SPDK NVMe-oF TCP (Target & Initiator) Performance Report Release 21.01

Testing Date: February 2021

Performed by:

Karol Latecki (karol.latecki@intel.com)

Maciej Wawryk (maciejx.wawryk@intel.com)

Acknowledgments:

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

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

Contents

Audience and Purpose

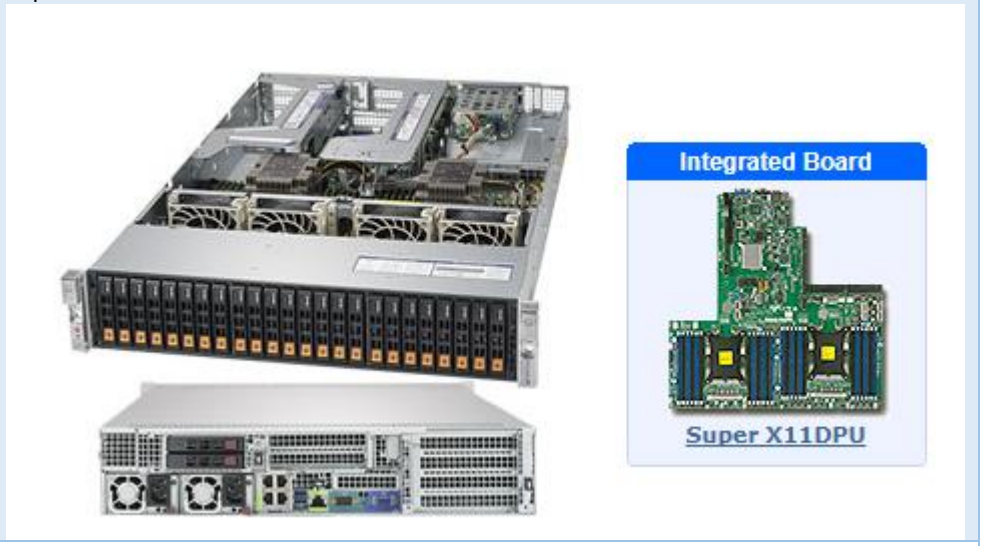
This report is intended for people who are interested in evaluating SPDK NVMe-oF (Target & Initiator) performance. This report contains SPDK NVMe-oF Target and Initiator performance characteristics and provides comparison data between SPDK and its Kernel NVMe-oF Target and Initiator counterparts. This report covers the TCP transport only.

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

Test setup

Target Configuration

Table 1: Hardware setup configuration – Target system

Item	Description
Server Platform	SuperMicro SYS-2029U-TN24R4T 
CPU	Intel® Xeon® Gold 6230 Processor (27.5MB L3, 2.10 GHz) Number of cores 20 per socket, number of threads 40 per socket (both sockets populated) Microcode: 0x5003003
Memory	12 x 32GB Hynix HMA84GR7AFR4N-VK, DDR4, 2666MHz Total of 384GB
Operating System	Fedora 33
BIOS	3.4
Linux kernel version	5.8.15-300.fc33.x86_64
SPDK version	SPDK 21.01
Storage	**OS:** 1x 120GB Intel SSDSC2BB120G4 **Storage Target**: 16x Intel® SSD DC P4610™ 1.6TB (FW: VDV10170) (8 on each CPU socket)
NIC	2x 100GbE Mellanox ConnectX-5 NICs. Both ports connected. 1 NIC per CPU socket.

Initiator 1 Configuration

Table 2: Hardware setup configuration – Initiator system 1

Item	Description
Server Platform	Intel® Server System R2208WFTZSR
CPU	Intel(R) Xeon(R) Gold 6252 CPU @ 2.10GHz (35.75MB Cache) Number of cores 24 per socket, number of threads 48 per socket (Both sockets populated) Microcode: 0x5003003
Memory	6 x 32GB Micron M393A1G40EB1-CRC, DDR4, 2933MHz Total 192GBs
Operating System	Fedora 33
BIOS	02.01.0012
Linux kernel version	5.8.15-300.fc33.x86_64
SPDK version	SPDK 21.01
Storage	**OS:** 1x 240GB INTEL SSDSC2BB240G6
NIC	1x 100GbE Mellanox ConnectX-5 Ex NIC. Both ports connected to Target server. (connected to CPU socket 0)

Initiator 2 Configuration

Table 3: Hardware setup configuration – Initiator system 2

Item	Description
Server Platform	Intel® Server System R2208WFTZSR
CPU	Intel(R) Xeon(R) Gold 6252 CPU @ 2.10GHz (35.75MB Cache) Number of cores 24 per socket, number of threads 48 per socket (Both sockets populated) Microcode: 0x5003003
Memory	6 x 32GB Micron M393A1G40EB1-CRC, DDR4, 2933MHz Total 192GBs
Operating System	Fedora 33
BIOS	02.01.0012
Linux kernel version	5.8.15-300.fc33.x86_64
SPDK version	SPDK 21.01
Storage	**OS:** 1x 240GB INTEL SSDSC2BB240G6
NIC	1x 100GbE Mellanox ConnectX-5 Ex NIC. Both ports connected to Target server. (connected to CPU socket 0)

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

TCP configuration

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

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

Kernel & BIOS spectre-meltdown information

All three server systems use 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.

All tests done by Intel as of 03/01/2020.

Zero-copy send option

It should be noted that for this report zero-copy send option for SPDK NVMe-oF Target was explicitly enabled using SPDK RPC calls during the test's execution. Enabling this option allows for higher SPDK performance results and for easier comparison with previous SPDK NVMe-oF documents (20.07 and older), where this option was enabled by default.

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 (InfiniBand™, iWARP, RoCE), 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 close to the CPU.

The SPDK NVMe-oF target and initiator uses the underlying transport layer API which in case of TCP are POSIX sockets. In case of RDMA-capable NICs Infiniband/RDMA verbs API is used which should work on all flavors of RDMA transports but is currently tested against RoCEv2 and iWARP 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 systems (two used as initiators and one used as the target). The NVMe-oF target is connected to both initiator systems point-to-point using QSFP28 cables without any switches. The target system has sixteen Intel® SSD DC P4610 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®-5 100GbE NIC connected directly to the target without any switch.

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

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

1) Linux Kernel libaio engine
2) SPDK bdev engine

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

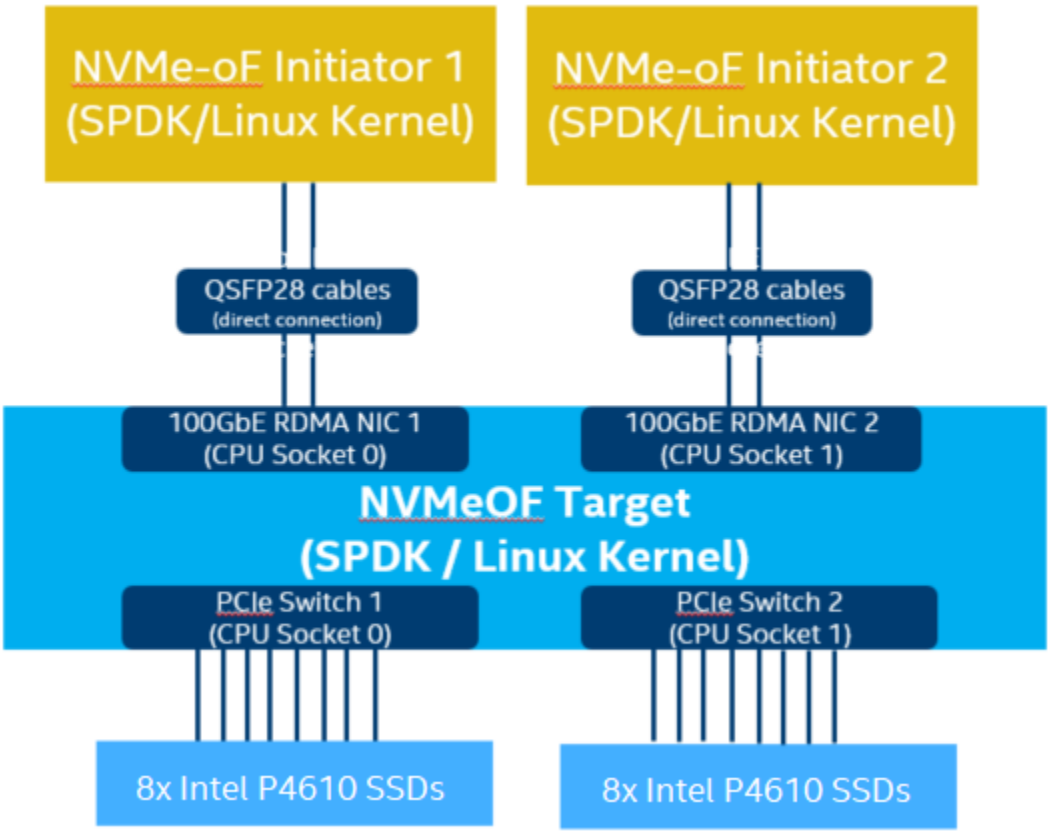


Figure 1: High-Level NVMe-oF TCP performance testing setup

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

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

The SPDK NVMe-oF TCP target was configured to run with 16 NVMe-oF subsystems. Each NVMe-oF subsystem ran on top of an individual NVMe bdev backed by a single Intel P4610 device. Each of the 2 host systems was connected to 8 NVMe-oF subsystems which were exported by the SPDK NVMe-oF Target over 1x 100GbE NIC. The SPDK bdev FIO plugin was used to target 8 NVMe-oF bdevs on each of the host. The SPDK Target was configured to use 1, 4, 8, 12, 16, 24, 32 and 40 CPU cores. We ran the following workloads on each initiator:

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

We scaled the fio jobs using fio parameter numjob=3 in order to generate more I/O requests. When using the SPDK fio plugin it is important to note the difference between the fio IO depth parameter and the NVMe device IO depth because we can configure an fio job to send IOs to more than one NVMe device and we can also scale the number of fio jobs using the numjobs parameter. The parameter values presented in the table below are actual queue depths used for each of the NVMe devices specified by the filename. 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.

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

Each workload was run three times at each CPU count and the reported results are the average of the 3 runs. For workloads which need preconditioning, 4KB rand read and 4KB 70% read 30% write we ran preconditioning once before running all of the workload to force the NVMe devices into a steady state so that we get consistent results.

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

Item	Description
Test Case	Test SPDK NVMe-oF Target I/O core scaling
SPDK NVMe-oF Target configuration	All the commands below were executed with spdk/scripts/rpc.py script. **Enable zero-copy send on Target side before initializing all other subsytems:** sock_impl_set_options –impl_name=posix –enable-zerocopy-send **Construct NVMe bdevs:** 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
	
	Create a TCP transport:
	nvmf_create_transport -t TCP
	{
	"trtype": "TCP",
	"max_queue_depth": 128,
	"max_io_qpairs_per_ctrlr": 127,
	"in_capsule_data_size": 4096,
	"max_io_size": 131072,
	"io_unit_size": 131072,
	"max_aq_depth": 128,
	"num_shared_buffers": 4096,
	"buf_cache_size": 32,
	"dif_insert_or_strip": false,
	"c2h_success": true,
	"sock_priority": 0,
	"abort_timeout_sec": 1
	}
	
	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 tcp \
	-f ipv4 -s 4420 -a ${ip}
	((i++))
	done
	done

SPDK NVMe-oF Initiator - FIO plugin configuration	**BDEV.conf:** See appendix A. **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, 64, 128, 192, 256} time_based=1 numjobs=3 ramp_time=60 runtime=300 [filename0] filename=Nvme0n1 [filename1] filename=Nvme1n1 [filename2] filename=Nvme2n1 [filename3] filename=Nvme3n1 [filename4] filename=Nvme4n1 [filename5] filename=Nvme5n1 [filename6] filename=Nvme6n1 [filename7] filename=Nvme7n1

4KB Random Read Results

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

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1882.75	482.0	4275.9
4 cores	8391.89	2148.3	958.1
8 cores	15863.13	4061.0	503.4
12 cores	19090.91	4887.3	417.6
16 cores	19892.17	5092.4	400.7
24 cores	20573.95	5266.9	386.6
32 cores	20455.01	5236.5	388.9
40 cores	20452.75	5235.9	388.9

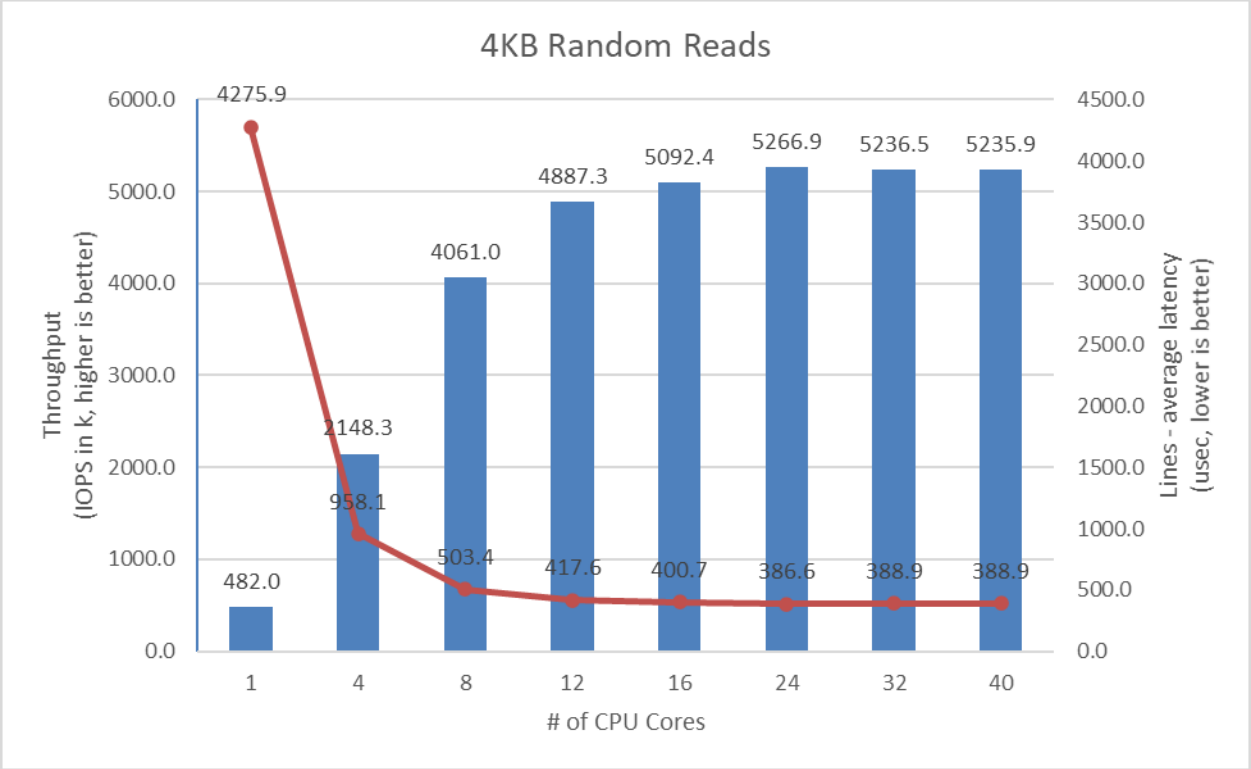


Figure 2: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Read workload at QD = 128

4KB Random Write Results

Disks were not preconditioned for this test case, which allows for higher IOPS numbers.

Table 7: SPDK NVMe-oF TCP Target Core Scaling results, Random Write IOPS, QD=192

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1161.58	297.4	10579.2
4 cores	4629.71	1185.2	2723.3
8 cores	8308.46	2127.0	1443.8
12 cores	11215.18	2871.1	1068.0
16 cores	14177.84	3629.5	844.1
24 cores	18963.97	4854.8	628.1
32 cores	20341.04	5207.3	584.0
40 cores	22001.05	5632.3	537.9

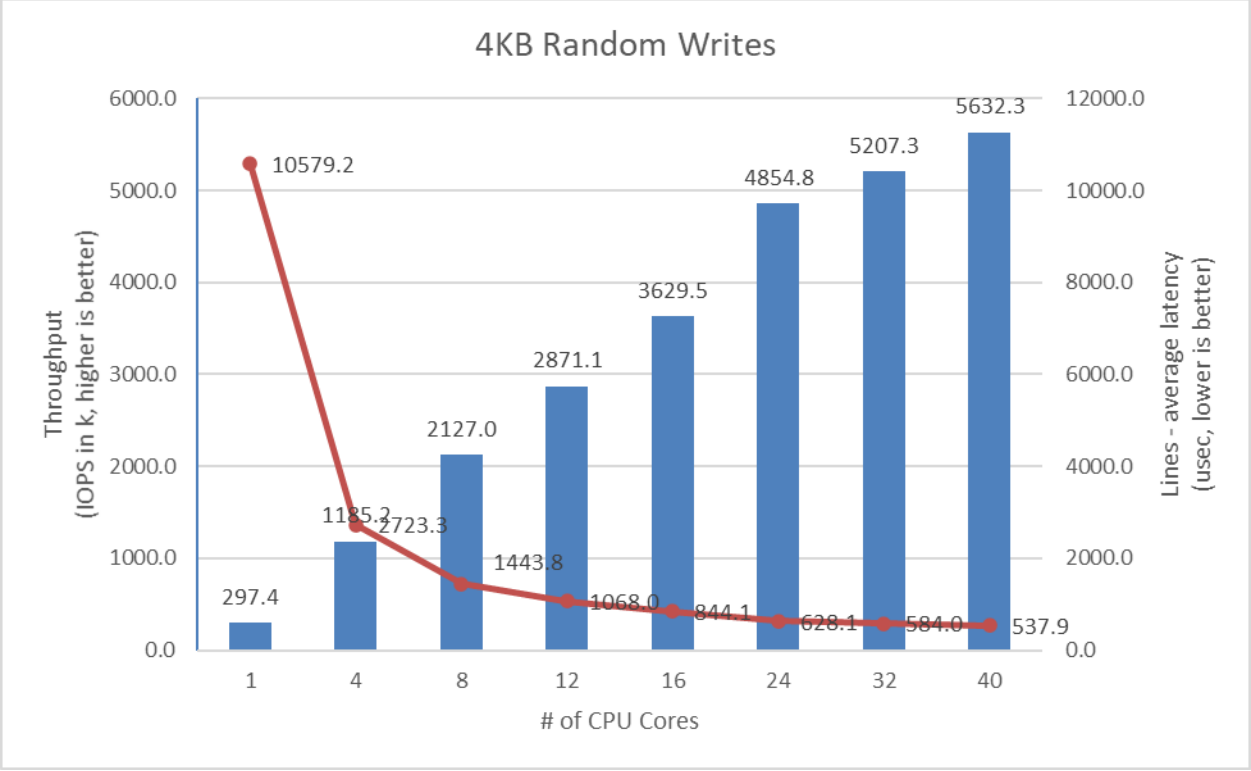


Figure 3: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write Workload at QD=192

4KB Random Read-Write Results

Table 8: SPDK NVMe-oF TCP Target Core Scaling results, Random Read/Write 70%/30% IOPS, QD=192

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1463.26	374.6	8194.5
4 cores	6004.37	1537.1	1995.3
8 cores	12072.53	3090.6	990.9
12 cores	16591.19	4247.3	719.2
16 cores	19917.76	5098.9	599.3
24 cores	21921.36	5611.9	542.8
32 cores	22238.66	5693.1	534.8
40 cores	22614.83	5789.4	525.7

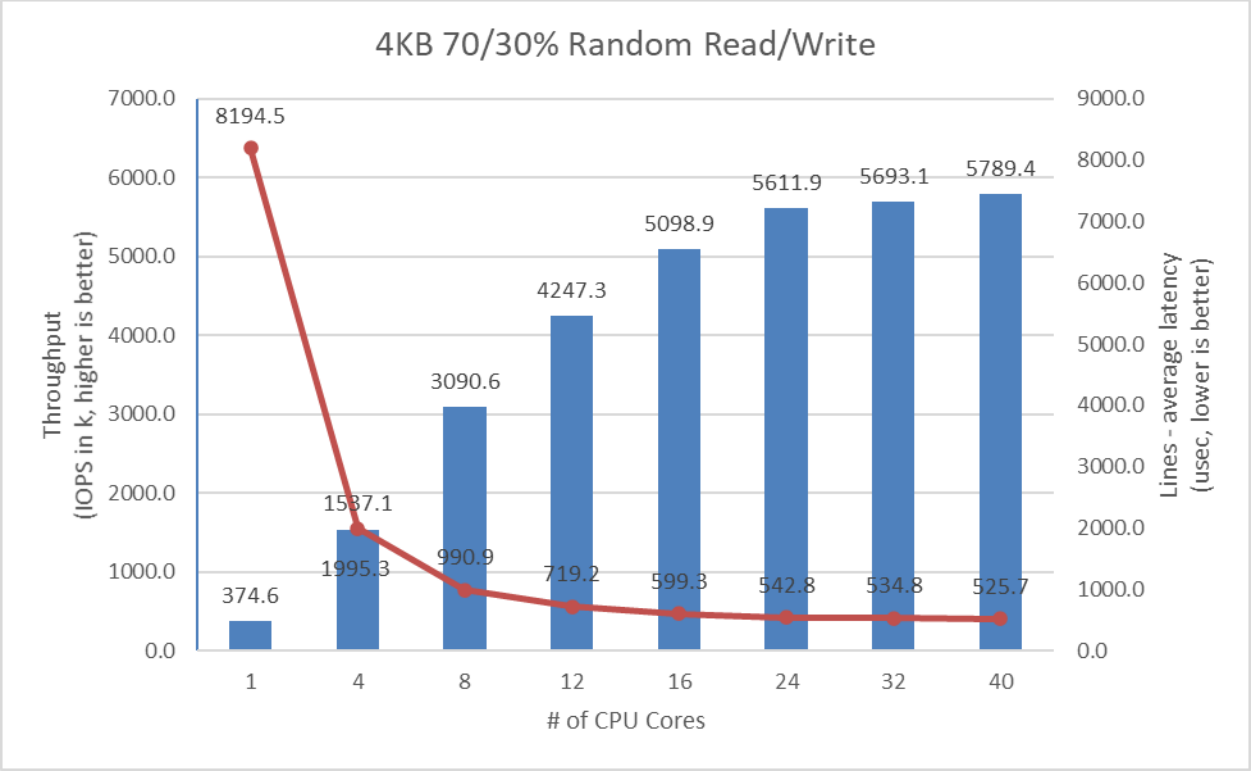


Figure 4: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 4KB Random 70/30 Read/Write workload at QD=192

Large Sequential I/O Performance

We measured the performance of large block I/O workloads by performing sequential I/Os of size 128KBs at queue depth 8. We used iodepth=8 because higher queue depth resulted in negligible bandwidth gain and a significant increase in the latency. The rest of the FIO configuration is similar to the 4KB test case in the previous part of this document.

Table 9: SPDK NVMe-oF TCP Target Core Scaling results, 128KB Sequential Read IOPS, QD=8

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	13519.22	108.2	1185.1
4 cores	23285.49	186.3	686.7
8 cores	23438.14	187.5	682.3
12 cores	23500.21	188.0	680.4
16 cores	23385.96	187.1	683.8
24 cores	23427.93	187.4	682.6
32 cores	23473.04	187.8	681.2
40 cores	23192.45	185.5	689.5

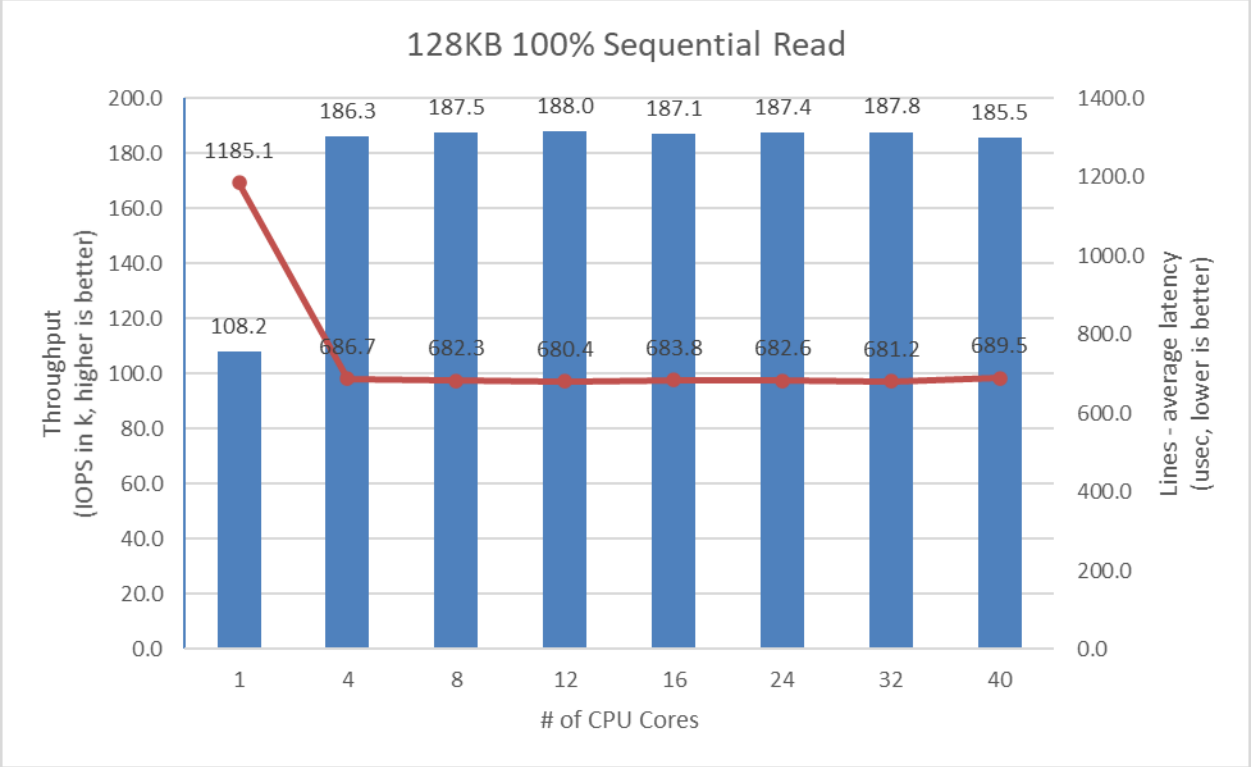


Figure 5: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 128KB 100% Sequential Read Workload at QD=8 and initiator FIO numjobs=2

Table 10: SPDK NVMe-oF TCP Target Core Scaling results, 128KB Sequential Write IOPS, QD=8

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	2463.49	19.7	6533.2
4 cores	9220.62	73.8	1740.4
8 cores	17702.41	141.6	903.5
12 cores	21726.60	173.8	736.2
16 cores	22415.41	179.3	715.5
24 cores	23311.40	186.5	686.0
32 cores	23372.26	187.0	684.3
40 cores	23168.85	185.4	690.3

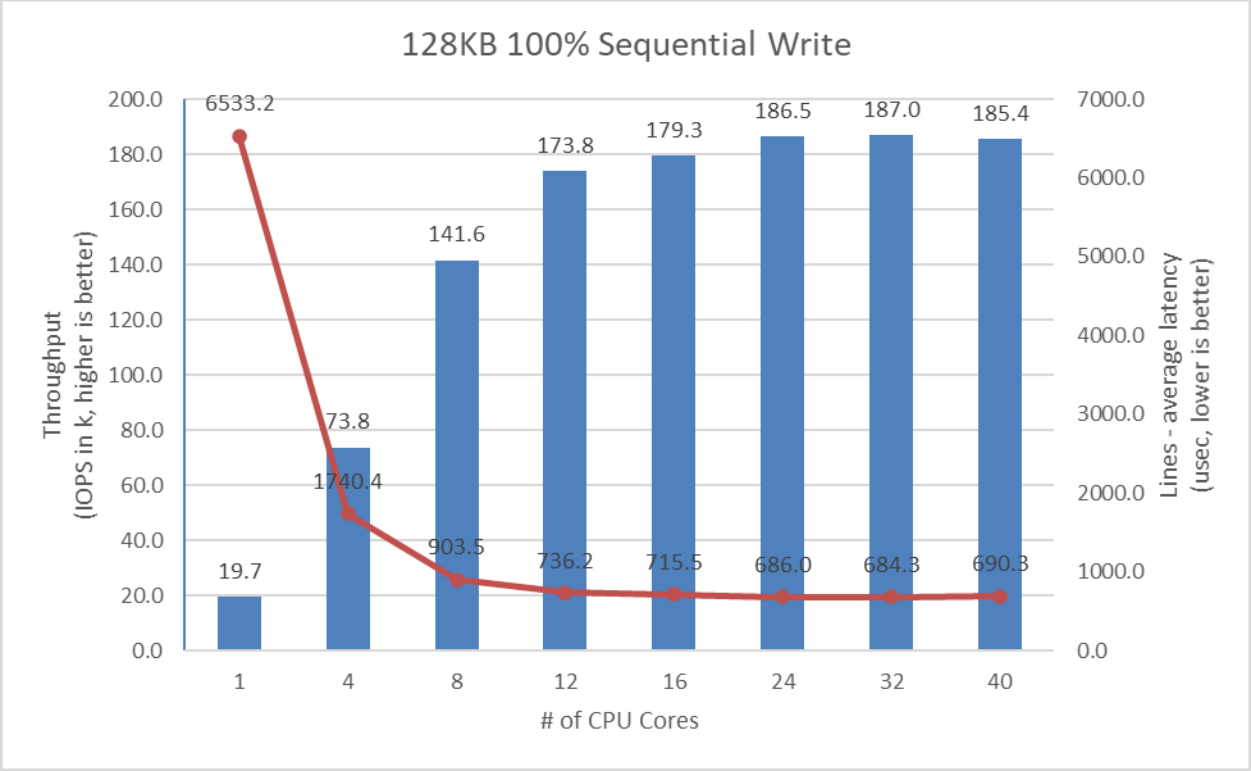


Figure 6: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 128KB 100% Sequential Write Workload at QD=8 and Initiator FIO numjobs=2

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

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	5278.43	42.2	1526.0
4 cores	17497.93	140.0	458.1
8 cores	23467.58	187.7	340.9
12 cores	25027.29	200.2	319.6
16 cores	26383.76	211.1	303.3
24 cores	26698.48	213.6	299.4
32 cores	26379.75	211.0	303.4
40 cores	24360.41	194.9	332.6

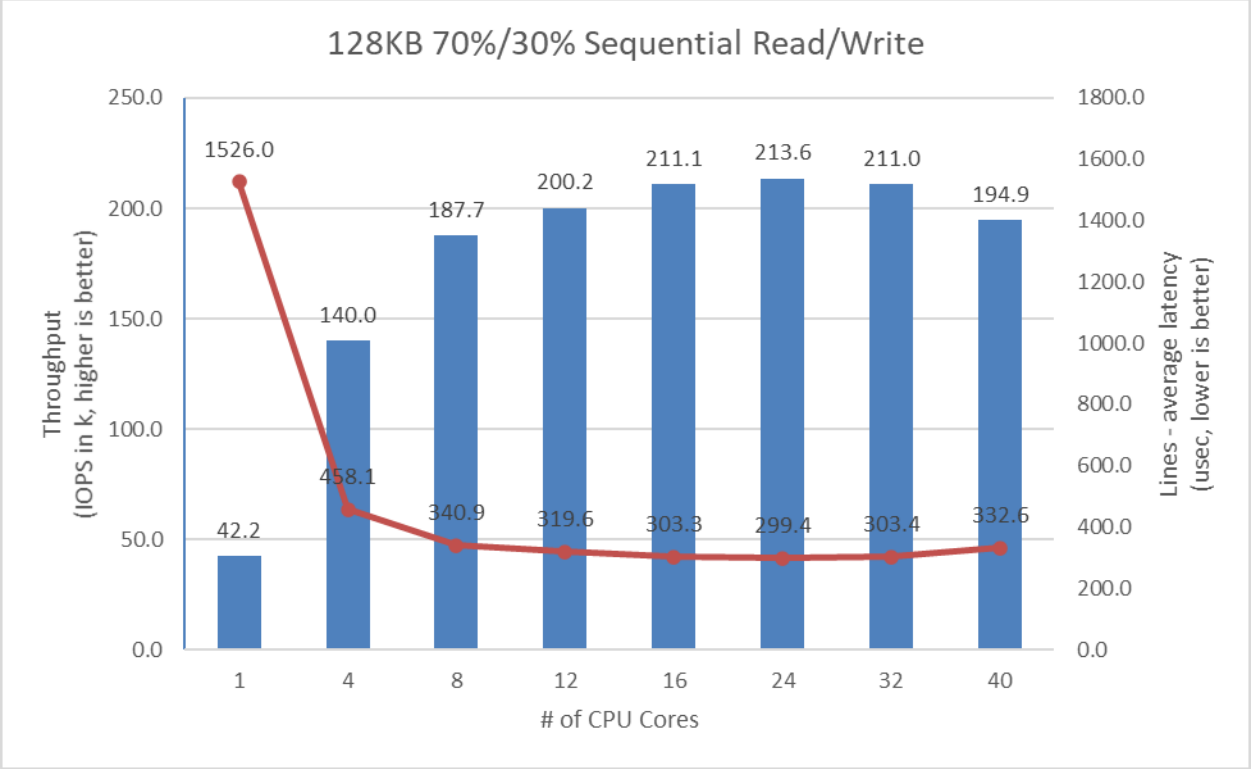


Figure 7: SPDK NVMe-oF TCP Target I/O core scaling: IOPS vs. Latency while running 128KB Sequential 70% Read 30% Write Workload at QD=8 and Initiator FIO numjobs=2

Conclusions

1. The SPDK NVMe-oF TCP Target IOPS throughput scales up linearly with addition of CPU cores for 4KB Random Read workload up to 8 CPU cores, reaching 140 GbE bandwidth. Adding more CPUs to Target configuration results in non-linear performance gains peaking at about 5.25 million IOPs at 24 CPU cores, almost completely saturating 200GbE network link.

2. For Random Write workload performance reaches 2.8 million IOPS, fully saturating a 100GbE link, at 12 CPU cores. It further scales linearly up to 24 CPU cores reaching 4.8 million IOPS. Beyond that point, performance scaling becomes nonlinear and reaches peak performance at 40CPU cores with 5.6 million IOPS which is close to saturating 200GbE link. There was no steady state or saturation reached during this test because we did not test with more than 40 CPU cores.

3. Random Read-Write workload scales linearly up to 16 CPU cores with 5.1 million IOPS and reaching peak performance of 5.8 million IOPS at 40 CPU cores.

4. The best trade-off between CPU efficiency and network saturation is when the Target is configured with between 8 and 12 CPU cores. The performance we achieved with these configurations fully saturated a 100Gbps NIC connection between Target and Initiator for all tested workloads.

5. For the 4KB Random Write workload, we saturated the Network because we did not precondition the drives. If preconditioned, the NVMe drives would max out at about 3.2 million IOPS. Not preconditioning the drives allowed us to artificially increase their throughput and serve more IO requests than usual.

6. The throughput of large block workloads scaled up with addition of CPU cores reaching peak performance at different CPU core counts. For the 128K Sequential reads workload, the peak throughput of 186 Gbps was observed at 4 CPU cores. For the 128K Sequential Writes, the throughput scaled linearly to 141 Gbps at 8 cores, however, beyond 8 cores the scaling was non-linear with a peak throughput of 186 Gbps observed at 24 CPU cores. For the 128K Sequential 70/30 Read/Write workload the scaling was non-linear, we observed 140 Gbps with just 4 cores and 187 Gbps at 8 cores, the peak throughput of 200 Gbps was observed at 12 CPU cores.

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

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

The test setup for this test case is slightly different than the set up described in introduction chapter, as we used just a single SPDK NVMe-oF TCP Initiator. The Initiator was connected to Target server with 100 Gbps network link.

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

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

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

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 filename. 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 TCP Initiator Core Scaling test configuration

Item	Description
Test Case	Test SPDK NVMe-oF TCP Initiator I/O core scaling
SPDK NVMe-oF Target configuration	Same as in Test Case #1, using 20 CPU cores.
SPDK NVMe-oF Initiator 1 - FIO plugin configuration	**BDEV.conf** 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} time_based=1 ramp_time=60 runtime=300 numjobs=1 [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 filename=Nvme4n1 filename=Nvme5n1 filename=Nvme6n1 filename=Nvme7n1 filename=Nvme8n1 filename=Nvme9n1 filename=Nvme10n1 filename=Nvme11n1 filename=Nvme12n1 filename=Nvme13n1 filename=Nvme14n1 filename=Nvme15n1
	FIO.conf **For X*4 CPU (up to 40) initiator configuration:** [global] ioengine=/tmp/spdk/examples/bdev/fio_plugin/fio_plugin spdk_conf=/tmp/spdk/bdev.conf thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={1, 32, 64, 128, 192} time_based=1 ramp_time=60 runtime=300 numjobs=X [filename0] filename=Nvme0n1 filename=Nvme1n1 filename=Nvme2n1 filename=Nvme3n1 [filename1] filename=Nvme4n1 filename=Nvme5n1 filename=Nvme6n1 filename=Nvme7n1 [filename2] filename=Nvme8n1 filename=Nvme9n1 filename=Nvme10n1 filename=Nvme11n1 [filename3] filename=Nvme12n1

	filename=Nvme13n1 filename=Nvme14n1 filename=Nvme15n1

4KB Random Read Results

Table 13: SPDK NVMe-oF TCP Initiator Core Scaling results, 4KB Random Read IOPS, QD=128

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1423.03	364.3	5584.4
4 cores	5641.96	1444.3	1403.9
8 cores	10005.27	2561.3	784.1
12 cores	11279.09	2887.4	698.1
16 cores	10768.69	2756.8	733.9
24 cores	9520.32	2437.2	831.4
32 cores	8647.29	2213.7	921.1
40 cores	8673.22	2220.3	915.6

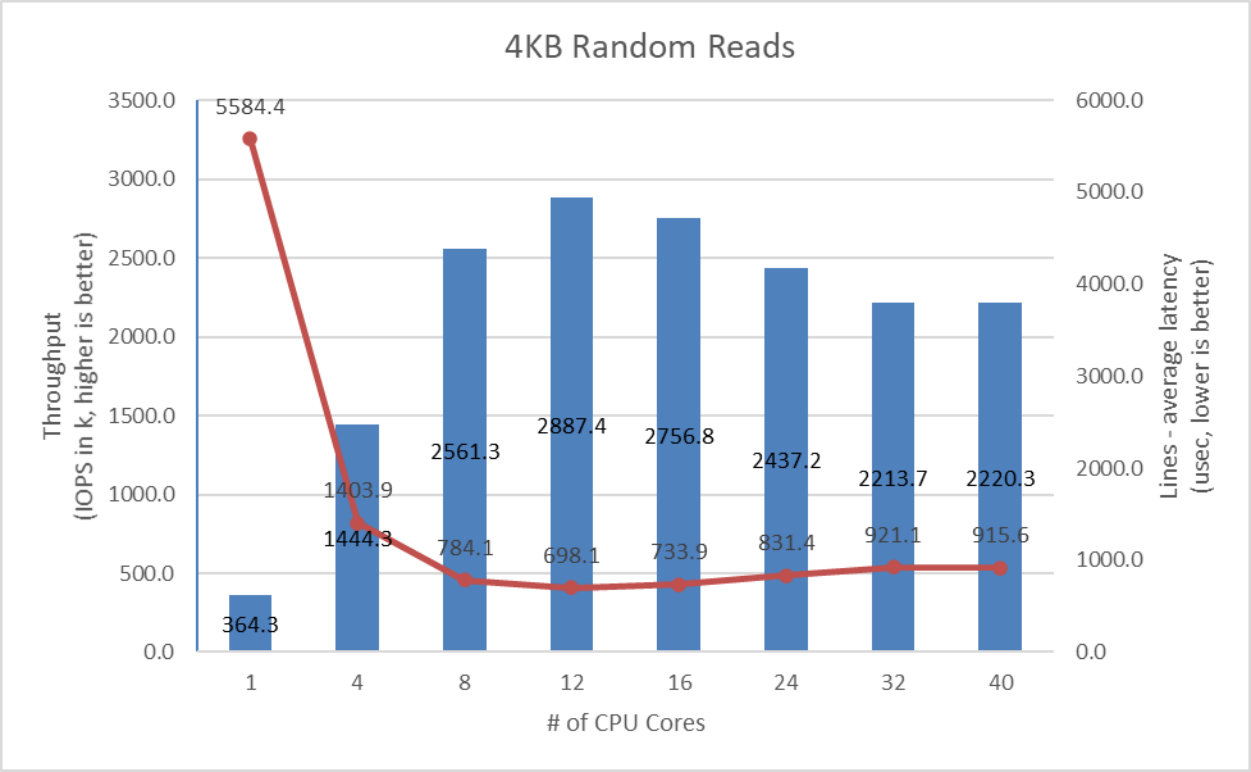


Figure 8: SPDK NVMe-oF TCP Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Read QD=128 workload

4KB Random Write Results

Table 14: SPDK NVMe-oF TCP Initiator Core Scaling results, 4KB Random Write IOPS, QD=128

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	2156.68	552.1	2274.1
4 cores	9114.38	2333.3	759.6
8 cores	11192.05	2865.2	699.7
12 cores	10797.68	2764.2	731.1
16 cores	10462.23	2678.3	758.9
24 cores	9465.66	2423.2	837.7
32 cores	8764.64	2243.7	909.5
40 cores	8346.26	2136.6	952.9

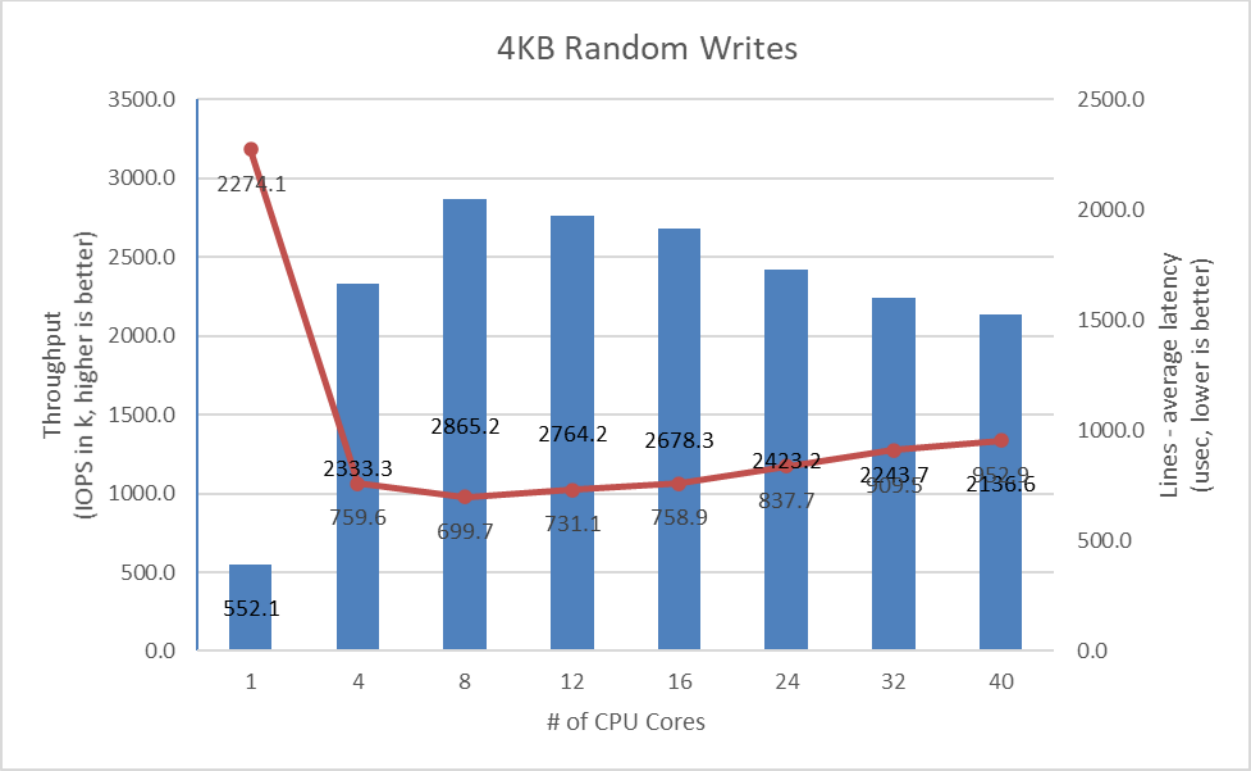


Figure 9: SPDK NVMe-oF TCP Initiator I/O core scaling: IOPS vs. Latency while running 4KB 100% Random Write Workload at QD=128

4KB Random Read-Write Results

Table 15: SPDK NVMe-oF TCP Initiator Core Scaling results, 4KB Random 70%/30% Read/Write IOPS, QD=128

# of Cores	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)
1 core	1525.33	390.5	5177.4
4 cores	6047.54	1548.2	1307.5
8 cores	10853.83	2778.6	719.4
12 cores	13495.60	3454.9	580.2
16 cores	12488.18	3197.0	633.2
24 cores	11927.03	3053.3	662.7
32 cores	11299.32	2892.6	703.3
40 cores	10583.37	2709.3	748.2

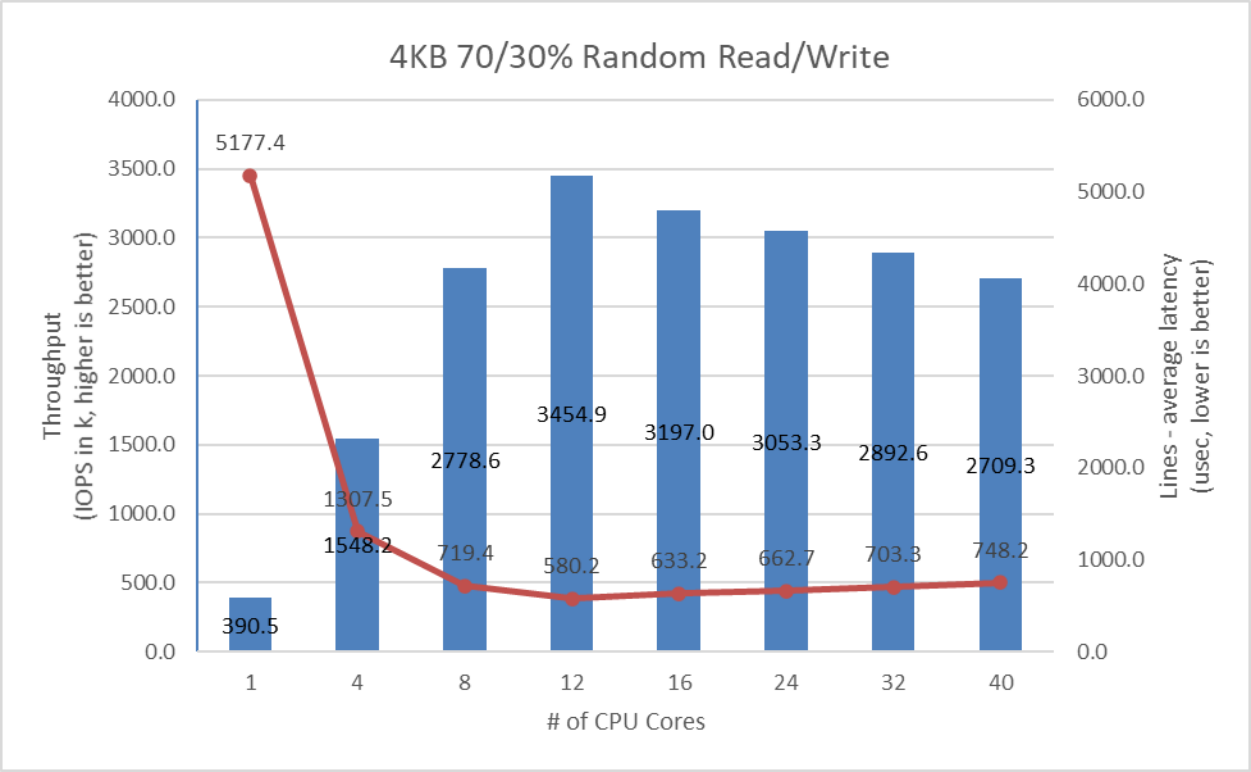


Figure 10: SPDK NVMe-oF TCP Initiator I/O core scaling: IOPS vs. Latency while running 4KB Random 70% Read 30% Write Workload at QD=128

Conclusions

1. For the 4KB Random Read workload, the SPDK NVMe-oF TCP Initiator performance scales linearly up to 8 CPU cores. The peak performance of 2.89 million IOPS was reached at 12 CPU cores, which saturates 100Gb link. Increasing the number of CPU cores beyond 12 CPU cores results in performance degradation.

2. In case of 4KB Random Write workload, performance scales linearly up to 4 CPU cores and reaches peak performance of 2.87 million IOPS at 8 cores, saturating 100Gb link. Increasing the number of Initiator cores beyond 8 cores does not improve results.

3. Mixed Random Read-Write workload performance scales linearly up to 8 CPU cores, reaching 2.78 million IOPS and reaches peak performance of 3.45 million IOPS at 12 CPU cores.

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

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

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

Item	Description
Test Case	Linux Kernel vs. SPDK NVMe-oF Latency
Test configuration	

SPDK NVMe-oF Target configuration	All below commands are executed with spdk/scripts/rpc.py script.
	
	nvmf_create_transport -t TCP
	(creates TCP transport layer with default values:
	{
	"trtype": "TCP",
	"max_queue_depth": 128,
	"max_io_qpairs_per_ctrlr": 127,
	"in_capsule_data_size": 4096,
	"max_io_size": 131072,
	"io_unit_size": 131072,
	"max_aq_depth": 128,
	"num_shared_buffers": 4096,
	"buf_cache_size": 32,
	"dif_insert_or_strip": false,
	"c2h_success": true,
	"sock_priority": 0,
	"abort_timeout_sec": 1
	}
	construct_null_bdev Nvme0n1 10240 4096
	nvmf_subsystem_create nqn.2018-09.io.spdk:cnode1 -s SPDK001 -a -m 8
	nvmf_subsystem_add_ns nqn.2018-09.io.spdk:cnode1 Nvme0n1
	nvmf_subsystem_add_listener nqn.2018-09.io.spdk:cnode1 -t tcp -f ipv4 -s 4420 -a 20.0.0.1

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

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

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

	iodepth=1 time_based=1 numjobs=1 ramp_time=60 runtime=300 [filename0] filename=/dev/nvme0n1

SPDK vs Kernel NVMe-oF Target Results

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

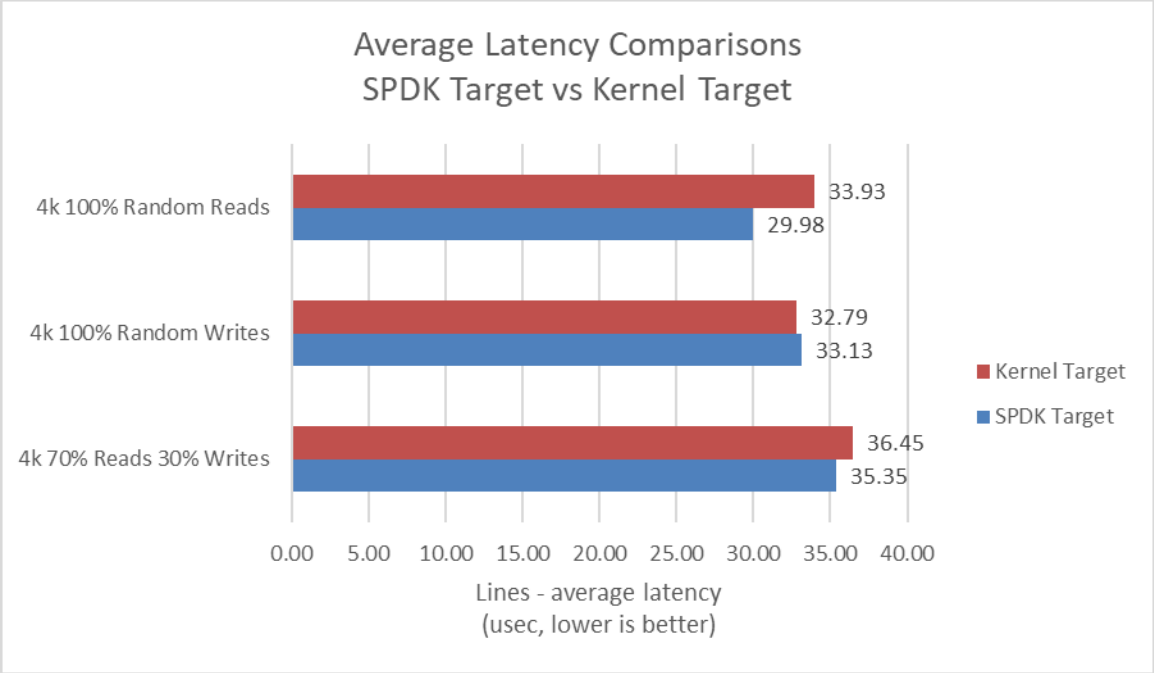


Figure 11: SPDK vs. Kernel NVMe-oF TCP Target Average I/O Latency for various workloads run using the Kernel Initiator

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

Access Pattern	Avg. Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KB 100% Random Reads IOPS	29.98	32999	47.9	80.0	130.4	162.1
4KB 100% Random Writes IOPS	33.13	29893	43.3	51.5	100.9	163.5
4KB 100% Random 70% Reads 30% Writes IOPS	35.35	27924	71.9	113.6	134.8	170.7

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

Access Pattern	Avg. Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KB 100% Random Reads IOPS	33.93	29138	52.5	72.9	89.9	109.1
4KB 100% Random Writes IOPS	32.79	30180	37.6	47.0	64.1	115.5
4KB 100% Random 70% Reads 30% Writes IOPS	36.45	27117	74.4	117.5	134.1	160.5

SPDK vs Kernel NVMe-oF TCP Initiator Results

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

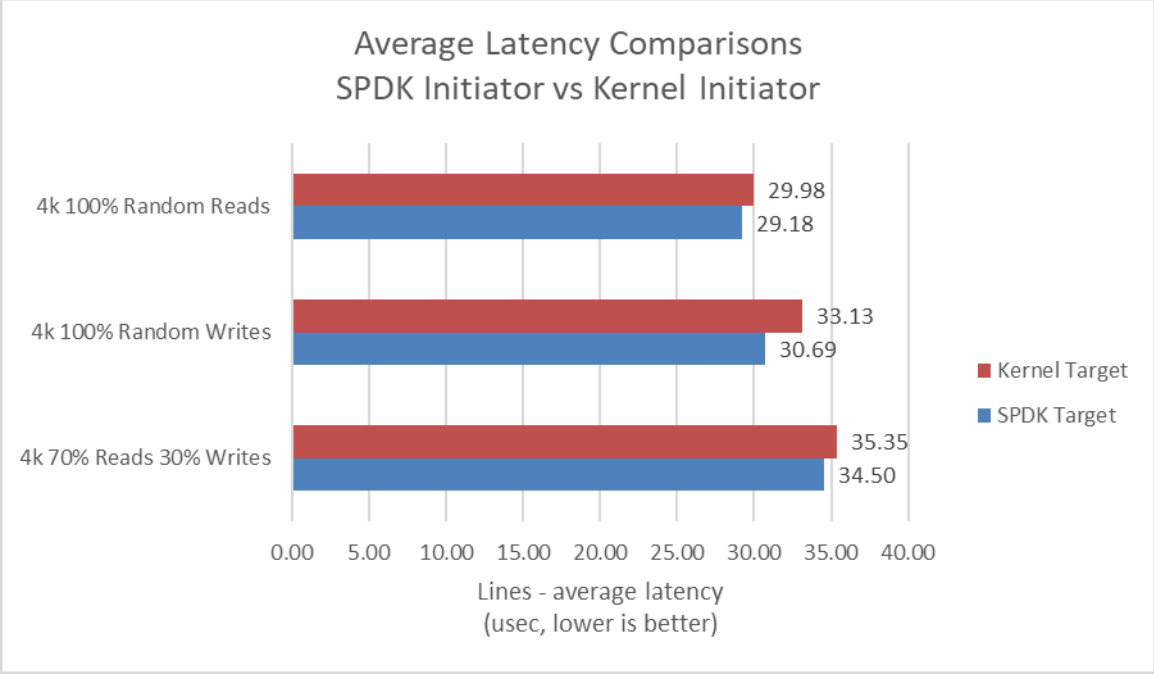


Figure 12: SPDK vs. Kernel NVMe-oF TCP Initiator Average I/O Latency for various workloads against SPDK Target

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

Access Pattern	Avg. Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KB 100% Random Reads IOPS	29.18	34007	39.3	46.8	68.4	160.1
4KB 100% Random Writes IOPS	30.69	32350	40.2	56.0	87.6	180.6
4KB 100% Random 70% Reads 30% Writes IOPS	34.50	28749	70.5	111.6	146.3	179.1

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

Access Pattern	Avg. Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KB 100% Random Reads IOPS	29.98	32999	47.9	80.0	130.4	162.1
4KB 100% Random Writes IOPS	33.13	29893	43.3	51.5	100.9	163.5
4KB 100% Random 70% Reads 30% Writes IOPS	35.35	27924	71.9	113.6	134.8	170.7

SPDK vs Kernel NVMe-oF Kernel + Initiator Results

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

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

Access Pattern	Avg. Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KB 100% Random Reads IOPS	29.18	34007	39.3	46.8	68.4	160.1
4KB 100% Random Writes IOPS	30.69	32350	40.2	56.0	87.6	180.6
4KB 100% Random 70% Reads 30% Writes IOPS	34.50	28749	70.5	111.6	146.3	179.1

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

Access Pattern	Avg. Latency (usec)	IOPS	p99 (usec)	p99.9 (usec)	p99.99 (usec)	p99.999 (usec)
4KB 100% Random Reads IOPS	33.93	29138	52.5	72.9	89.9	109.1
4KB 100% Random Writes IOPS	32.79	30180	37.6	47.0	64.1	115.5
4KB 100% Random 70% Reads 30% Writes IOPS	36.45	27117	74.4	117.5	134.1	160.5

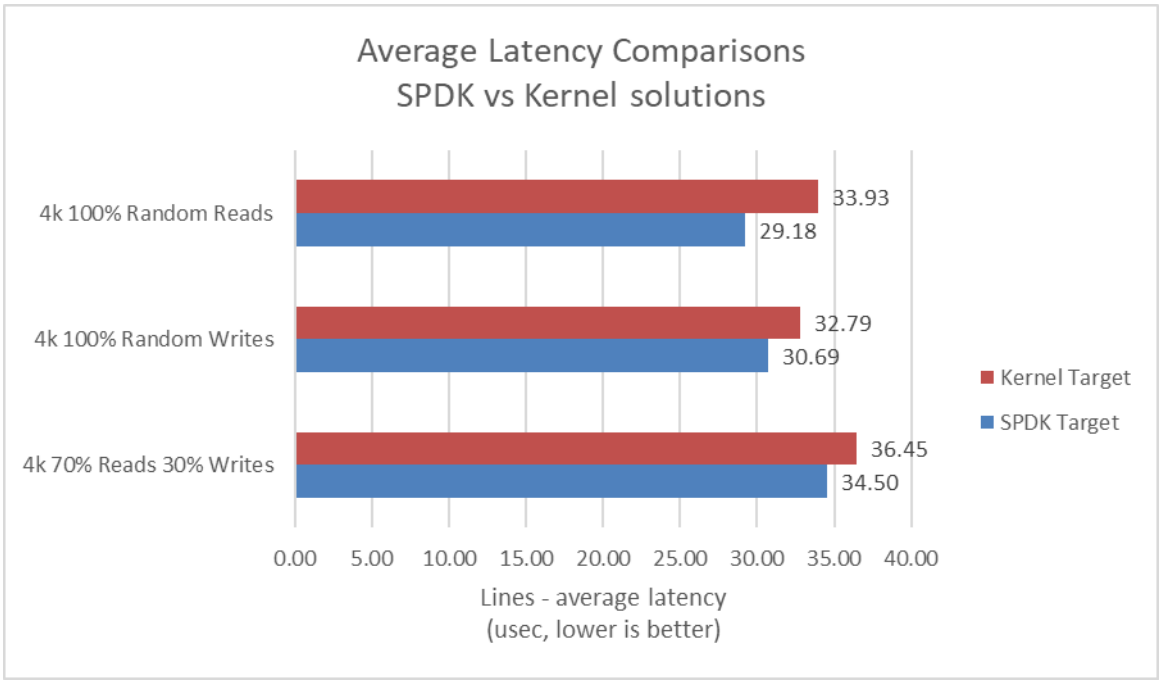


Figure 13: SPDK vs. Kernel NVMe-oF TCP solutions Average I/O Latency for various workloads

Conclusions

1. The SPDK NVMe-oF Target reduces the NVMe-oF average round trip I/O latency by up to 3.95 usec vs. the Linux Kernel NVMe-oF Target, which eliminates up to 11% NVMe-oF software overhead. It is worth noting that Kernel NVMe-oF Target was able to achieve slightly better latency result than SPDK while testing Random Write workload.

2. SPDK NVMe-oF Initiator reduces the average latency by up to 2.44 usec vs. the Linux Kernel NVMe-oF Initiator, which eliminates up to 7% NVMe-oF software overhead.

3. The SPDK NVMe-oF TCP target and initiator reduced the average latency by up to 11% vs. the Linux Kernel NVMe-oF target and initiator.

4. Latency-wise, the performance of SPDK NVMe-oF and Kernel NVMe-oF solutions is very comparable.

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. The number of connections (or I/O queue pairs) per NVMe-oF subsystem were varied and corresponding aggregated IOPS and number of CPU cores metrics were reported. The number of CPU cores metric was calculated from %CPU utilization measured using sar (systat package in Linux). The SPDK NVMe-oF Target was configured to run on 30 cores, 16 NVMe-oF subsystems (1 per Intel P4610) and 2 initiators were used both running I/Os to 8 separate subsystems using Kernel NVMe-oF initiator. We ran the following workloads on the host systems:

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

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

Item	Description
Test Case	NVMe-oF Target performance under varying # of connections
SPDK NVMe-oF Target configuration	Same as in Test Case #1, using 30 CPU cores.
Kernel NVMe-oF Target configuration	Target configuration file loaded using nvmet-cli tool. For detail configuration file contents please see Appendix D.
Kernel NVMe-oF Initiator #1	**Device config** Performed using nvme-cli tool. modprobe nvme-fabrics nvme connect –n nqn.2018-09.io.spdk:cnode1 –t tcp –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode2 –t tcp –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode3 –t tcp –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode4 –t tcp –a 20.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode5 –t tcp –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode6 –t tcp –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode7 –t tcp –a 20.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode8 –t tcp –a 20.0.1.1 –s 4420
Kernel NVMe-oF Initiator #2	**Device config** Performed using nvme-cli tool. modprobe nvme-fabrics nvme connect –n nqn.2018-09.io.spdk:cnode9 –t tcp –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode10 –t tcp –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode11 –t tcp –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode12 –t tcp –a 10.0.0.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode13 –t tcp –a 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode14 –t tcp –a 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode15 –t tcp –a 10.0.1.1 –s 4420 nvme connect –n nqn.2018-09.io.spdk:cnode16 –t tcp –a 10.0.1.1 –s 4420
FIO configuration (used on both initiators)	**FIO.conf** [global] ioengine=libaio

	thread=1 group_reporting=1 direct=1 norandommap=1 rw=randrw rwmixread={100, 70, 0} bs=4k iodepth={8, 32, 64, 128} time_based=1 ramp_time=60 runtime=300 numjobs={1, 4, 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 [filename8] filename=/dev/nvme7n1

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

The metrics in the graph represent relative efficiency in 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.

4KB Random Read Results

Figure 14: Relative Efficiency Comparison of Linux Kernel vs. SPDK NVMe-oF Target IOPS/Core for 4KB 100% Random Reads QD=128 using the Kernel Initiator

Table 24: Linux Kernel NVMe-oF TCP Target: 4KB 100% Random Reads, QD=128

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	9912.09	2537.5	806.8	15.1
4	15830.59	4052.6	504.9	41.2
16	14170.60	3627.7	563.4	61.6

Table 25: SPDK NVMe-oF TCP Target: 4KB 100% Random Reads, QD=128

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	10746.04	2751.0	744.2	30.0
4	17308.38	4430.9	461.8	30.0
16	13876.12	3552.3	575.4	30.0

4KB Random Write Results

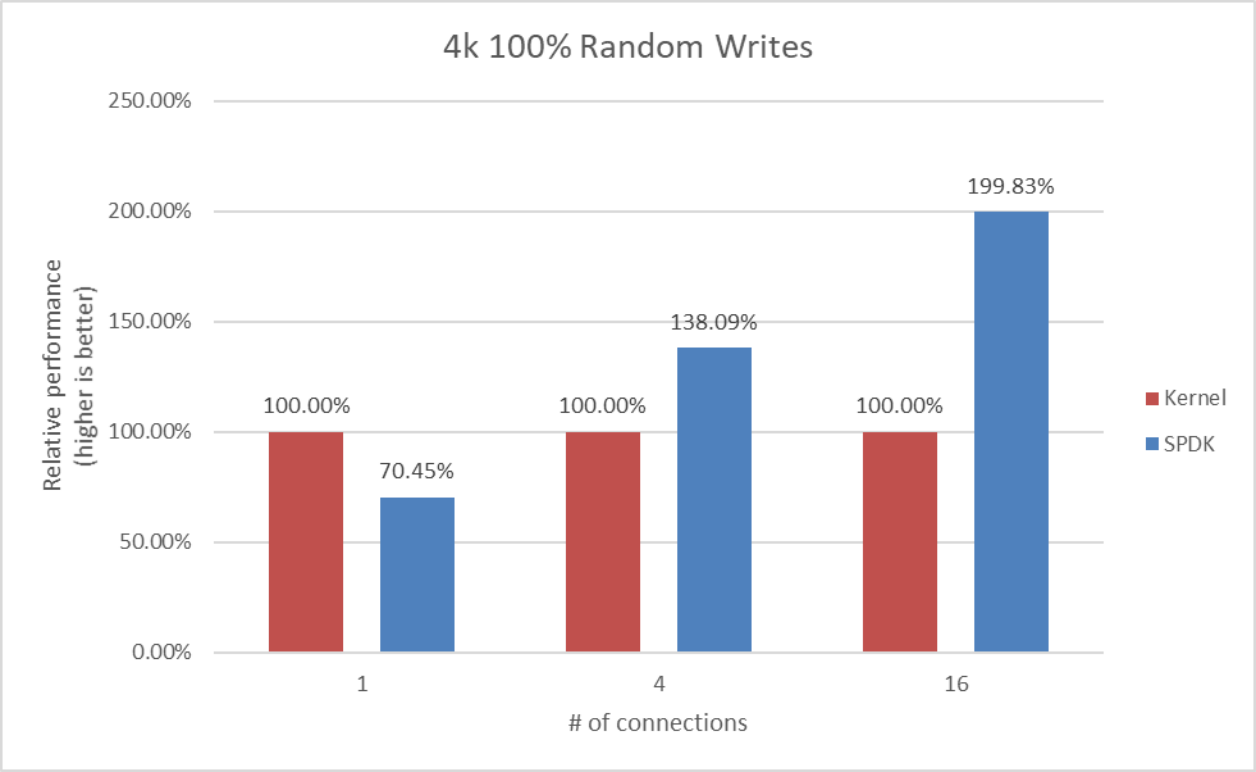


Figure 15: Relative Efficiency Comparison of Linux Kernel vs. SPDK NVMe-oF Target IOPS/Core for 4KB 100% Random Writes QD=128 using the Kernel Initiator

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

Table 26: Linux Kernel NVMe-oF TCP Target: 4KB 100% Random Writes, QD=64

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	9482.89	2427.6	843.6	17.7
4	14994.34	3838.5	538.0	41.4
16	15395.90	3941.3	519.0	65.9

Table 27: SPDK NVMe-oF TCP Target: 4KB 100% Random Writes, QD=64

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	11349.75	2905.5	704.6	30.0
4	15008.42	3842.1	537.4	30.0
16	14001.69	3584.4	570.9	30.0

4KB Random Read-Write Results

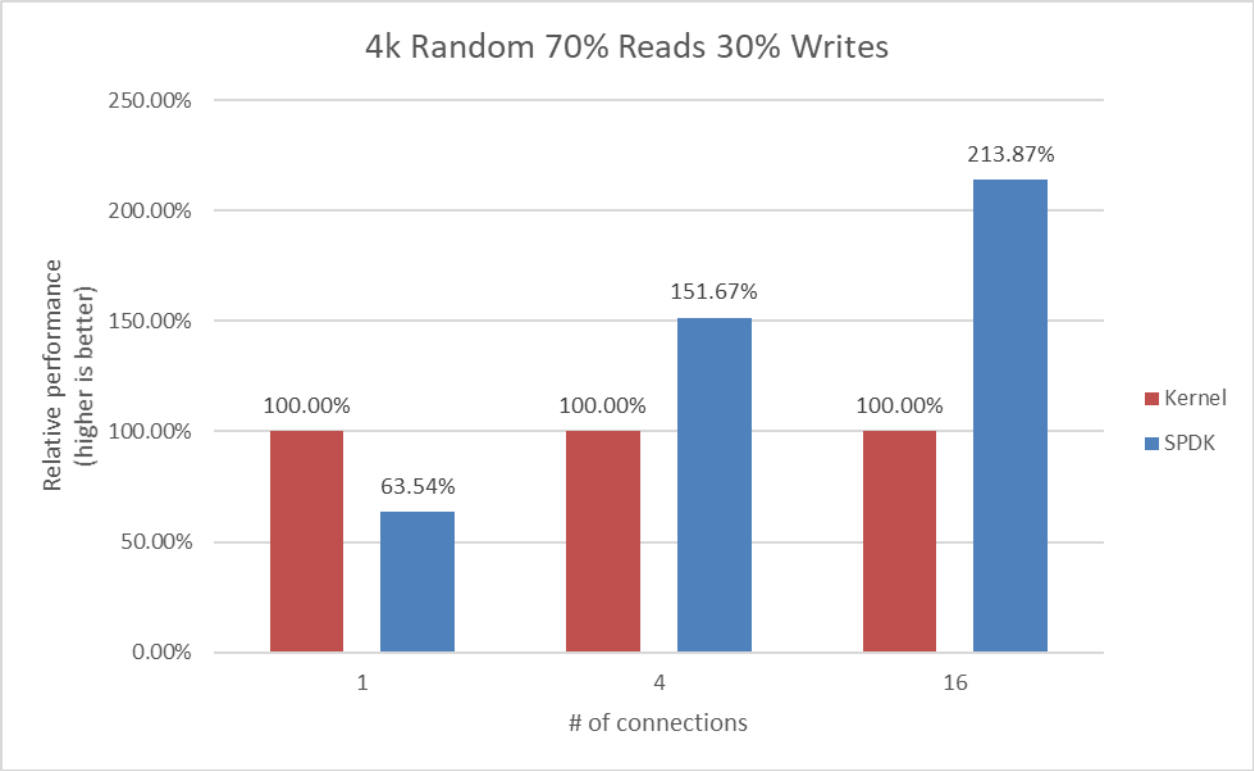


Figure 16: Relative Efficiency Comparison of Linux Kernel vs. SPDK NVMe-oF Target IOPS/Core for 4KB Random 70% Reads 30% Writes QD=128 using Kernel Initiator

Table 28: Linux Kernel NVMe-oF TCP Target: 4KB 70% Random Read 30% Random Write, QD=128

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	10209.40	2613.6	783.4	16.8
4	15667.39	4010.8	510.2	44.5
16	14567.93	3729.4	548.1	67.3

Table 29: SPDK NVMe-oF TCP Target: 4KB 70% Random Read 30% Random Write, QD=128

Connections per subsystem	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
1	11554.21	2957.9	692.2	30.0
4	16033.95	4104.7	498.6	30.0
16	13896.70	3557.5	575.0	30.0

Low Connections Results

During testing it was observed that relative efficiency of SPDK Target is about 40% of Kernel Target because SPDK uses a fixed number of CPU cores that was configured at start up and does not have a mechanism to decrease the number of I/O cores on the fly if the SPDK target does not need all of the CPU resources.

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

Table 30: SPDK & Kernel NVMe-oF TCP Target relative efficiency comparison for various workloads, QD=64, 1 connection per subsystem

Workload	Target	Bandwidth (MBps)	Throughput (IOPS k)	Avg. Latency (usec)	# CPU Cores
Random Read	Linux	8558.36	2190.9	467.1	14.6
	SPDK	9104.31	2330.7	439.2	4.0
Random Write	Linux	8702.89	2227.9	459.5	16.8
	SPDK	9429.14	2413.9	424.0	4.0
Random Read/Write	Linux	7978.45	2042.5	501.1	11.5
	SPDK	8836.74	2262.2	452.5	4.0

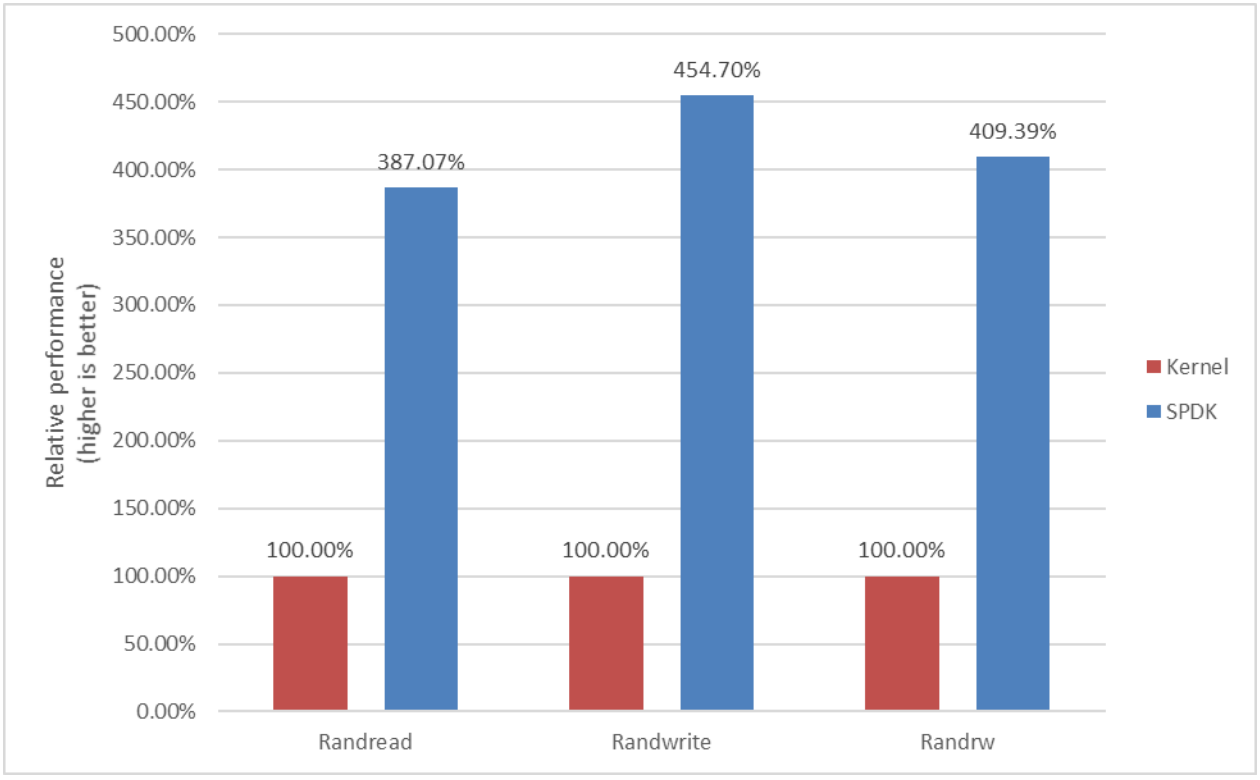


Figure 17: Relative Efficiency Comparison of Linux Kernel vs. SPDK NVMe-oF Target IOPS/Core for various workloads, 1 connection per subsystem and reduced number of SPDK Target CPU Cores (4)

Conclusions

1. The Linux Kernel NVMe-oF TCP target relative efficiency in IOPS/Core was better than SPDK when there was just one connection per subsystem because the SPDK NVMe-oF target uses a fixed number of CPU cores and there is no mechanism to dynamically decrease the number of CPU cores if the target does not need all the cores. Therefore, we re-run the test cases with 1 connections per subsystem but lowered the number of I/O cores used by the SPDK Target to 4 and added the results to the tables which show a relative performance up to 3.9x-4.5x times better than Linux Kernel NVMe-oF TCP target.

2. The performance peaked for all workloads for both SPDK and Kernel NVMe-oF TCP Targets at 4 connections per subsystem.

3. The SPDK NVMe-oF TCP target relative efficiency in IOPS/Core was about 1.5 times better than the Linux Kernel NVMe-oF target at 4 connections per NVMe-oF subsystem about 2 times better at 16 connections per NVMe-oF subsystem.

Summary

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

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

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

Throughput scales up and latency decreases almost linearly with the scaling of SPDK NVMe-oF target I/O cores when serving 4KB random workloads. The SPDK NVMe-oF target saturates a 100 Gbps network link using just 8 CPU cores for the 4KB Random Read and 4KB Random Read/Write 70/30 workloads, and 12 CPU cores for the 4KB Random Write workload. The IOPS scalability remains close to linear for all workloads until the results are close to saturating network 200 Gbps link.

For the SPDK NVMe-oF TCP Initiator, the IOPS throughput scales almost linearly with addition of CPU cores until the network was almost saturated. Further increasing the number of CPU cores results in performance degradation. A single initiator was able to saturate a 100Gb link.

For the NVMe-oF TCP latency comparison, the SPDK NVMe-oF Target and Initiator average latency is up to about 10% and 7% lower than their Linux Kernel counterparts respectively when testing against null block device-based backend.

The SPDK NVMe-oF TCP Target performed up to 2.1 times better w.r.t IOPS/core than Linux Kernel NVMe-oF target while running 4KB 100% Random Read workload with increasing number of connections per NVMe-oF subsystem.

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

List of Figures

List of Tables

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

Initiator system 1

```
{
  "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"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme1",
            "trtype": "tcp",
            "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": "tcp",
            "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": "tcp",
            "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": "tcp",
            "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": "tcp",
            "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": "tcp",
            "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": "tcp",
            "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": "tcp",
      "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": "tcp",
      "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": "tcp",
      "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": "tcp",
      "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": "tcp",
      "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": "tcp",
            "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": "tcp",
            "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": "tcp",
            "traddr": "10.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode7",
            "adrfam": "IPv4"
          }
        }
      ]
    }
  ]
}
```

Appendix B – Test Case 2 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"
          }
        },
        {
```

```
  "method": "bdev_nvme_attach_controller",
  "params": {
    "name": "Nvme1",
    "trtype": "tcp",
    "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": "tcp",
    "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": "tcp",
    "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": "tcp",
    "traddr": "20.0.0.1",
    "trsvcid": "4420",
    "subnqn": "nqn.2018-09.io.spdk:cnode4",
    "adrfam": "IPv4"
  }
},
{
  "method": "bdev_nvme_attach_controller",
  "params": {
    "name": "Nvme5",
    "trtype": "tcp",
    "traddr": "20.0.0.1",
    "trsvcid": "4420",
    "subnqn": "nqn.2018-09.io.spdk:cnode5",
    "adrfam": "IPv4"
  }
},
{
  "method": "bdev_nvme_attach_controller",
  "params": {
```

```
      "name": "Nvme6",
      "trtype": "tcp",
      "traddr": "20.0.0.1",
      "trsvcid": "4420",
      "subnqn": "nqn.2018-09.io.spdk:cnode6",
      "adrfam": "IPv4"
    }
  },
  {
    "method": "bdev_nvme_attach_controller",
    "params": {
      "name": "Nvme7",
      "trtype": "tcp",
      "traddr": "20.0.0.1",
      "trsvcid": "4420",
      "subnqn": "nqn.2018-09.io.spdk:cnode7",
      "adrfam": "IPv4"
    }
  },
  {
    "method": "bdev_nvme_attach_controller",
    "params": {
      "name": "Nvme8",
      "trtype": "tcp",
      "traddr": "20.0.1.1",
      "trsvcid": "4420",
      "subnqn": "nqn.2018-09.io.spdk:cnode8",
      "adrfam": "IPv4"
    }
  },
  {
    "method": "bdev_nvme_attach_controller",
    "params": {
      "name": "Nvme9",
      "trtype": "tcp",
      "traddr": "20.0.1.1",
      "trsvcid": "4420",
      "subnqn": "nqn.2018-09.io.spdk:cnode9",
      "adrfam": "IPv4"
    }
  },
  {
    "method": "bdev_nvme_attach_controller",
    "params": {
      "name": "Nvme10",
      "trtype": "tcp",
      "traddr": "20.0.1.1",
      "trsvcid": "4420",
      "subnqn": "nqn.2018-09.io.spdk:cnode10",
      "adrfam": "IPv4"
    }
  },
  {
    "method": "bdev_nvme_attach_controller",
    "params": {
      "name": "Nvme11",
      "trtype": "tcp",
```

```
            "traddr": "20.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode11",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme12",
            "trtype": "tcp",
            "traddr": "20.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode12",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme13",
            "trtype": "tcp",
            "traddr": "20.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode13",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme14",
            "trtype": "tcp",
            "traddr": "20.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode14",
            "adrfam": "IPv4"
          }
        },
        {
          "method": "bdev_nvme_attach_controller",
          "params": {
            "name": "Nvme15",
            "trtype": "tcp",
            "traddr": "20.0.1.1",
            "trsvcid": "4420",
            "subnqn": "nqn.2018-09.io.spdk:cnode15",
            "adrfam": "IPv4"
          }
        }
      ]
    }
  ]
}
```

Appendix C – 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 D – Kernel NVMe-oF TCP Target configuration

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

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

```
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.0.1",
        "trsvcid": "4422",
        "trtype": "tcp"
      },
      "portid": 3,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode3"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.0.1",
        "trsvcid": "4423",
        "trtype": "tcp"
      },
      "portid": 4,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode4"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.1.1",
        "trsvcid": "4424",
        "trtype": "tcp"
      },
      "portid": 5,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode5"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.1.1",
        "trsvcid": "4425",
        "trtype": "tcp"
      },
      "portid": 6,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode6"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
```

```
        "traddr": "20.0.1.1",
        "trsvcid": "4426",
        "trtype": "tcp"
      },
      "portid": 7,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode7"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "20.0.1.1",
        "trsvcid": "4427",
        "trtype": "tcp"
      },
      "portid": 8,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode8"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.0.1",
        "trsvcid": "4428",
        "trtype": "tcp"
      },
      "portid": 9,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode9"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.0.1",
        "trsvcid": "4429",
        "trtype": "tcp"
      },
      "portid": 10,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode10"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.0.1",
        "trsvcid": "4430",
        "trtype": "tcp"
      },
      "portid": 11,
```

```
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode11"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.0.1",
        "trsvcid": "4431",
        "trtype": "tcp"
      },
      "portid": 12,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode12"
      ]
    },
{
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4432",
        "trtype": "tcp"
      },
      "portid": 13,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode13"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4433",
        "trtype": "tcp"
      },
      "portid": 14,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode14"
      ]
    },
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4434",
        "trtype": "tcp"
      },
      "portid": 15,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode15"
      ]
    },
```

```
    {
      "addr": {
        "adrfam": "ipv4",
        "traddr": "10.0.1.1",
        "trsvcid": "4435",
        "trtype": "tcp"
      },
      "portid": 16,
      "referrals": [],
      "subsystems": [
        "nqn.2018-09.io.spdk:cnode16"
      ]
    }
  ],
"hosts": [],
 "subsystems": [
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme0n1",
            "uuid": "b53be81d-6f5c-4768-b3bd-203614d8cf20"
          },
          "enable": 1,
          "nsid": 1
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode1"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme1n1",
            "uuid": "12fcf584-9c45-4b6b-abc9-63a763455cf7"
          },
          "enable": 1,
          "nsid": 2
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode2"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
```

```
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme2n1",
            "uuid": "ceae8569-69e9-4831-8661-90725bdf768d"
          },
          "enable": 1,
          "nsid": 3
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode3"
    },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme3n1",
          "uuid": "39f36db4-2cd5-4f69-b37d-1192111d52a6"
        },
        "enable": 1,
        "nsid": 4
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode4"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme4n1",
          "uuid": "984aed55-90ed-4517-ae36-d3afb92dd41f"
        },
        "enable": 1,
        "nsid": 5
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode5"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
```

```
          "device": {
            "path": "/dev/nvme5n1",
            "uuid": "d6d16e74-378d-40ad-83e7-b8d8af3d06a6"
          },
          "enable": 1,
          "nsid": 6
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode6"
    },
   {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme6n1",
            "uuid": "a65dc00e-d35c-4647-9db6-c2a8d90db5e8"
          },
          "enable": 1,
          "nsid": 7
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode7"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme7n1",
            "uuid": "1b242cb7-8e47-4079-a233-83e2cd47c86c"
          },
          "enable": 1,
          "nsid": 8
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode8"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme8n1",
            "uuid": "f12bb0c9-a2c6-4eef-a94f-5c4887bbf77f"
```

```
        },
        "enable": 1,
        "nsid": 9
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode9"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme9n1",
          "uuid": "40fae536-227b-47d2-bd74-8ab76ec7603b"
        },
        "enable": 1,
        "nsid": 10
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode10"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme10n1",
          "uuid": "b9756b86-263a-41cf-a68c-5cfb23c7a6eb"
        },
        "enable": 1,
        "nsid": 11
      }
    ],
    "nqn": "nqn.2018-09.io.spdk:cnode11"
  },
  {
    "allowed_hosts": [],
    "attr": {
      "allow_any_host": "1",
      "version": "1.3"
    },
    "namespaces": [
      {
        "device": {
          "path": "/dev/nvme11n1",
          "uuid": "9d7e74cc-97f3-40fb-8e90-f2d02b5fff4c"
        },
        "enable": 1,
        "nsid": 12
```

```
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode12"
    },
{
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme12n1",
            "uuid": "d3f4017b-4f7d-454d-94a9-ea75ffc7436d"
          },
          "enable": 1,
          "nsid": 13
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode13"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme13n1",
            "uuid": "6b9a65a3-6557-4713-8bad-57d9c5cb17a9"
          },
          "enable": 1,
          "nsid": 14
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode14"
    },
    {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme14n1",
            "uuid": "ed69ba4d-8727-43bd-894a-7b08ade4f1b1"
          },
          "enable": 1,
          "nsid": 15
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode15"
```

```
    },
  {
      "allowed_hosts": [],
      "attr": {
        "allow_any_host": "1",
        "version": "1.3"
      },
      "namespaces": [
        {
          "device": {
            "path": "/dev/nvme15n1",
            "uuid": "5b8e9af4-0ab4-47fb-968f-b13e4b607f4e"
          },
          "enable": 1,
          "nsid": 16
        }
      ],
      "nqn": "nqn.2018-09.io.spdk:cnode16"
    }
  ]
}
```

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.

§