

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

Tomasz Zawadzki (tomasz.zawadzki@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 of the SPDK vs. Linux Kernel NVMe-oF Target and Initiator for the RDMA transport only.

The purpose of report 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 and Initiator components.

Test setup

Target Configuration

Table 1: Hardware setup configuration – Target system

Item	Description
Server Platform	SuperMicro® Ultra SuperServer SYS-220U-TNR 
Motherboard	Server board **X12DPU-6**
CPU	2 CPU sockets, Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz Number of cores 28 per socket, number of threads 56 per socket Both sockets populated Microcode: 0xd000389
Memory	16 x 32GB SK Hynix HMA84GR7DJR4N-XN, DDR4, 3200MHz Total of 512GB
Operating System	Fedora 37
BIOS	1.4b
Linux kernel version	6.0.18-300.fc37.x86_64 Spectre-meltdown mitigations enabled
SPDK version	SPDK 23.05
Storage	**OS:** 1x 250GB Crucial CT250MX500SSD1 **Storage Target**: 14x Kioxia® KCM61VUL3T20 3.2TBs (FW: 0105) (6 on CPU NUMA Node 0, 8 on CPU NUMA Node 1)
NIC	4x 100GbE Intel(R) Ethernet Network Adapter E810-CQDA2. Single port connected on each NIC. ice driver version: 1.11.14 irdma driver version: 1.11.58 NVM FW version: v4.20 2 NICs per CPU socket. Protocol: RoCEv2

Initiator 1 Configuration

Table 2: Hardware setup configuration – Initiator system 1

Item	Description
Server Platform	Intel® Server System M50CYP2UR208
CPU	Intel® Xeon® Gold 6348 Processor @ 2.60GHz (42MB Cache) Number of cores 28 per socket, number of threads 56 per socket (Both sockets populated) Microcode: 0xd00037b
Memory	16 x 32GB Micron 36ASF4G72PZ-3G2J3, DDR4, 3200MHz Total 512GBs
Operating System	Fedora 37
BIOS	SE5C620.86B.01.01.0007.2210270543
Linux kernel version	6.0.18-300.fc37.x86_64 Spectre-meltdown mitigations enabled
SPDK version	SPDK 23.05
Storage	**OS:** 1x 250GB Crucial CT250MX500SSD1
NIC	2x 100GbE Intel(R) Ethernet Network Adapter E810-CQDA2. Single port connected on each NIC. ice driver version: 1.11.14 irdma driver version: 1.11.58 NVM FW version: v4.20 Protocol: RoCEv2

Initiator 2 Configuration

Table 3: Hardware setup configuration – Initiator system 2

Item	Description
Server Platform	Intel® Server System M50CYP2UR208
CPU	Intel® Xeon® Gold 6348 Processor @ 2.60GHz (42MB Cache) Number of cores 28 per socket, number of threads 56 per socket (Both sockets populated) Microcode: 0x0xd00037b
Memory	16 x 32GB Micron 36ASF4G72PZ-3G2J3, DDR4, 3200MHz Total 512GBs
Operating System	Fedora 37
BIOS	SE5C620.86B.01.01.0007.2210270543
Linux kernel version	6.0.18-300.fc37.x86_64 Spectre-meltdown mitigations enabled
SPDK version	SPDK 23.05
Storage	**OS:** 1x 250GB Crucial CT250MX500SSD1
NIC	2x 100GbE Intel(R) Ethernet Network Adapter E810-CQDA2. Single port connected on each NIC. ice driver version: 1.11.14

	irdma driver version: 1.11.58 NVM FW version: v4.20 Protocol: RoCEv2

BIOS settings

Table 4: Test systems 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

SPDK Build Options

All measurements included in this report document were done with SPDK build with "--enable-lto" option enabled. Link time optimization allows better SPDK performance thanks to code optimization done by inlining functions across compilation units, which in turn results in reduced function call overhead.

SPDK 23.05 iobufs

SPDK v23.05 introduced a common pool of buffers to be used across libraries in SPDK called "iobuf". Over time more components are being converted to share the "iobufs". The default counts of elements are defined at values common for some use cases. Depending on the test scenario those values might need to be increased via "iobuf_set_options" RPC. Please see "./scripts/calc-iobuf.py" for guidance on minimum "iobuf" pool sizes.

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 and TCP. 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 for the purpose of this report document are tested against RoCEv2. 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 fourteen Kioxia® KCM61VUL3T20 SSDs which were used as block devices for NVMe-oF subsystems 100GbE Intel® E810-CQDA2 NICs connected to provide up to 200GbE of network bandwidth. Each Initiator system has two Intel® E810-CQDA2 100GbE NICs 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 (sysstat).

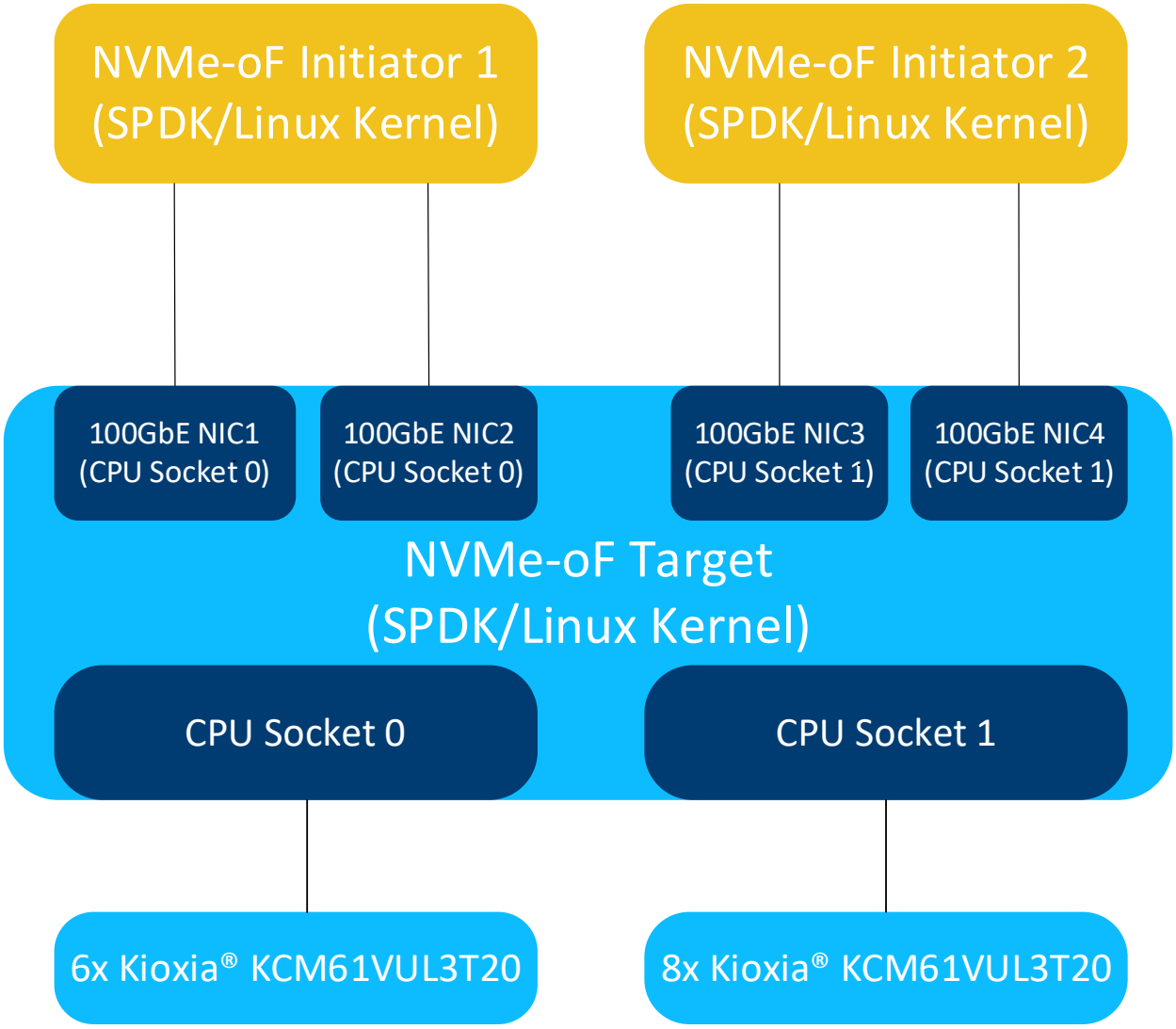

Figure 1: High-Level NVMe-oF RDMA performance testing setup

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

This test case was designed to demonstrate how the SPDK NVMe-oF target throughput in IOPS (I/O per second) scales when additional CPU cores are added to the SPDK NVMe-oF target application.

The SPDK NVMe-oF RDMA target was configured to run with 14 NVMe-oF subsystems. Each NVMe-oF subsystem ran on top of an individual NVMe bdev backed by a single Kioxia KCM61VUL3T20 NVMe drive. Each of the 2 host systems was connected to 7 NVMe-oF subsystems, which were exported by the SPDK NVMe-oF Target over 2 x 100GbE link. The SPDK bdev fio plugin was used to target 7 NVMe-oF bdevs on each of the host. SPDK Target Reactor Mask was configured to use up to 10 CPU cores tests while running following workloads on each initiator:

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

Below table contains information about the test configuration in form of a sequence of commands used by spdk/scripts/rpc.py script to configure the SPDK NVMe-oF Target. The SPDK NVMe-oF Initiator (bdev fio_plugin) still uses plain configuration files.

Each workload was run three times at each CPU count and the reported results are the average of the 3 runs. We preconditioned the SSDs once before running the 4KiB Rand Read and 4KiB Rand 70/30 Read/Write workloads to ensure that the SSDs reached their steady state where we get repeatable results. However, for the 4KiB Rand Write workload we didn't precondition the NVMe devices to ensure workload saturated the network rather than being limited to the steady state performance of the SSDs which is much lower than the available network bandwidth.

Table 5: SPDK NVMe-oF RDMA Target Core Scaling test configuration

Item	Description
Test Case	SPDK NVMe-oF Target I/O core scaling
SPDK NVMe-oF Target configuration	All the commands below were executed with spdk/scripts/rpc.py script. **Set iobuf buffer pool options:** iobuf_set_options --small-pool-count 32767 --large-pool-count 16383 **Construct NVMe bdevs:** bdev_nvme_attach_controller -t PCIe -b Nvme0 -a 0000:17:00.0 bdev_nvme_attach_controller -t PCIe -b Nvme1 -a 0000:18:00.0 bdev_nvme_attach_controller -t PCIe -b Nvme2 -a 0000:65:00.0 bdev_nvme_attach_controller -t PCIe -b Nvme3 -a 0000:66:00.0 bdev_nvme_attach_controller -t PCIe -b Nvme4 -a 0000:67:00.0 bdev_nvme_attach_controller -t PCIe -b Nvme5 -a 0000:68:00.0 bdev_nvme_attach_controller -t PCIe -b Nvme6 -a 0000:98:00.0 bdev_nvme_attach_controller -t PCIe -b Nvme7 -a 0000:99:00.0 bdev_nvme_attach_controller -t PCIe -b Nvme8 -a 0000:9a:00.0 bdev_nvme_attach_controller -t PCIe -b Nvme9 -a 0000:9b:00.0 bdev_nvme_attach_controller -t PCIe -b Nvme10 -a 0000:e3:00.0

	bdev_nvme_attach_controller -t PCIe -b Nvme11 -a 0000:e4:00.0
	bdev_nvme_attach_controller -t PCIe -b Nvme12 -a 0000:e5:00.0
	bdev_nvme_attach_controller -t PCIe -b Nvme13 -a 0000:e6:00.0
	
	Create RDMA transport layer:
	nvmf_create_transport -t RDMA -n 8192
	{
	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: 8192
	buf_cache_size: 32
	}
	
	Create NVMe-oF subsystems and add NVMe bdevs as namespaces:
	for i in $(seq 1 16); do
	nvmf_subsystem_create nqn.2018-09.io.spdk:cnode${i} -s SPDK00${i} -a -m 8
	nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode${i} Nvme$((i-1))n1
	done
	
	Add listeners to NVMe-oF Subsystems:
	i=1
	ips=(20.0.0.1 20.0.1.1 10.0.0.1 10.0.1.1)
	for ip in ${ips[@]}; do
	for j in $(seq 1 4); do
	nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode${i} -t rdma \
	-f ipv4 -s 4420 -a ${ip}
	((i++))
	done
	done

SPDK NVMe-oF Initiator - configuration and fio plugin configuration	**BDEV.conf** See Appendix A **fio.conf** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_json_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

4KiB Random Read Results

Table 6: SPDK NVMe-oF RDMA Target Core Scaling results, Random Read IOPS, QD=128

# of Cores	Bandwidth (MiBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	6341.12	1623.3	1103.7
2 cores	14927.70	3821.5	468.5
3 cores	23770.35	6085.2	300.5
4 cores	32682.16	8366.6	214.9
5 cores	39189.53	10032.5	178.2
6 cores	41539.68	10634.2	168.1
8 cores	42587.65	10902.4	163.9
10 cores	42710.23	10933.8	163.5

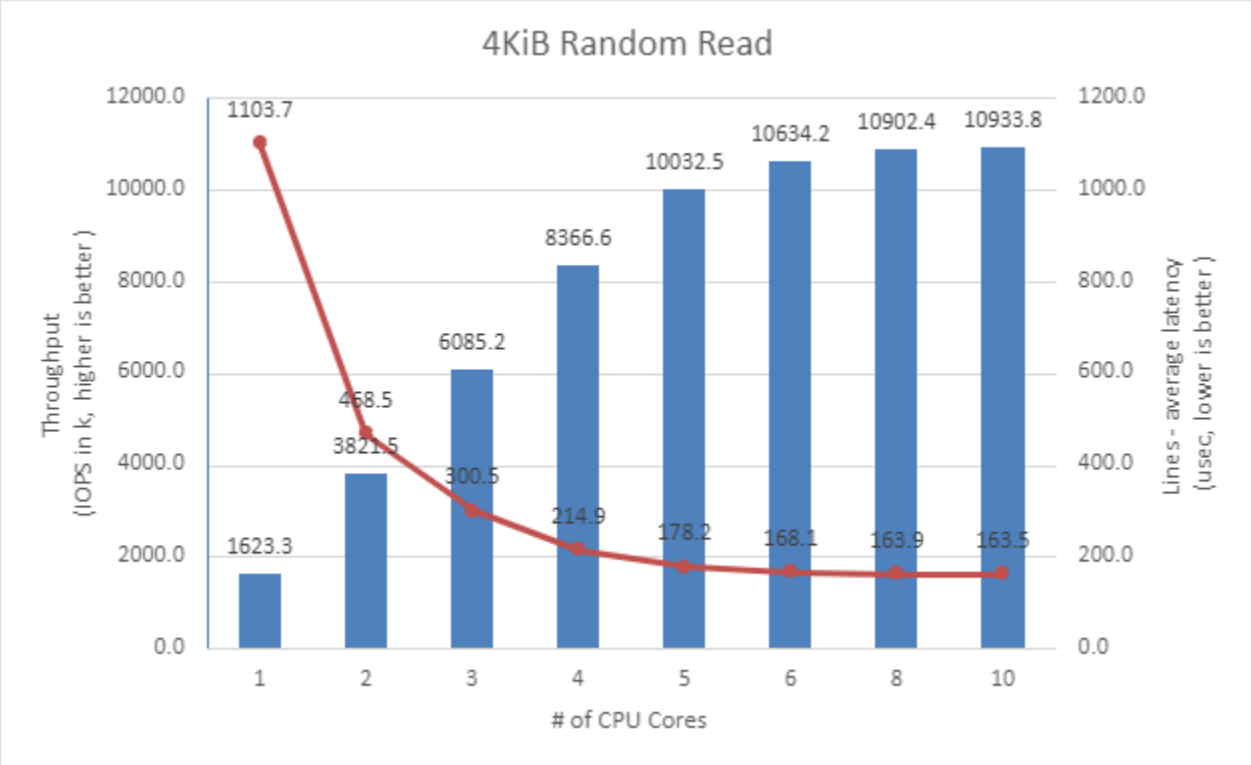

Figure 2: SPDK NVMe-oF RDMA Target I/O core scaling: IOPS vs. Latency while running 4KiB 100% Random Read workload at QD = 128

4KiB Random Write Results

Table 7: SPDK NVMe-oF RDMA Target Core Scaling results, Random Write IOPS, QD=64

# of Cores	Bandwidth (MiBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	9057.09	2318.6	385.1
2 cores	20390.27	5219.9	169.6
3 cores	32091.05	8215.3	106.6
4 cores	38385.38	9826.7	89.3
5 cores	40554.99	10382.1	85.1
6 cores	40715.16	10423.1	85.0
8 cores	40753.59	10432.9	85.2
10 cores	41004.69	10497.2	84.8

Note that the SSDs were not preconditioned for the 4KiB random write workload because that would limit the workload performance to the SSDs steady state performance.

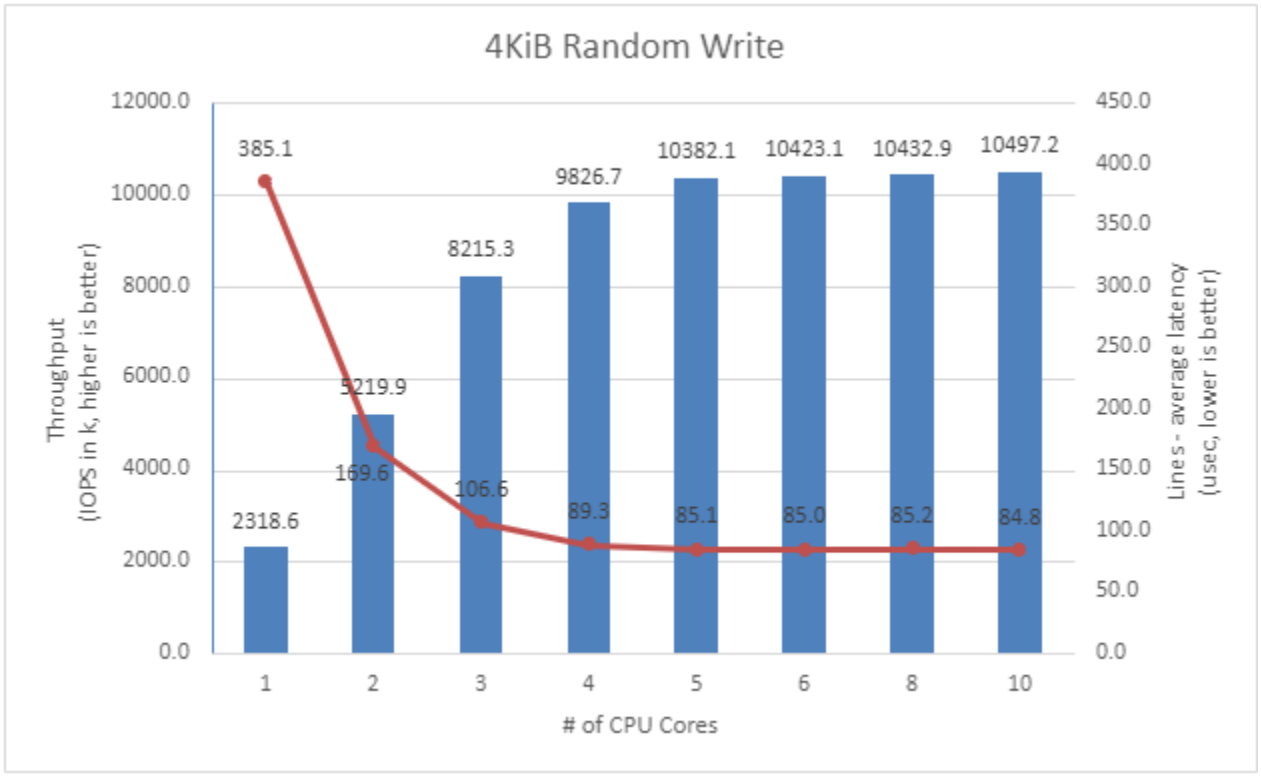

Figure 3: SPDK NVMe-oF RDMA Target I/O core scaling: IOPS vs. Latency while running 4KiB 100% Random Write Workload at QD=64

4KiB Random Read-Write Results

Table 8: SPDK NVMe-oF RDMA Target Core Scaling results, Random Read/Write 70%/30% IOPS, QD=128

# of Cores	Bandwidth (MiBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	6561.82	1679.8	1066.5
2 cores	15309.46	3919.2	456.3
3 cores	24341.62	6231.4	286.5
4 cores	32235.84	8252.4	216.2
5 cores	38477.38	9850.2	181.1
6 cores	41919.75	10731.4	166.0
8 cores	45366.20	11613.7	153.5
10 cores	45842.64	11735.7	152.0

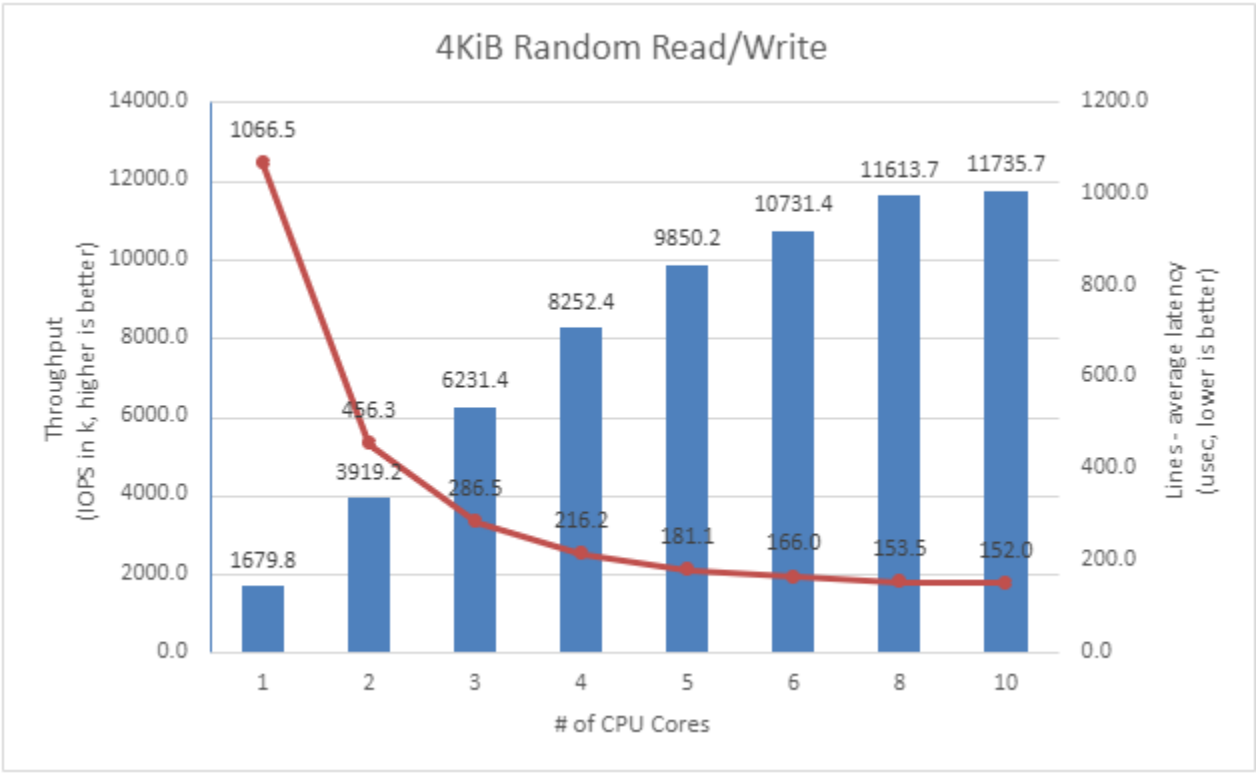

Figure 4: SPDK NVMe-oF RDMA Target I/O core scaling: IOPS vs. Latency while running 4KiB Random 70/30 Read/Write workload at QD=128

Large Sequential I/O Performance

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

Table 9: SPDK NVMe-oF RDMA Target Core Scaling results, 128KiB Sequential Read IOPS, QD=4

# of Cores	Bandwidth (MiBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	42300.12	338.4	165.3
2 cores	42491.07	339.9	164.6
3 cores	42499.29	340.0	164.6
4 cores	42522.54	340.2	164.5

Table 10: SPDK NVMe-oF RDMA Target Core Scaling results, 128KiB Sequential Write IOPS, QD=4

# of Cores	Bandwidth (MiBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	44010.64	352.1	158.9
2 cores	44035.20	352.3	158.8
3 cores	44040.99	352.3	158.8
4 cores	43989.16	351.9	159.0

Table 11: SPDK NVMe-oF RDMA Target Core Scaling results, 128KiB Sequential 70% Read 30% Write IOPS, QD=8

# of Cores	Bandwidth (MiBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	47809.33	382.5	292.5
2 cores	47986.48	383.9	291.4
3 cores	47994.32	384.0	291.4
4 cores	48010.93	384.1	291.3

Conclusions

1. 100% 4KiB Random Read workload throughput scales up and latency decreases linearly with the addition of I/O cores up to 5 cores, reaching network saturation.

2. 100% 4KiB Random Write workload throughput scales up and latency decreases linearly with the addition of I/O cores up to 4 cores reaching 9.8 million IOPS. Further increasing the number of CPU cores results in non-linear or no performance improvement. Peak performance is reached with 10 CPU cores and 10.5 million IOPS.

3. 70/30% 4KiB Random Read/Write throughput scales up and latency decreases linearly with the addition of I/O cores up to 5 cores reaching 9.85 million IOPS. Further increasing the number of CPU cores results in non-linear or no performance improvement. Peak performance is reached with 8 CPU cores and 11.6 million IOPS, saturating network link.

4. For large sequential I/Os, a single CPU core Target saturated the network bandwidth. Therefore, adding more CPU cores did not result in increased performance for these workloads because the network was saturated.

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

This test case was designed to demonstrate how the SPDK NVMe-oF initiator throughput in IOPS (I/O per second) scales when additional CPU cores are added to the SPDK NVMe-oF initiator.

The test setup for this test case is slightly different than the set up described in introduction chapter, as we used just a single SPDK NVMe-oF RDMA Initiator. The Initiator was connected to Target server with two 100 Gbps network links, using single port from two separate network interface cards.

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 14 individual NVMe-oF subsystems exported by the Target. The number of CPU threads used by the fio process was managed by setting the fio job sections and numjobs parameter and ranged from 1 to 8 CPUs. For detailed fio job configuration see table below.

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

It is important to note that fio io depth parameter values presented in the table below are actual queue depths used for each of the connected subsystem. These values were calculated in test based on number of fio job sections, numjobs parameter and the number of "filename" targets grouped in each of the fio job sections.

Table 12: SPDK NVMe-oF RDMA Initiator Core Scaling test configuration

Item	Description
Test Case	SPDK NVMe-oF RDMA Initiator I/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** See appendix B. **fio.conf** **For 1 CPU initiator configuration:** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_conf=/tmp/spdk/bdev.conf thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={1,32, 64, 128, 192, 256} time_based=1

	ramp_time=60 runtime=300 numjobs=1 [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 filename=Nvme4n1 filename=Nvme5n1 filename=Nvme6n1 filename=Nvme7n1 filename=Nvme8n1 filename=Nvme9n1 filename=Nvme10n1 filename=Nvme11n1 filename=Nvme12n1 filename=Nvme13n1
	fio.conf **For CPU > 1 (up to N=8) initiator configuration "filename=NvmeXn1" are evenly spread across fio job threads:** [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={1,32, 64, 128, 192, 256} time_based=1 ramp_time=60 runtime=300 numjobs=X [filename0] filename=Nvme0n1 filename=Nvme1n1 [filename1] filename=Nvme2n1 filename=Nvme3n1 […] [filename N-1] filename=Nvme10n1 filename=Nvme11n1 [filename N] filename=Nvme12n1 filename=Nvme13n1

4KiB Random Read Results

Table 13: SPDK NVMe-oF RDMA Initiator Core Scaling results, 4KiB Random Read IOPS, QD=64, SPDK Target 6 CPU Cores

# of Initiator CPU Cores	Bandwidth (MiBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	6107.25	1563.5	472.3
2 cores	13752.74	3520.7	220.0
3 cores	19448.92	4978.9	165.8
4 cores	21348.97	5465.3	160.9
5 cores	21305.38	5454.2	162.8
6 cores	21285.25	5449.0	163.9
7 cores	21284.99	5449.0	164.0
8 cores	21288.48	5449.8	164.0

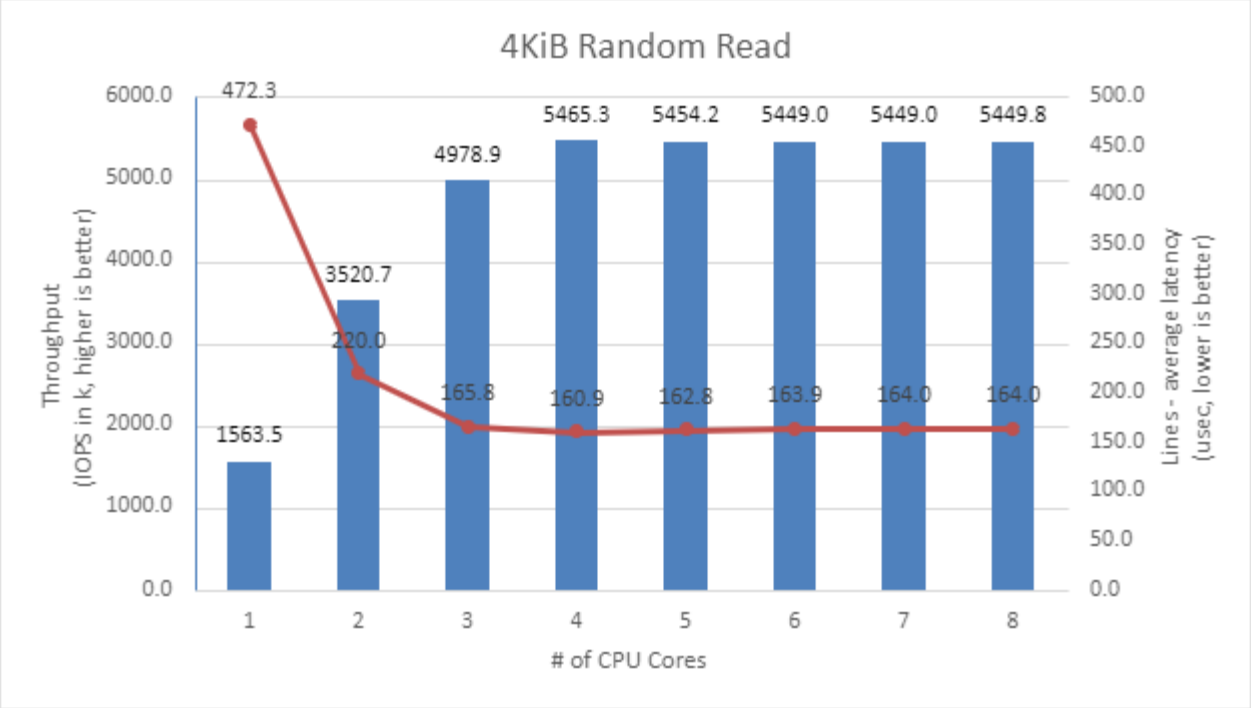

Figure 5: SPDK NVMe-oF RDMA Initiator I/O core scaling: IOPS vs. Latency while running 4KiB 100% Random Read QD=64 workload

4KiB Random Write Results

Note: The SSDs were not pre-conditioned before running 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).

Table 14: SPDK NVMe-oF RDMA Initiator Core Scaling results, 4KiB Random Write IOPS, QD=64, SPDK Target 6 CPU Cores

# of Initiator CPU Cores	Bandwidth (MiBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5722.43	1464.9	498.0
2 cores	12078.74	3092.2	239.1
3 cores	15981.37	4091.2	196.2
4 cores	18176.71	4653.2	189.5
5 cores	18880.17	4833.3	186.6
6 cores	20363.14	5213.0	171.3
7 cores	20947.41	5362.5	166.5
8 cores	21150.55	5414.5	165.0

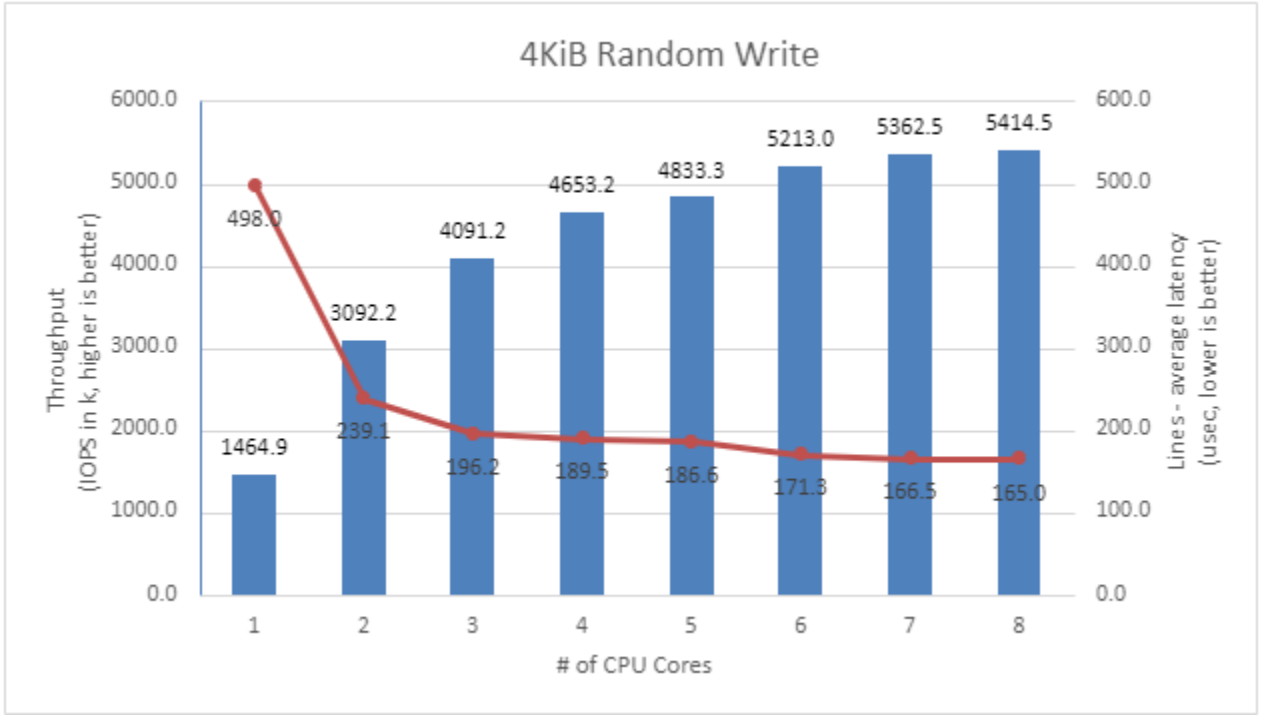

Figure 6: SPDK NVMe-oF RDMA Initiator I/O core scaling: IOPS vs. Latency while running 4KiB 100% Random Write Workload at QD=64

4KiB Random 70/30 Read/Write Results

Table 15: SPDK NVMe-oF RDMA Initiator Core Scaling results, 4KiB Random 70%/30% Read/Write IOPS, QD=64, SPDK Target 6 CPU Cores

# of Initiator CPU Cores	Bandwidth (MiBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5890.97	1508.1	481.0
2 cores	13437.92	3440.1	220.0
3 cores	18208.76	4661.4	168.6
4 cores	23328.69	5972.1	141.8
5 cores	26181.53	6702.5	129.7
6 cores	27186.34	6959.7	126.7
7 cores	27274.86	6982.4	127.5
8 cores	28424.65	7276.7	122.2

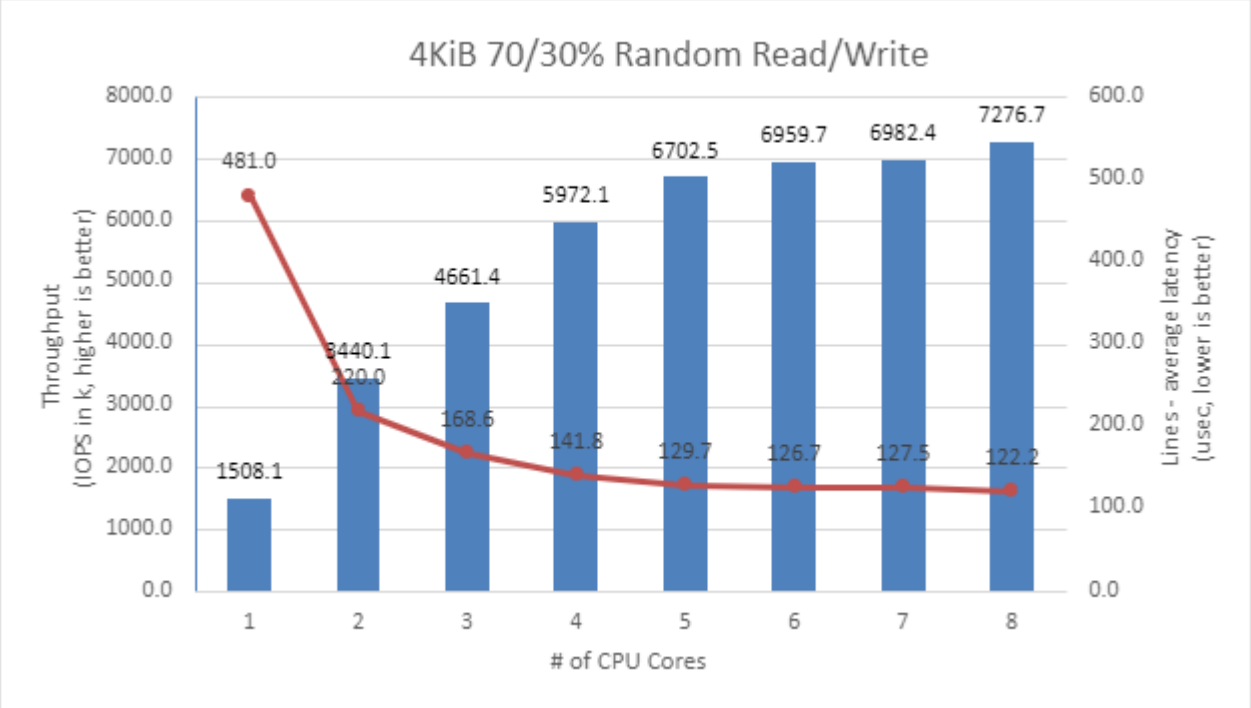

Figure 7: SPDK NVMe-oF RDMA Initiator I/O core scaling: IOPS vs. Latency while running 4KiB Random 70% Read 30% Write Workload at QD=64

Conclusions

1. Random Read workload scaling was not linear when increasing the number of initiator CPU cores. Peak performance of about 5.4 million IOPS was reached with SPDK NVMe-oF RDMA Initiator using 4 CPU cores, saturating 200 GbE link between Target and Initiator.

2. Random Write workload scaling was not linear when increasing the number of initiator CPU cores. Peak performance of 5.4 million IOPS was reached with SPDK NVMe-oF RDMA Initiator using 8 CPU cores, saturating Target NVMe drives 4KiB Random Write processing capability.

3. Random Read-Write scaling was not linear when increasing the number of initiator CPU cores. Performance peaked at 7.27 million IOPS with the SPDK NVMe-oF RDMA initiator using 8 CPU cores. Throughput exceeding 200 GbE link between Target and Initiator can be explained by the fact that mixed read-write workload can be considered a bi-directional network traffic, not being strictly affected by 200 GbE “one-way” network limit.

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 backed by a Null Block Device. The null block device (bdev) was chosen as the backend block device to eliminate the media latency during these tests.

Kernel NVMe-oF Initiator disclaimer:

For establishing Kernel NVMe-oF RDMA Initiator connections "nvme-cli" tool was used. While performing benchmark tests two issues were encountered:

- It was not possible to establish connection and create a NVMe block device on Initiator side with poll queues enabled (link). Using "nvme-cli" with "--nr-poll-queues" parameter present resulted in "Kernel Oops" to be generated. Because of this issue the fio workload for Kernel Initiator connection was configured to use "libaio" engine.
- Attempts to establish connection with default number of IO queues (which is equal to number of CPU cores on Initiator system) resulted in connection timeouts. To work around this issue "--nr-io-queues=32" was added to nvme-cli command. This does not affect the results in this test case as only a single connection with very small queue depth is tested.

Table 16: Linux Kernel vs. SPDK NVMe-oF RDMA Latency test configuration

Item	Description
Test Case	Linux Kernel vs. SPDK NVMe-oF RDMA Latency
	Test configuration
SPDK NVMe-oF Target configuration	The following commands are executed with spdk/scripts/rpc.py script to configure the SPDK NVMe-oF target. **Set iobuf buffer pool options:** iobuf_set_options --small-pool-count 32767 --large-pool-count 16383 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: 8192 buf_cache_size: 32) bdev_null_create 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	The following 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** See Appendix B. **fio.conf** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_json_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** The following configuration was 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 **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 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.

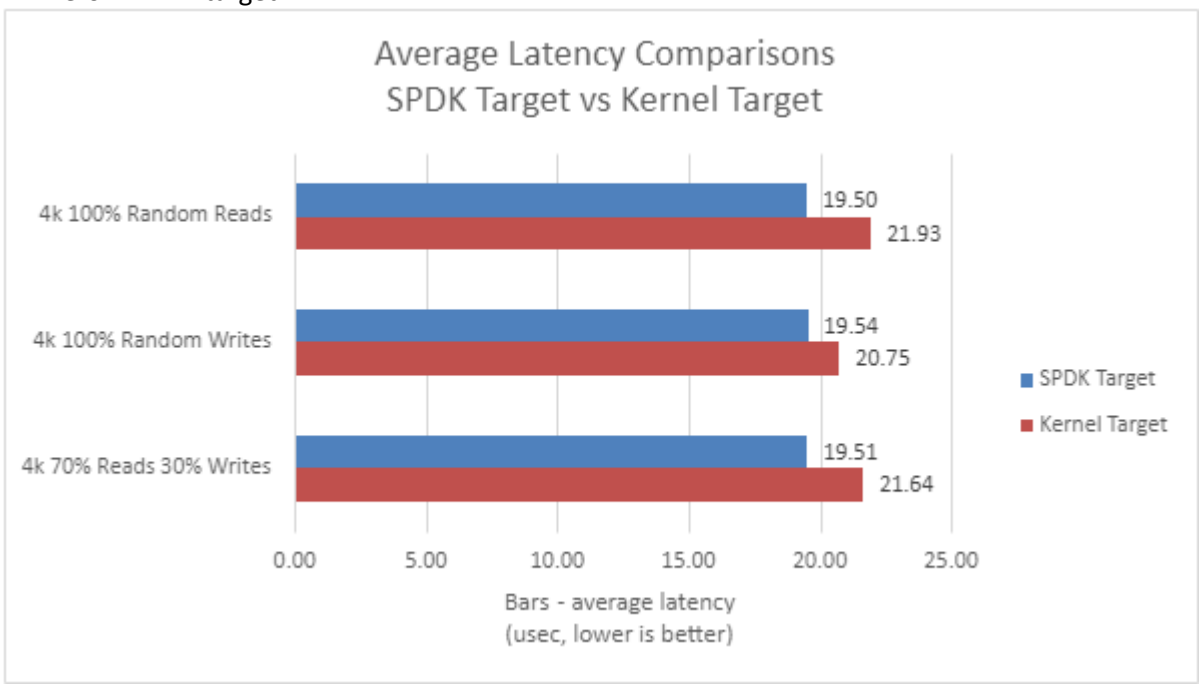

Figure 8: SPDK vs. Kernel NVMe-oF RDMA Target average I/O latency for various workloads run using the Kernel Initiator

Table 17: SPDK NVMe-oF RDMA Target Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KiB 100% Random Read	19.50	50606	19.8	23.6	129.9	381.6
4KiB 100% Random Write	19.54	50551	19.5	60.2	196.3	384.3
4KiB 70/30% Random Read/Write	19.51	50509	20.1	25.5	148.1	388.3

Table 18: Linux Kernel NVMe-oF RDMA Target Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KiB 100% Random Read	21.93	45069	24.6	27.3	146.9	257.0
4KiB 100% Random Write	20.75	47642	21.7	58.5	153.5	252.9
4KiB 70/30% Random Read/Write	21.64	45597	22.5	77.1	126.0	256.1

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.

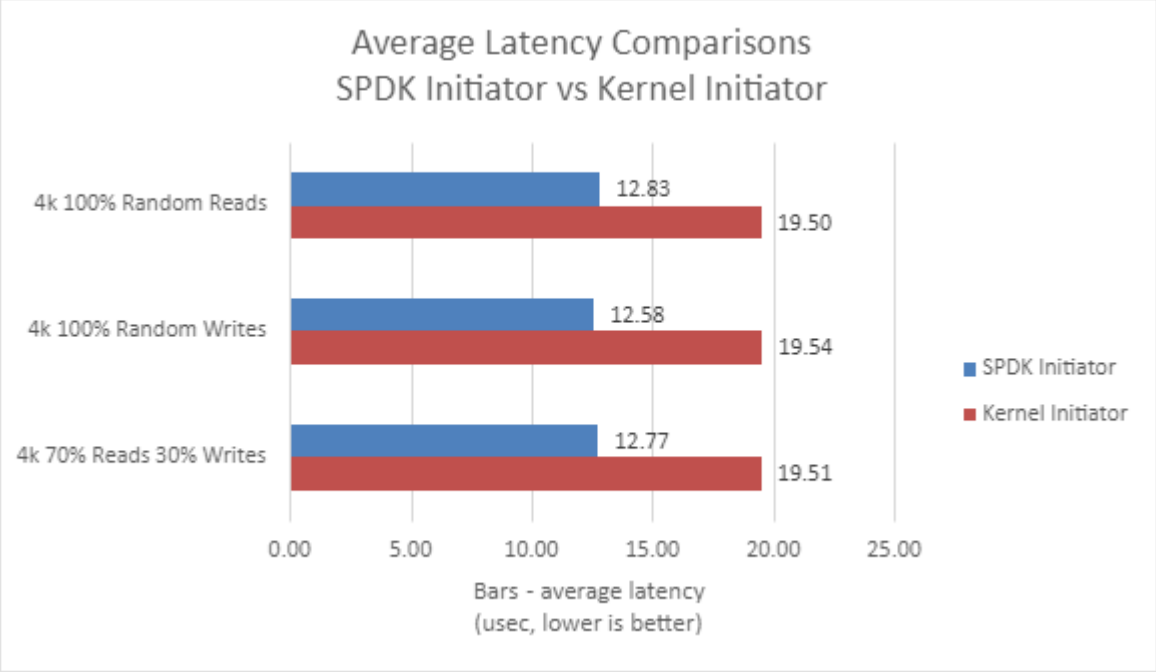

Figure 9: SPDK vs. Kernel NVMe-oF RDMA Initiator average I/O latency for various workloads against SPDK Target

Table 19: SPDK NVMe-oF RDMA Initiator Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KiB 100% Random Read	12.83	76953	15.1	18.9	41.4	328.4
4KiB 100% Random Write	12.58	78477	14.3	18.4	40.0	332.8
4KiB 70/30% Random Read/Write	12.77	77246	14.5	20.5	43.2	362.2

Table 20: Linux Kernel NVMe-oF RDMA Initiator Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KiB 100% Random Read	19.50	50606	19.8	23.6	129.9	381.6
4KiB 100% Random Write	19.54	50551	19.5	60.2	196.3	384.3
4KiB 70/30% Random Read/Write	19.51	50509	20.1	25.5	148.1	388.3

SPDK vs Kernel NVMe-oF RDMA Kernel + Initiator Results

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

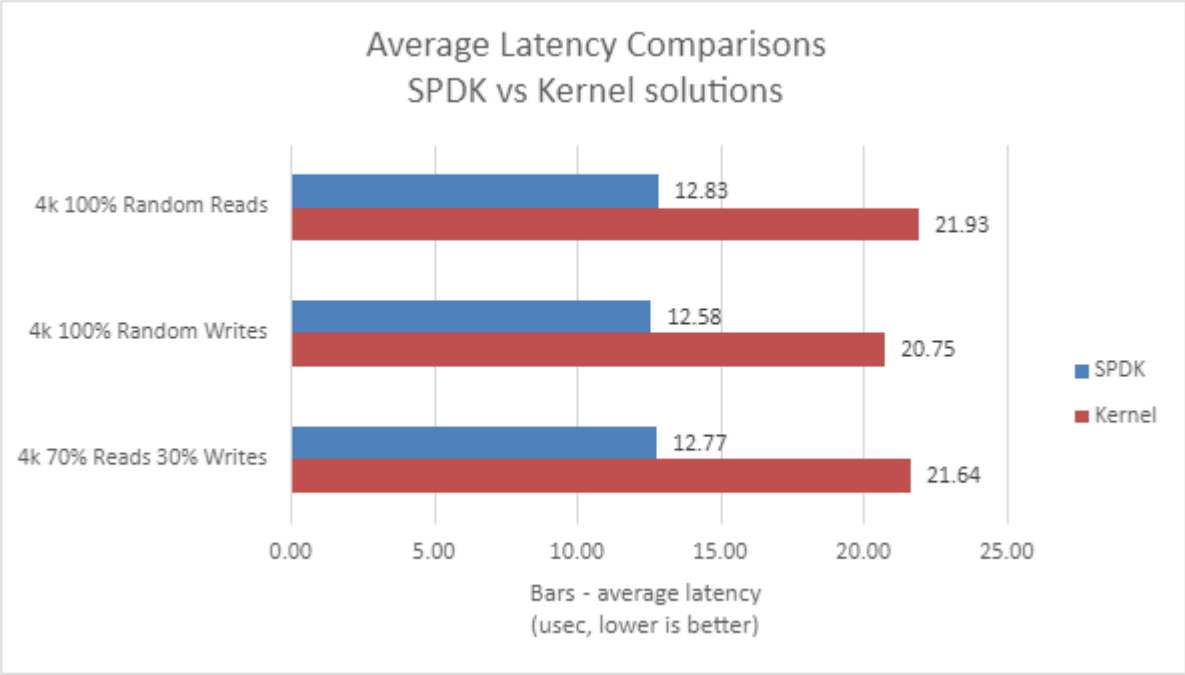

Figure 10: SPDK vs. Kernel NVMe-oF RDMA solutions average I/O Latency for various workloads against SPDK Target

Table 21: SPDK NVMe-oF RDMA Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KiB 100% Random Read	12.83	76953	15.1	18.9	41.4	328.4
4KiB 100% Random Write	12.58	78477	14.3	18.4	40.0	332.8
4KiB 70/30% Random Read/Write	12.77	77246	14.5	20.5	43.2	362.2

Table 22: Linux Kernel NVMe-oF RDMA Latency and IOPS at QD=1, Null Block Device

Access Pattern	Average Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KiB 100% Random Read	21.93	45069	24.6	27.3	146.9	257.0
4KiB 100% Random Write	20.75	47642	21.7	58.5	153.5	252.9
4KiB 70/30% Random Read/Write	21.64	45597	22.5	77.1	126.0	256.1

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 2.43 usec vs. the Linux Kernel NVMe-oF target used in Fedora 37 6.0.18 setup. This is entirely software overhead, therefore, using the SPDK NVMe-oF target reduces the NVMe-oF software overhead by approximately 11.08% vs. the Linux Kernel NVMe-oF target.

2. The SPDK NVMe-oF Initiator reduces the NVMe-oF software overhead by up to 6.96 usec vs. the Linux Kernel NVMe-oF Initiator for the RDMA transport, which is approximately 35.6% of Linux Kernel NVMe-oF Initiator overhead.

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

This test case was designed to demonstrate the throughput and latency of the SPDK NVMe-oF RDMA Target vs. Linux Kernel NVMe-oF RDMA Target under increasing number of connections per subsystem. The number of active connections (or I/O queue pairs) per NVMe-oF subsystem was varied, we measured the aggregated IOPS and number of CPU cores used by each target. The number of CPU cores metric was calculated from %CPU utilization measured using sar (sysstat package in Linux). The SPDK NVMe-oF RDMA Target was configured to run on 8 CPU cores, export 14 NVMe-oF subsystems (1 per Kioxia NVMe SSD) and 2 initiators were used both running the I/O workloads below to 7 separate subsystems using Kernel NVMe-oF RDMA initiator.

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

Kernel NVMe-oF Initiator disclaimer:

For establishing Kernel NVMe-oF RDMA Initiator connections “nvme-cli” tool was used. While performing benchmark tests two issues were encountered:

- It was not possible to establish connection and create a NVMe block device on Initiator side with poll queues enabled (link). Using “nvme-cli” with “--nr-poll-queues” parameter present resulted in “Kernel Oops” to be generated. Because of this issue the fio workload for Kernel Initiator connection was configured to use “libaio” engine.
- Attempts to establish connection with default number of IO queues (which is equal to number of CPU cores on Initiator system) resulted in connection timeouts. To work around this issue “--nr-io-queues=32” was added to nvme-cli command. This does not affect the results in this test case as connections are not scaled beyond 16 per NVMe-oF subsystem, thus maximum number of used IO queues per NVMe-oF subsystem is 16.

Table 23: NVMe-oF RDMA Performance with increasing number of connections test configuration

Item	Description
Test Case	NVMe-oF RDMA Target performance with increasing # of connections
SPDK NVMe-oF Target configuration	Same as in Test Case #1, using 8 CPU cores.
Kernel NVMe-oF Target configuration	Target configuration file loaded using nvmet-cli tool. For detailed configuration file contents please see Appendix C.
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
Kernel NVMe-oF Initiator #2	**Device config** Performed using nvme-cli tool. modprobe nvme-fabrics nvme connect –n nqn.2018-09.io.spdk:cnode8 –t rdma –a 10.0.0.1 –s 4420 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 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode13 –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={32, 64, 128, 192} time_based=1 ramp_time=60 runtime=300 numjobs={1, 4, 8, 12, 16} [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

The SPDK NVMe-oF Target was configured to use 8 CPU cores for Random Read and Random Read/Write workloads and 4 CPU cores for Random Write workloads. We did not limit the number of CPU cores for the Linux Kernel NVMe-oF target. The graph below shows the relative efficiency 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.

4KiB Random Read Results

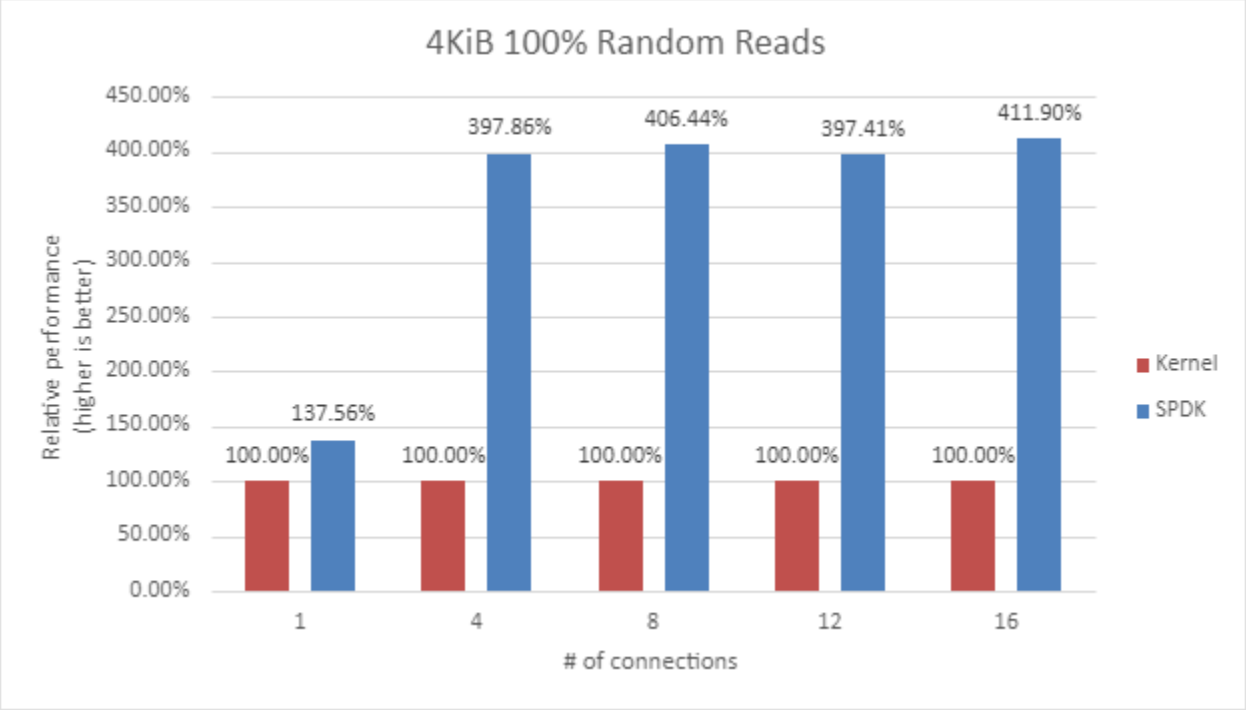

Figure 11: Relative Efficiency Comparison of Linux Kernel vs. SPDK NVMe-oF RDMA Target IOPS/Core for 4KiB 100% Random Reads QD=192, using the Kernel Initiator

Table 24: Linux Kernel NVMe-oF RDMA Target: 4KiB 100% Random Reads, QD=192

Connections per subsystem	Bandwidth (MiBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	19368.64	4958.4	544.5	12.0
4	39488.62	10109.1	265.8	29.8
8	38810.24	9935.4	270.4	33.5
12	38686.72	9903.8	271.1	32.9
16	38810.29	9935.4	270.2	34.6

Table 25: SPDK NVMe-oF RDMA Target: 4KiB 100% Random Reads, QD=192

Connections per subsystem	Bandwidth (MiBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	19922.06	5100.0	531.7	9.0
4	42621.46	10911.1	246.1	8.1
8	42606.39	10907.2	246.2	9.0
12	42284.51	10824.8	248.0	9.0
16	41788.55	10697.9	250.9	9.0

4KiB Random Write results

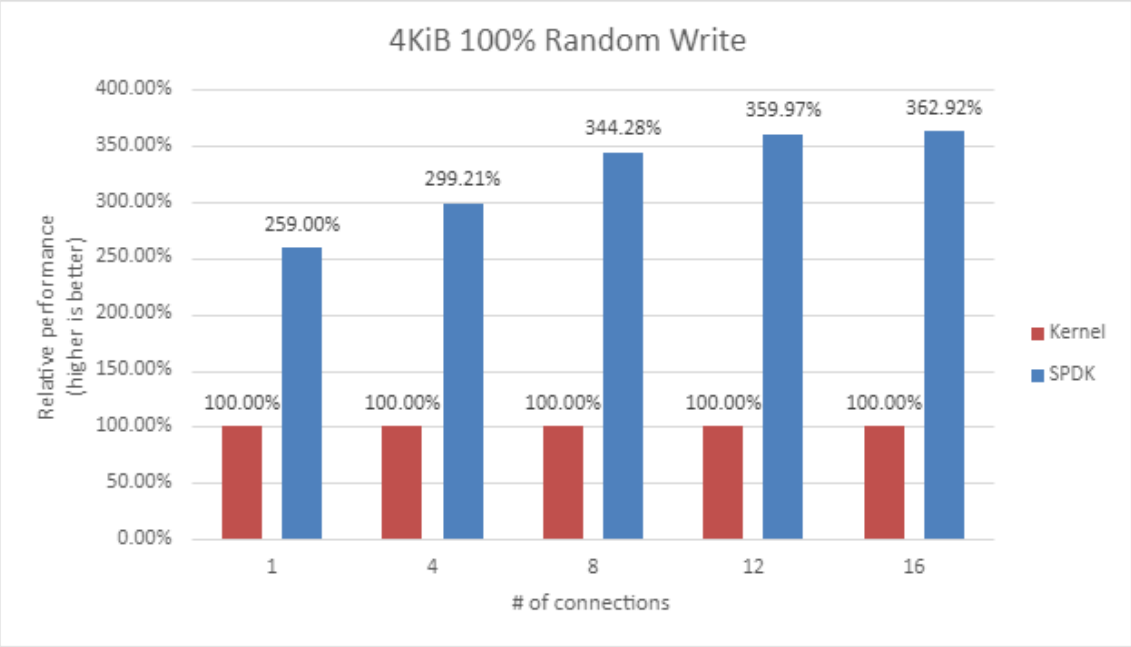

Figure 12: Relative Efficiency Comparison of Linux Kernel vs. SPDK NVMe-oF RDMA Target IOPS/Core for 4KiB 100% Random Writes QD=128 workload, using Kernel Initiators

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

Table 26: Linux Kernel NVMe-oF RDMA Target: 4KiB 100% Random Writes, QD=128

Connections per subsystem	Bandwidth (MiBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	20462.11	5238.3	347.9	10.2
4	26256.98	6721.8	273.1	15.1
8	27421.11	7019.8	263.0	18.1
12	28954.23	7412.3	257.8	19.6
16	29656.84	7592.1	244.2	20.6

Table 27: SPDK NVMe-oF RDMA Target: 4KiB 100% Random Writes, QD=128

Connections per subsystem	Bandwidth (MiBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	21257.29	5441.9	334.4	4.1
4	26402.13	6758.9	271.8	5.1
8	26442.07	6769.2	269.8	5.1
12	26871.37	6879.1	272.4	5.0
16	26298.55	6732.4	269.1	5.0

4KiB Random Read-Write Results

Figure 13: Relative Efficiency Comparison of Linux Kernel vs. SPDK NVMe-oF RDMA Target IOPS/Core for 4KiB Random 70% Reads 30% Writes QD=192 Workload, using Kernel Initiators

Table 28: Linux Kernel NVMe-oF RDMA Target: 4KiB 70% Random Read 30% Random Write, QD=192

Connections per subsystem	Bandwidth (MiBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	19624.74	5023.9	537.2	11.7
4	40630.22	10401.3	258.3	29.3
8	40393.91	10340.8	259.7	32.0
12	40299.24	10316.6	260.2	32.9
16	40068.57	10257.5	261.6	34.5

Table 29: SPDK NVMe-oF RDMA Target: 4KiB 70% Random Read 30% Random Write, QD=192

Connections per subsystem	Bandwidth (MiBps)	IOPS (k)	Avg. Latency (usec)	# CPU Cores
1	20192.51	5169.3	524.3	9.0
4	41108.04	10523.6	255.2	9.0
8	40213.34	10294.6	260.8	9.0
12	40262.93	10307.3	260.4	8.1
16	39955.52	10228.6	262.4	9.1

Conclusions

1. SPDK NVMe-oF Target performance peaked at 4 connections for all workloads. Increasing the number of connections per subsystem beyond these values did not result in significant changes to the IOPS or latency.

2. The performance for the Linux Kernel NVMe-oF Target peaked at 4 connections per subsystem for all Random Read and 70/30 Random Read/Write workloads. Increasing the number of connections for these workloads increases the latency and CPU utilization and lowers IOPS. For Random Write workload performance peaked at 16 connections per subsystem.

3. The SPDK NVMe-oF target shows up to 4.12x 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 previous reports, throughput scales up and latency decreases almost linearly with the scaling of SPDK NVMe-oF Target cores.

It was also observed that the SPDK NVMe-oF Target average latency is up to 2.43 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 lower by up to 35.6% lower than Kernel initiator.

The SPDK NVMe-oF Target performed up to 4.12 times better w.r.t IOPS/core than Linux Kernel NVMe-oF target while running 4KiB 100% Random workloads with increasing number of active 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.

List of tables

List of figures

Appendix A – Test Case 1 & 2 SPDK NVMe-oF Initiator bdev configuration

Initiator system 1

```
{
  "subsystems": [
    {
      "subsystem": "bdev",
      "config": [
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme0",
            “trtype”: “rdma”,
            "traddr": "20.0.0.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode0",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme1",
            “trtype”: “rdma”,
            "traddr": "20.0.0.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode1",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme2",
            “trtype”: “rdma”,
            "traddr": "20.0.0.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode2",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme3",
            “trtype”: “rdma”,
            "traddr": "20.0.0.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode3",
            "adrfam": "IPv4"
          }
```

```
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme4",
            “trtype”: “rdma”,
            "traddr": "20.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode4",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme5",
            “trtype”: “rdma”,
            "traddr": "20.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode5",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme6",
            “trtype”: “rdma”,
            "traddr": "20.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode6",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme7",
            “trtype”: “rdma”,
            "traddr": "20.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode7",
            "adrfam": "IPv4"
          }
        }
      ]
    }
  ]
```

Initiator system 2

```
{
  "subsystems": [
    {
      "subsystem": "bdev",
      "config": [
        {
```

```
  "method": "bdev_nvme_attach_controller",
  "params": {
    "name": "Nvme0",
    “trtype”: “rdma”,
    "traddr": "10.0.0.1",
    "trsvcid": "4420",
    "subnqn": "nqn.2018-09.io.spdk:cnode0",
    "adrfam": "IPv4"
  }
},
{
  "method": "bdev_nvme_attach_controller",
  "params": {
    "name": "Nvme1",
    “trtype”: “rdma”,
    "traddr": "10.0.0.1",
    "trsvcid": "4420",
    "subnqn": "nqn.2018-09.io.spdk:cnode1",
    "adrfam": "IPv4"
  }
},
{
  "method": "bdev_nvme_attach_controller",
  "params": {
    "name": "Nvme2",
    “trtype”: “rdma”,
    "traddr": "10.0.0.1",
    "trsvcid": "4420",
    "subnqn": "nqn.2018-09.io.spdk:cnode2",
    "adrfam": "IPv4"
  }
},
{
  "method": "bdev_nvme_attach_controller",
  "params": {
    "name": "Nvme3",
    “trtype”: “rdma”,
    "traddr": "10.0.0.1",
    "trsvcid": "4420",
    "subnqn": "nqn.2018-09.io.spdk:cnode3",
    "adrfam": "IPv4"
  }
},
{
  "method": "bdev_nvme_attach_controller",
  "params": {
    "name": "Nvme4",
    “trtype”: “rdma”,
    "traddr": "10.0.1.1",
    "trsvcid": "4420",
    "subnqn": "nqn.2018-09.io.spdk:cnode4",
    "adrfam": "IPv4"
  }
},
{
  "method": "bdev_nvme_attach_controller",
  "params": {
```

```
            "name": "Nvme5",
            “trtype”: “rdma”,
            "traddr": "10.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode5",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme6",
            “trtype”: “rdma”,
            "traddr": "10.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode6",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme7",
            “trtype”: “rdma”,
            "traddr": "10.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode7",
            "adrfam": "IPv4"
          }
        }
      ]
    }
  ]
}
```

Appendix B – Test Case 3 SPDK NVMe-oF Initiator bdev configuration

```
{
  "subsystems": [
    {
      "subsystem": "bdev",
      "config": [
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme0",
            "trtype": "tcp",
            "traddr": "20.0.0.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode0",
            "adrfam": "IPv4"
          }
        }
      ]
    }
  ]
}
```

Appendix C - Kernel NVMe-oF RDMA Target configuration

Example Linux 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 & Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates.

Your costs and results may vary.

No product or component can be absolutely secure.

Intel technologies may require enabled hardware, software or service activation.

§